

# **Security in IT Networks: Multilateral Security in Distributed and by Distributed Systems**

**Script for the lectures**

**“Security and Cryptography” I+II**

Andreas Pfitzmann

February 27, 2012



# Contents

<b>1</b>	<b>Introduction</b>	<b>17</b>
1.1	What are computer networks (distributed open systems)? . . . . .	17
1.2	What does security mean? . . . . .	22
1.2.1	What is to be protected . . . . .	23
1.2.2	Against whom is to be protect . . . . .	25
1.2.3	How and through what can security be achieved? . . . . .	30
1.2.4	Preview of precautions . . . . .	30
1.2.5	Attacker model . . . . .	32
1.3	What does security in computer networks mean? . . . . .	34
1.4	Why multilateral security . . . . .	35
<b>2</b>	<b>Security in Single Computers</b>	<b>37</b>
2.1	Physical security assumptions . . . . .	37
2.1.1	What can we expect at best? . . . . .	37
2.1.2	Design of security measures . . . . .	38
2.1.3	A negative example: chip cards . . . . .	39
2.1.4	Useful physical security measures . . . . .	40
2.2	Protection of isolated computers from unauthorized access and computer viruses . . . . .	40
2.2.1	Identification . . . . .	41
2.2.2	Admission control . . . . .	43
2.2.3	Access control . . . . .	44
2.2.4	Limitation of the thread through “computer viruses” towards the one of “transitive Trojan Horses” . . . . .	45
2.2.5	Remaining problems . . . . .	47
<b>3</b>	<b>Cryptography Basics</b>	<b>49</b>
3.1	Systematics . . . . .	49
3.1.1	Cryptographic systems and their key-distribution . . . . .	51
3.1.1.1	Cryptographic Systems, Overview . . . . .	52
3.1.1.2	Authentication-systems, overview . . . . .	55
3.1.2	Further annotations to the key-distribution . . . . .	59
3.1.2.1	To whom are keys assigned? . . . . .	60
3.1.2.2	How many keys need to be exchanged? . . . . .	60
3.1.2.3	Security of the key-distribution limits the cryptographically reachable security . . . . .	61

## Contents

3.1.3	Basics of security . . . . .	62
3.1.3.1	Attack-aims and -successes . . . . .	62
3.1.3.2	Types of attacks . . . . .	63
3.1.3.3	Connection between observing/modifying and passive/-active attacker . . . . .	65
3.1.3.4	Basics of “cryptographically strong” . . . . .	66
3.1.3.5	Overview of the cryptographic systems presented in the following . . . . .	69
3.1.4	Hybrid cryptographic systems . . . . .	70
3.2	One-Time Pad . . . . .	70
3.3	Authentication Codes . . . . .	76
3.4	The $s^2 - \text{mod} - n$ -pseudo-random-bitsequence-generator . . . . .	79
3.4.1	Basics for systems with factorizing assumptions . . . . .	79
3.4.1.1	Computation modulo $\mathbf{n}$ . . . . .	83
3.4.1.2	The number of elements in $\mathbb{Z}_n^*$ and its relevance for exponentiation . . . . .	86
3.4.1.3	The relation between $\mathbb{Z}_n$ and $\mathbb{Z}_p, \mathbb{Z}_q$ ( $p \neq q$ , $p, q$ being prime numbers, $n := p * q, \dots$ ) . . . . .	87
3.4.1.4	Squares and roots in general . . . . .	88
3.4.1.5	Squares and roots $(\text{mod } p)$ . . . . .	89
3.4.1.6	Squares and roots $(\text{mod } p)$ for $p \equiv 3 \pmod{4}$ . . . . .	90
3.4.1.7	Squares and roots $(\text{mod } n)$ <u>with</u> knowledge of $p, q$ . . . . .	91
3.4.1.8	Squares and roots $(\text{mod } n)$ <u>with</u> knowledge of $p, q \equiv 3 \pmod{4}$ . . . . .	92
3.4.1.9	Squares and roots $(\text{mod } n)$ <u>without</u> knowledge of $p, q$ . . . . .	93
3.4.2	Requirements concerning the Pseudo-Random Bit Generator . . . . .	96
3.4.3	The $s^2 - (\text{mod } n)$ -generator . . . . .	99
3.4.4	$s^2 - (\text{mod } n)$ -generator used as an asymmetric encryption system	101
3.5	GMR: A strong cryptographic signature system . . . . .	104
3.5.1	Basic function: collision resistant permutation pairs . . . . .	104
3.5.2	Mini-GMR for a message containing only one bit . . . . .	107
3.5.3	Basic signatures: big tuples of collision resistant permutations . . . . .	108
3.5.4	Complete GMR: authentication of many references . . . . .	109
3.5.4.1	Further efficiency improvements . . . . .	112
3.6	RSA: The most known system for asymmetric encryption and digital signatures . . . . .	113
3.6.1	The asymmetric cryptographic system RSA . . . . .	114
3.6.2	Naive and insecure usage of RSA . . . . .	116
3.6.2.1	RSA as asymmetric encryption system . . . . .	117
3.6.2.2	RSA as digital signature system . . . . .	118
3.6.3	Attacks, especially multiplicative attacks from Davida and Moore	118
3.6.3.1	RSA as a digital signature system . . . . .	119
3.6.3.2	RSA as asymmetric encryption system . . . . .	120
3.6.4	Thwarting the attack . . . . .	121

3.6.4.1	RSA as asymmetric encryption system . . . . .	121
3.6.4.2	RSA as digital signature system . . . . .	122
3.6.5	Efficient implementations of RSA . . . . .	123
3.6.5.1	Public exponent with almost only zeros . . . . .	123
3.6.5.2	The owner of the secret key calculates the factors modulo	125
3.6.5.3	Encryption performance . . . . .	126
3.6.6	Security and usage of RSA . . . . .	126
3.7	DES: The most known system for symmetric concealation and authentication . . . . .	127
3.7.1	Overview of DES . . . . .	127
3.7.2	An iteration round . . . . .	128
3.7.3	The decryption principle of DES . . . . .	128
3.7.4	The encryption function . . . . .	130
3.7.5	Partial key generation . . . . .	131
3.7.6	The complementary property of DES . . . . .	131
3.7.7	Generalization of DES: G-DES . . . . .	133
3.7.8	Decryption performance . . . . .	135
3.8	Cryptographic systems in operation . . . . .	135
3.8.1	Block Ciphers, Stream Ciphers . . . . .	136
3.8.2	Operation modes of block ciphers . . . . .	137
3.8.2.1	Electronic code book (ECB) . . . . .	138
3.8.2.2	Cipher block chaining (CBC) . . . . .	140
3.8.2.3	Cipher Feedback (CFB) . . . . .	142
3.8.2.4	Output feedback (OFB) . . . . .	145
3.8.2.5	Plain cipher block chaining (PCBC) . . . . .	147
3.8.2.6	Output cipher feedback (OCFB) . . . . .	150
3.8.2.7	Annotations to expanding block ciphers and memory initialization . . . . .	151
3.8.2.8	Summary on the properties of operation modes . . . . .	152
3.8.3	Construction of a collision-resistant hash function from block ciphers	154
3.9	Outline of further systems . . . . .	156
3.9.1	Diffie-Hellman key exchange . . . . .	156
3.9.2	For the signee unconditional secure signatures . . . . .	159
3.9.3	Unconditional secure pseudo-signatures . . . . .	163
3.9.4	Undeniable signatures . . . . .	163
3.9.5	Blind signatures . . . . .	164
3.9.6	Threshold scheme . . . . .	165
<b>4</b>	<b>Basics in Steganography</b>	<b>169</b>
4.1	Systematics . . . . .	172
4.1.1	Stenographic systems . . . . .	172
4.1.1.1	Stenographic secrecy-systems, overview . . . . .	172
4.1.1.2	Stenographic authentication-systems, overview . . . . .	173
4.1.1.3	Comparison . . . . .	174

## Contents

4.1.2	An note on key-distribution: steganography with public keys . . . . .	175
4.1.3	Basics of security . . . . .	175
4.1.3.1	Attack-aims and -successes . . . . .	175
4.1.3.2	Types of attacks . . . . .	175
4.1.3.3	Stegoanalysis leads to improved steganography . . . . .	175
4.1.3.3.1	construction of $S$ and $S^{-1}$ . . . . .	176
4.1.3.3.2	construction of $E$ and $E^{-1}$ . . . . .	177
4.2	Information-theoretic secure steganography . . . . .	180
4.2.1	Information theoretic secure steganographic secrecy . . . . .	180
4.2.2	Information-theoretic secure steganographic authentication . . . . .	181
4.3	Two opposed stego-paradigms . . . . .	181
4.3.1	Changing as little as possible . . . . .	181
4.3.2	Simulating the normal process . . . . .	181
4.4	Parity-encoding to improve concealment . . . . .	181
4.5	Steganography in digital telephone calls . . . . .	181
4.6	Steganography in pictures . . . . .	181
4.7	Steganography in video conferences . . . . .	181
<b>5</b>	<b>Security in Communication Networks</b> . . . . .	<b>183</b>
5.1	The problem . . . . .	183
5.1.1	The social problem . . . . .	183
5.1.2	information-theoretical problem and solution drafts . . . . .	185
5.2	Usage and limits of encryption in communication networks . . . . .	188
5.2.1	Usage of encryption in communication networks . . . . .	188
5.2.1.1	Link by link encryption . . . . .	189
5.2.1.2	End-to-end encryption . . . . .	189
5.2.2	Limits of encryption in communication networks . . . . .	192
5.3	Basic measures settled outside the communication network . . . . .	194
5.3.1	Public nodes . . . . .	194
5.3.2	Time independent processing . . . . .	194
5.3.3	Local choice . . . . .	195
5.4	Basic measures settled inside the communication network . . . . .	195
5.4.1	Broadcasting: The protection of the receiver . . . . .	196
5.4.1.1	Modifying attacks on broadcasting . . . . .	197
5.4.1.2	Implicit addressing . . . . .	197
5.4.1.2.1	Implementation of invisible implicit addressing . . . . .	198
5.4.1.2.2	Equivalence of invisible and implicit addressing and encryption . . . . .	198
5.4.1.2.3	Implementation of visible and implicit addressing . . . . .	199
5.4.1.2.4	Address distribution . . . . .	199
5.4.1.3	Fault tolerance and modifying attacks on broadcasting . . . . .	200
5.4.2	Query and superpose: Protecting the receiver . . . . .	201
5.4.3	Meaningless messages: Weak protection of the sender and the receiver . . . . .	203

5.4.4	Unobservability of bordering connections and stations as well as digital signal regeneration: Protection of the sender . . . . .	204
5.4.4.1	Circular cabling (RING-Network) . . . . .	205
5.4.4.1.1	An efficient 2-anonymous ring access system . . . . .	206
5.4.4.1.2	Fault tolerance in RING-Networks . . . . .	208
5.4.4.2	Collision avoiding tree network (TREE-Network) . . . . .	212
5.4.5	Superposed sending (DC-network): Protection of the sender . . . . .	213
5.4.5.1	A first outline . . . . .	213
5.4.5.2	Definition and Proof of sender anonymity . . . . .	216
5.4.5.3	Receiver's anonymity achieved by the snap-snatch-distribution . . . . .	219
5.4.5.3.1	Comparing the global result of superpositions . . . . .	220
5.4.5.3.2	Deterministic snap-snatch-key generation . . . . .	221
5.4.5.3.3	Probabilistic snap-snatch-key generation . . . . .	223
5.4.5.4	Superposed receiving . . . . .	223
5.4.5.4.1	Criteria for the preservation of anonymity and unlinkability . . . . .	224
5.4.5.4.2	Algorithm for resolution of collisions with averaging and superposed receiving . . . . .	225
5.4.5.4.3	Booking-scheme (Roberts' scheme) . . . . .	229
5.4.5.5	Optimality, complexity and implementations . . . . .	230
5.4.5.6	Fault tolerance of the DC-network . . . . .	239
5.4.6	MIX–networks: protection of communication relations . . . . .	245
5.4.6.1	Fundamental considerations about possibilities and limits of re-encryption . . . . .	248
5.4.6.2	Sender anonymity . . . . .	250
5.4.6.3	Receiver's anonymity . . . . .	252
5.4.6.4	Mutual anonymity . . . . .	257
5.4.6.5	Re-encryption maintaining length of messages . . . . .	258
5.4.6.6	Efficient avoidance of re-encryption repeatings . . . . .	267
5.4.6.7	Short preview . . . . .	269
5.4.6.8	Necessary properties of the asymmetric encryption system and breaking the direct RSA–implementation . . . . .	269
5.4.6.9	Faster transmission by using MIX-channels . . . . .	273
5.4.6.10	Fault-tolerance at MIX–nets . . . . .	278
5.4.6.10.1	Different MIX–sequences . . . . .	279
5.4.6.10.2	Substitution of MIXs . . . . .	281
5.4.6.10.2.1	The coordination problem . . . . .	282
5.4.6.10.2.2	MIXs with reserve–MIXs . . . . .	283
5.4.6.10.2.3	Leaving out a MIX . . . . .	285
5.4.7	Tolerating modifying attacks on RING-, DC- and MIX-Networks	288
5.4.8	Active chaining attacks over limited resources . . . . .	292
5.4.9	Classification within a layer model . . . . .	293
5.4.10	Comparison of strength and effort of RING-, DC, and MIX-Network	297

## Contents

5.5	Upgrading towards an integrated broadband network . . . . .	299
5.5.1	Telephone MIXes . . . . .	299
5.5.1.1	Basics of time slice channels . . . . .	301
5.5.1.2	Structure of the time slice channels . . . . .	302
5.5.1.3	Establishing an unobservable connection . . . . .	304
5.5.1.4	Shutting down an unobservable connection . . . . .	307
5.5.1.5	Accounting the network usage . . . . .	307
5.5.1.6	Connections between LSCs with MIX cascades and conventional LSCs . . . . .	308
5.5.1.7	Reliability . . . . .	309
5.5.2	Effort . . . . .	310
5.5.2.1	Parameters for describing the cryptographic systems needed . . . . .	310
5.5.2.1.1	Symmetrical Concealment Systems . . . . .	311
5.5.2.1.2	Asymmetrical Concealment Systems . . . . .	311
5.5.2.1.3	Hybrid Encryption . . . . .	311
5.5.2.2	Minimal duration of a time slice . . . . .	312
5.5.2.2.1	Length of a MIX message . . . . .	312
5.5.2.2.2	Estimation . . . . .	313
5.5.2.3	Maximum amount of interfaces to be served by one MIX-ing cascade . . . . .	314
5.5.2.4	Computation complexity for the interfaces and MIXes . . . . .	315
5.5.2.5	Connection startup duration . . . . .	315
5.6	Network management . . . . .	316
5.6.1	Operating a Network: Responsibility for service quality vs. threats of Trojan horses . . . . .	316
5.6.1.1	End-to-End encryption and distribution . . . . .	317
5.6.1.2	RING- and TREE-Networks . . . . .	317
5.6.1.3	DC-Network . . . . .	319
5.6.1.4	query and superpose as well as MIX-Networks . . . . .	319
5.6.1.5	Combination as well as heterogeneous networks . . . . .	320
5.6.1.6	Connected, hierarchical Networks . . . . .	320
5.6.2	Accounting . . . . .	321
5.7	Public Mobile Radio . . . . .	322
<b>6</b>	<b>Value Exchange and Payment Systems</b>	<b>327</b>
6.1	Anonymity of participants . . . . .	327
6.2	Legal security of business processes while preserving anonymity . . . . .	329
6.2.1	Declarations of intention . . . . .	330
6.2.1.1	Anonymous declaring or receiving of declarations . . . . .	330
6.2.1.2	Authentication of declarations . . . . .	330
6.2.1.2.1	Digital signatures . . . . .	330
6.2.1.2.2	Types of authentication . . . . .	332
6.2.2	Other actions . . . . .	334
6.2.3	Securing of evidence . . . . .	334

6.2.4	Legal investigations . . . . .	336
6.2.5	Settlement of damages . . . . .	336
6.3	Fault secure value exchange . . . . .	338
6.3.1	A third party guarantees de-anonymization . . . . .	338
6.3.2	Trustees guarantee fraud protection for anonymous partners . . . . .	340
6.4	Anonymous digital payment systems . . . . .	344
6.4.1	Basic scheme of a secure and anonymous digital payment system	345
6.4.2	Restriction of anonymity through predefined accounts . . . . .	350
6.4.3	Suggestions from literature . . . . .	351
6.4.4	Marginal conditions for anonymous payment systems . . . . .	353
6.4.4.1	Transfer between payment systems . . . . .	353
6.4.4.2	Interest on the balance and accommodation of loans . . . . .	354
6.4.5	Secure devices as witnesses . . . . .	354
6.4.6	Payment systems with security based on the possibility of making someone public . . . . .	356
<b>7</b>	<b>Credentials</b>	<b>359</b>
<b>8</b>	<b>Multiparty Computation Protocols</b>	<b>361</b>
<b>9</b>	<b>Controlability of Security Technologies</b>	<b>365</b>
9.1	User requirements on signature- and encryption keys . . . . .	366
9.2	Digital signatures allow the exchange of encryption keys . . . . .	367
9.3	Key-Escrow Systems can be used for an undetectable key exchange . . . . .	368
9.4	Symmetric authentication allows concealment . . . . .	369
9.5	Steganography: Finding confidential message is impossible . . . . .	371
9.6	Measures to take after key loss . . . . .	372
9.7	Conclusions . . . . .	373
<b>10</b>	<b>Multilateral Secure IT Systems</b>	<b>375</b>
<b>11</b>	<b>Glossary</b>	<b>377</b>
<b>A</b>	<b>Exercises</b>	<b>409</b>
1	Exercises for “Introduction” . . . . .	409
2	Exercises for “Security in single computers and its limits” . . . . .	411
3	Exercises for “Cryptography” . . . . .	413
4	Exercises for “Basics of Steganography” . . . . .	430
5	Exercises for chapter 5 “Security in communication networks” . . . . .	434
6	Exercises for “value exchange and payment systems” . . . . .	443
9	Exercises for “Regulation of security technologies” . . . . .	444
<b>B</b>	<b>Answers</b>	<b>447</b>
B.1	Answers for “Introduction” . . . . .	447
B.2	Answers for “Security of single computers and its limits” . . . . .	454

*Contents*

B.3	Answers for “Cryptography” . . . . .	459
B.4	Answers for “Basics of Steganography” . . . . .	524
B.5	Answers for chapter “Security in communication networks” . . . . .	529
B.6	Answers for “Value exchange and payment systems” . . . . .	532
B.9	Answers for “Regulation of security technologies” . . . . .	534

# List of Figures

1.1	A sector of a network . . . . .	20
1.2	Development of the wire-based communication networks of the Deutsche Bundespost Telekom (now: Deutsche Telekom AG), as planned in 1984 and largely executed . . . . .	21
1.3	People use software programs to develop other software programs and computers . . . . .	26
1.4	Transitive distribution of errors and attacks . . . . .	27
1.5	Universal Trojan Horse . . . . .	29
1.6	Which security measures are safe from which attackers? . . . . .	31
1.7	Observing vs. modifying attacker . . . . .	33
2.1	The five basic functions arranged in circular layers (upper left figure) and three possibilities of iterated layers. . . . .	39
2.2	Identification of humans by IT systems . . . . .	41
2.3	Identification of IT systems by humans . . . . .	42
2.4	Identification of IT systems by IT systems . . . . .	43
2.5	Access control by an access monitor . . . . .	44
2.6	Computer virus, transitive Trojan Horse and restriction of security threats caused by computer viruses to threats caused by transitive Trojan Horses	46
3.1	Basic scheme of cryptographic systems shown with the example of a symmetric encryption system . . . . .	52
3.2	Symmetric encryption system (black box with a lock; two identical keys) . . . . .	52
3.3	Key exchange for symmetric encryption systems . . . . .	54
3.4	Asymmetric encryption system . . . . .	55
3.5	Key distribution for asymmetric encryption systems by a public-key register	56
3.6	Symmetric authentication system ( $\approx$ Glass box with a lock, there are two identical keys for putting things in) . . . . .	56
3.7	Digital signature system (Glass box with a lock; there is only one key for putting things in) . . . . .	57
3.8	Key distribution for digital signature system by a public-key register . . . . .	58
3.9	Goals and key distribution of asymmetric encryption system, symmetric cryptographic system and digital signature system . . . . .	60
3.10	Generation of a random number $z$ used for key generation: XOR of random numbers generated by a device, received from a manufacturer, rolled by the user, or calculated with time intervals . . . . .	62
3.11	Combinations of attacker properties observing/modifying and passive/active	66

## List of Figures

3.12 Overview of the following cryptographic systems . . . . .	71
3.13 ciphertext can correspond to all possible plaintexts . . . . .	72
3.14 Every ciphertext should correspond to every possible plaintext . . . . .	73
3.15 A simple authentication code . . . . .	77
3.16 Structure of a Turing reduction from factorization $\mathcal{F}$ to extracting roots $\mathcal{W}$	94
3.17 Pseudo random number generator . . . . .	96
3.18 Pseudo-one-time pad . . . . .	97
3.19 $s^2 \pmod n$ -generator used as an asymmetric encryption system . . . . .	102
3.20 Chosen <i>Ciphertext-cleartext attack</i> on the $s^2 \pmod n$ -generator used as an asymmetric encryption system . . . . .	103
3.21 Collision of a permutation pair . . . . .	105
3.22 Mini-GMR for a one-bit-message. $s(0)$ and $s(1)$ are the possible signatures	107
3.23 Principle of GMR's basic signature . . . . .	108
3.24 Basic idea, how to use GMR for signing multiple messages . . . . .	110
3.25 Tree of references . . . . .	110
3.26 Signature for $m$ relative to reference $R_2$ . . . . .	112
3.27 “Top-down” and “bottom-up” calculations . . . . .	113
3.28 The digital signature system GMR . . . . .	114
3.29 Naive and insecure usage of RSA as an asymmetric encryption system . .	117
3.30 Naive and insecure usage of RSA as a digital signature system . . . . .	118
3.31 Secure usage of RSA as an asymmetric encryption system: Redundancy is checked by a collision resistant hash function $h$ and indeterministic encryption . . . . .	123
3.32 Secure usage of RSA as a digital signature system with a collision resistant hash function $h$ . . . . .	124
3.33 Sequence of steps in DES . . . . .	128
3.34 One iteration round of DES . . . . .	129
3.35 Decryption principle of DES . . . . .	130
3.36 Encryption function $f$ of DES . . . . .	131
3.37 Generation of partial keys with DES . . . . .	132
3.38 Complementarity in every iteration round of DES . . . . .	133
3.39 Complementarity in the encryption function $f$ of DES . . . . .	134
3.40 Operation mode “electronic code book” . . . . .	138
3.41 Block patterns with ECB . . . . .	139
3.42 Construction of symmetrical resp. asymmetrical self-synchronizing stream cipher from symmetrical resp. asymmetrical block cipher: block cipher with block chaining . . . . .	140
3.43 Block cipher with block chaining used for authentication: Constructed using a deterministic block cipher . . . . .	142
3.44 Pathological block cipher (only secure with plaintext blocks, which contain a 0 on the right-hand side) . . . . .	143
3.45 Block cipher expanding by one bit . . . . .	143
3.46 Construction of a symmetrical self-synchronizing stream cipher from a deterministic block cipher: ciphertext feedback . . . . .	145

3.47 Cipher feedback for authentication: Construction using a deterministic block cipher . . . . .	146
3.48 Construction of a symmetrical synchronous stream cipher from a deterministic block cipher: output feedback . . . . .	148
3.49 Construction of a symmetrical resp. asymmetrical synchronous stream cipher from a symmetrical resp. asymmetrical block cipher: block cipher with ciphertext and plaintext block chaining . . . . .	150
3.50 aa . . . . .	151
3.51 Properties of the operation modes . . . . .	153
3.52 Construction of a collision resistant hash function from a deterministic block cipher . . . . .	155
3.53 Diffie–Hellman key exchange . . . . .	159
3.54 Usual digital signature system: There is one and only one signature $s(x)$ for one message $x$ and one validation key $t$ (and possibly one signature environment, see GMR). . . . .	160
3.55 For the signee unconditionally secure signature system: For one message $x$ and one validation key $t$ there are multiple signatures (gray area), which are rather sparse within the signature space . . . . .	160
3.56 Proof of falsification in the form of a collision in a signature system unconditional secure for the signee . . . . .	161
3.57 Fail-stop signature system (Glass box with a lock; there is only one key for putting things in. The glass can be broken, but it can not be replaced unnoticed.) . . . . .	162
3.58 Signature system with undeniable signatures (black box with a spyhole; there is only one key for putting things in. The key owner controls access to the spyhole.) . . . . .	163
3.59 Signature system for blind signatures . . . . .	165
4.1 Cryptography vs. steganography . . . . .	170
4.2 Steganographic system of secrecy . . . . .	171
4.3 Steganographic authentication system . . . . .	171
4.4 Steganographic system of secrecy according to the model of embedding . . . . .	172
4.5 Steganographic system of secrecy according to the model of synthesis . . . . .	173
5.1 Observability of the user in an astral integrated broadband mediating network, which mediates all services. . . . .	184
5.2 Link encryption between network connection and switching center . . . . .	190
5.3 Point-to-point encryption between user stations . . . . .	191
5.4 Point-to-point encryption between user stations and link encryption between network connections and switching centers, and between switching centers . . . . .	193
5.5 Evaluation of efficiency and unobservability of the addressing type/address distribution combination . . . . .	199
5.6 Broadcast vs. requesting . . . . .	201

## List of Figures

5.7 Example for a message service with $m = 5$ and $s = 3$ , requested is message 3	202
5.8 Anonymity property of the RING network . . . . .	207
5.9 Reconfiguration of a braided ring . . . . .	211
5.10 superposed sending . . . . .	215
5.11 Illustration of the induction step . . . . .	219
5.12 Pairwise superposed receiving of the user stations $T_1$ and $T_2$ . . . . .	225
5.13 Message format of the algorithm for resolution of collisions with averaging and superposed receiving . . . . .	226
5.14 Detailed example of the algorithm for resolution of collisions with averaging and superposed receiving . . . . .	227
5.15 Booking scheme with generalized superposed sending for the stations $T_i$ .	230
5.16 Practical encoding of information units, keys, and local and global superposition results during transmission: unique binary encoding . . . . .	234
5.17 The DC network's three topologies . . . . .	236
5.18 Superposition topology with minimum delay for gates with two input channels and a driver power of two for the example of binary superposed sending . . . . .	238
5.19 Sender-partitioned DC network with 10 stations, configured so that any desired faulty behavior of any desired but one and only one station can be tolerated without fault diagnosis . . . . .	241
5.20 Farthermost propagation of an error (or a modifying attack) of station 3	242
5.21 Error detection, localization and correction in a DC network . . . . .	245
5.22 Basis functions of a MIX . . . . .	247
5.23 Maximum security: passing through the MIXes simultaneously and, because of that, in the same sequence . . . . .	250
5.24 MIXes hide the connections between incoming and outgoing messages .	253
5.25 Transmission and encryption structure of messages in a MIX network, using a direct scheme of re-encryption for sender anonymity . . . . .	254
5.26 Reducing message length during re-encryption . . . . .	259
5.27 Indirect length preserving recoding scheme . . . . .	260
5.28 Indirect recoding scheme for sender anonymity . . . . .	262
5.29 Indirect recoding scheme for receiver anonymity . . . . .	264
5.30 Indirect recoding scheme for sender and receiver anonymity . . . . .	265
5.31 Indirect length preserving recoding scheme for special symmetrical encryption systems. . . . .	266
5.32 Breaking the direct RSA implementation of MIXes . . . . .	271
5.33 Processing of the channel request message sent by R . . . . .	275
5.34 Processing of channel establishing messages . . . . .	275
5.35 Connecting channels . . . . .	276
5.36 two additional alternative ways over disjoint MIX sequences . . . . .	281
5.37 $MIX_i$ can be substituted alternatively by $MIX_{i'}$ or $MIX_{i*}$ ( $i=1,2,3,4,5$ )	284
5.38 One MIX can be left out each; the coordination protocols are transacted in the picture within groups of minimal circumference . . . . .	286
5.39 Allocation of end-to-end and link encryption in the ISO OSI reference model	294

5.40 Allocation of basic techniques for securing traffic data and data on interest in the ISO OSI reference model . . . . .	295
5.41 Integration of different security measures into one network . . . . .	300
5.42 Interfaces of caller and callee with respective local switching centers (di- vided in two functional parts) and attached MIX cascade . . . . .	302
5.43 Data needed by MIXes $MR_i$ for TS channel establishment . . . . .	303
5.44 Data needed by MIXes $MR_i$ for TR channel establishment . . . . .	303
5.45 Connection establishment in the simplest case. The sending of start mes- sages, i.e., the original start of the connection, is not shown . . . . .	305
5.46 Connection request message of $Q$ for $S$ . . . . .	307
5.47 Parameters of the cryptographic systems used . . . . .	311
5.48 Functions of the interfaces . . . . .	318
5.49 Connection of wireless users to a MIX network . . . . .	323
6.1 Classification of pseudonyms according to personal relation . . . . .	328
6.2 Transfer of a signed message from X to Y, functional view (left), graphical view (right) . . . . .	331
6.3 Authenticated anonymous declarations between business partners who could be made public . . . . .	339
6.4 Fraud protection for completely anonymous business partners through an active trustee, who is able to check the merchandise . . . . .	342
6.5 Fraud protection for completely anonymous business partners through an active trustee, who is not able to check the merchandise . . . . .	343
6.6 Basic scheme of a secure anonymous digital payment system . . . . .	348
8.1 Centralized computation protocol between participants $T_i$ . Every $T_i$ sends an input $I_i$ to a central computer everybody trust. It calculates the func- tion $f_j$ and sends every $T_i$ its output $O_i$ . . . . .	362
8.2 Distributed computation protocol between participants $T_i$ . Every $T_i$ sends in its input $I_i$ and receives its output $O_i$ . . . . .	363
9.1 Secure digital signatures allow autonomous and secure concealment. . . . .	367
9.2 Key-Escrow concealment without permanent monitoring allows exchang- ing keys and therefore a) “normal” concealment, b) asymmetric conceal- ment without Key-Escrow and c) symmetric concealment without Key- Escrow. . . . .	368
9.3 Key-Escrow concealment without retroactive decryption allows direct con- cealment without Key-Escrow. . . . .	368
9.4 Concealment through symmetric authentication by Rivest. . . . .	370
9.5 Concealment through symmetric authentication without participant activity	371
9.6 Where is key backup useful, unnecessary or an additional risk? . . . . .	373

*List of Figures*

# 1 Introduction

In order to outline the area of application, first a short overview of *computer networks* and their services is given.

Afterwards it is discussed more in detail, what is understood by *security*, in particular, which general questions have to be asked: What is to be protected? Against whom is it to be protected? How and whereby can security be achieved?

Finally, at the end of the introduction the meaning of the concept of *security in computer networks* is summarized and the reason why *multilateral security* is practical and necessary explained.

## 1.1 What are computer networks (distributed open systems)?

First the area of application of this script will be sketched briefly:

What are computer networks and what are they supposed to do?

What is to be understood by distributed open systems?

What kinds of systems exist?

Which services are implemented or planned and which are conceivable?

After humans had created and developed computers to a certain perfection, they saw that computers would be more useful if they could communicate with each other. They would not only be able to compensate for the limited computing and observational capabilities of humans, but also for their limited communication ranges and progressive movement possibilities. People started to connect **computers** to **computer networks** (*of first type*) through **communication networks**. Step by step parts of the communication networks, e.g., electromechanical switching equipment, were replaced by processing computers. For that reason, today practically all communication networks are dependent on the functions of computers, leading to the use of *second type* computer networks.

At the next cultural level, humans realized that they had created many different computers which did not give the same meaning to electrical signal levels, bits, characters, data structures (for the meanings were not God-given). This meant that a **spatially distributed** information-technical system (IT system), which uses the possibility of physical communication, existed, but was not **open** in the sense that two randomly connected computers could exchange messages while still preserving the meaning and usefulness of the content (files, documents, pictures). Since then, masses of technicians

## 1 Introduction

have been concerned with the standardization of the following things: first a reference model for open communication, then individual communication protocols, which fit into this reference model, and finally their implementation on most different computer types. Meanwhile, resourceful technicians discovered that computers communicating with each other do not have to be kilometers away from each other to do so. An increase of availability (error tolerance is supported by the limitation of the physical effects of errors) or throughput (parallel work) through uncoupling, spoke increasingly for the implementation of distributed IT systems **in regards to their control and implementation** structure. Consequently, if there is no authority, one generally speaks of a **distributed** system as having a global internal view over the system. Therefore a distributed system cannot be centrally controlled by another authority.

In regards to the history of computer networks:

- 1833 First electromagnetic telegraph [Meye\_87]
- 1858 First cable connection between Europe and North America [Meye\_87]
- 1876 Telephone communications across a 8,5 km long test section [Meye\_87]
- 1881 First telephone local area network [Meye\_87]
- 1900 Start of wireless telegraphy [Meye\_87]
- 1906 Introduction of direct distance dialing to Germany, implemented by two-motion switches, i.e., first fully automatic switching by electro-mechanics [Meye\_87]
- 1928 Telephone service Germany - USA introduced (through radio) [Meye\_87]
- 1949 First functioning von-Neumann computer [BaGo\_84]
- 1956 First transatlantic telephone cable [Meye\_87]
- 1960 First satellite for remote communication [Meye\_87]
- 1967 Start of commission of the Datex-network by German Federal Post Office, i.e., the first communication networks implemented specifically for computer communication (computer network of first type). The transfer takes place digitally and the switching via computer (computer network of second type). [Meye\_81]
- 1977 The German Federal Post Office introduces the use of electronic switching systems (EMS) for telephoning i.e., switching in the telephone network by computer for the first time, (computer network of second type), while continuing analogue transfer [Meye\_87]
- 1981 First personal computer (PC) of the computer family (IBM PC), which also spread widely within the private sector
- 1982 Investments within the transmission system of the telephone network increasingly take place in digital technology [ScSc\_84, ScSc\_86]
- 1985 Investments within the switching systems of the telephone network increasingly take place in computer controlled technology, which now mediates digital signals instead of analogue signals [ScSc\_84, ScSc\_86]
- 1988 Operation of the ISDN (Integrated Services Digital Network) begins
- 1989 First vest-pocket-sized PC: Atari Portfolio; with this computers are, strictly speaking, personal and mobile

## 1.1 What are computer networks (*distributed open systems*)?

- 1993 The cellular radio nets with GSM standard (in Germany: D1, D2) become a mass service with several hundred thousand participants. Only digital signals are transferred and mediated by computers.
- 1994 The World Wide Web (WWW), a distributed hypertext system into which diagrams and pictures can also be merged, opens so many application possibilities for the Internet, even for untrained users, that commercialization, and subsequently the necessity for jurisdiction and sadly censorship as well, is seriously discussed.
- 1998 All switching centers of the German Telekom are digitized [[Tele\\_00](#)], i.e., the process of converting all switching systems to computer controlled technology that began in 1985 is closed.
- 2000 Through so-called WAP-capable cellular radio telephones, mobile access to suitably arranged contents of the WWW is possible. In the spring WAP-capable Handys (140 g) with an asset map (25 DM starting assets) are sold without contract attachment for 199 DM in order to promote the development of a mass-market.

The following services were realized in 1991 by means of the following communications networks:

- Sound broadcasting, television, videotext, etc. (or more precisely: their information) are services, that are **distributed** over the broadcasting station net, but more and more over the developing broadband cable network. The purpose of the broadband cable distributed network is the improvement of the service quality by increase of the available range: more receivable television programs are then called cable television; larger information availability for videotext is then called cable text.
- Telephone, interactive videotext, electronic p.o. box (TELEBOX) for electronic letter and voice mail, telex (TELETEX), remote copying (FAX) and remote working (TEMEX) as well as Telebanking (electronics cash) are services, that are **mediated** over the telephone network and/or the digital text and the data network, cp. Figure 1.1. The telephone network, which is in the participant exchange area still nearly everywhere analogue, but since 1991 in the interconnection area already to a large extent digital, gets digitized step by step. Digitization began in 1988 in some local area networks and since approximately 1993 also in the participant exchange area throughout the whole Federal Republic, afterwards furnishing in each case all these services with higher quality. The analogue telephone network and the participant's access line using digital telephone network are **narrow-band** nets (= 1 Mbit/s), which, in contrast to **wide-band** nets (> 1 Mbit/s), are not able to transfer video in high quality (e.g., television).

At present we use two fundamentally different types of communications networks:

## 1 Introduction

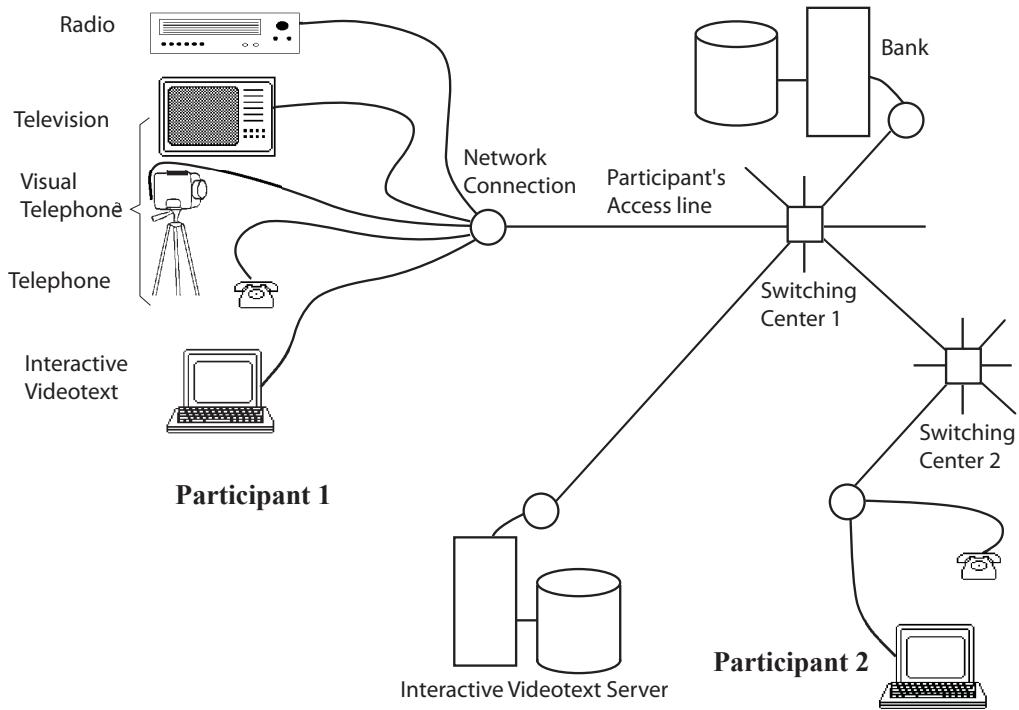


Figure 1.1: A sector of a network

- **Distributed networks**, in which all subsets of the net receive the same data with each participant selecting locally whether or not to receive as well what to receive, and
- **Switching networks**, in which each subset just receives from the net what the participant requested or another participant sent to him.

In nearly all executed and planned public distributed networks, communication takes place only in one direction: from the net to the participant [Krat\_84, DuD\_86]; communication in switching networks generally takes place in both directions.

Because a long-term basis net for all services, a so-called **service-integrating net**, is at least in the participant exchange area cheaper than having several different nets, and since it is possible to mediate all distributed services, it is striven for that all services be mediated in one net [Schö\_84, ScSc\_84, ScS1\_84, Rose\_85, Thom\_87].

So that mediated wide-band services can also be transferred to the participant, new access cables (**fiber-optics**) must be laid. As a result, a broadband cable distributed network gradually becomes redundant. A wide-band service-integrating switching network can offer not only all services which can be offered over a broadband cable distributed network and a narrow-band switching network together, but also additional

## 1.1 What are computer networks (distributed open systems)?

services which require wide-band communication between participants, e. g., for picture telephony.

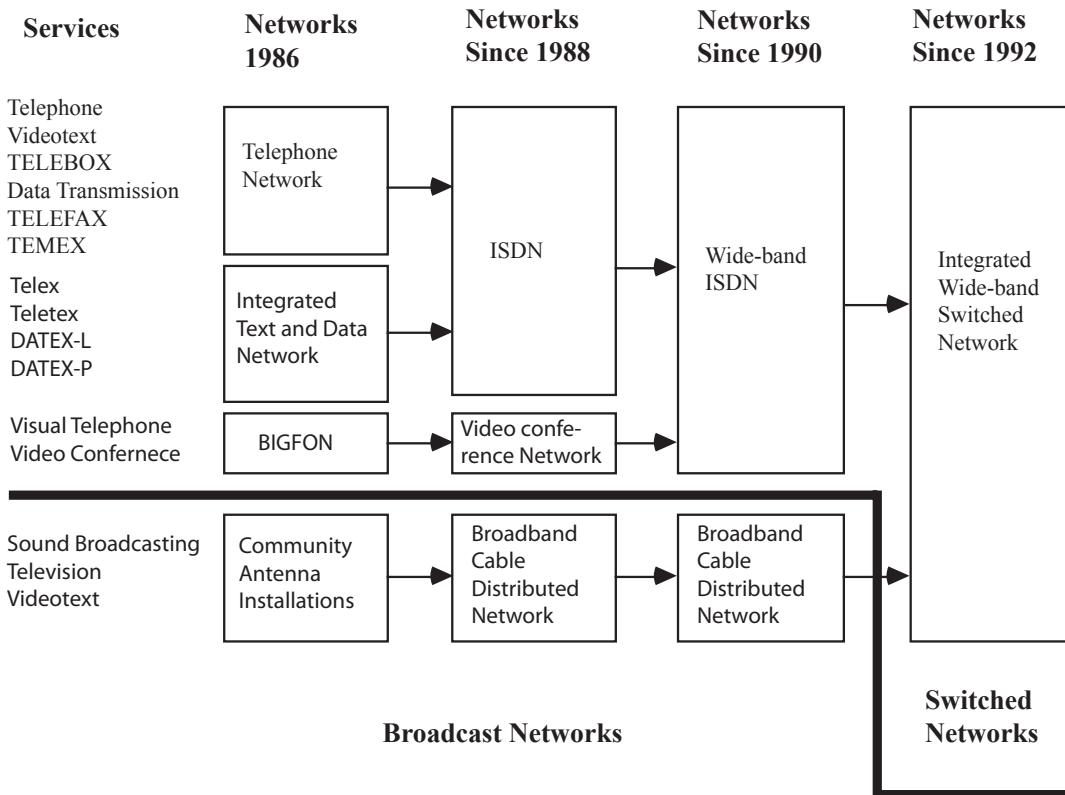


Figure 1.2: Development of the wire-based communication networks of the Deutsche Bundespost Telekom (now: Deutsche Telekom AG), as planned in 1984 and largely executed

Since digital values can not only be transferred in modern technical systems more easily, but also more easily processed, particularly mediated, the service-integrating net will be a **digital** net from subset to subset. I call such a communication network “**service-integrating digital network**”, whereas I use the abbreviation **ISDN** (Integrated Services Digital Network) only for the service-integrating digital network, operated by German Telekom AG, which requires further restrictions. By 1986 digitization had already saved 40% of the costs associated with the remote exchange technique, and was, with falling tendency, already no more expensive than the local exchange technology [Schö\_86]. Both save 65% space, which lowers the above ground construction costs enormously, and the faster connection establishment combined with larger net flexibility saves 15 to 20% of the necessary transmission capacity.

When a broadband ISDN (shortened b-isdn [Steg\_85]) replaces a broadband cable distributed network, whereby on the one hand the maintenance of the broadband cable distributed network can be saved and on the other hand high definition TV, HDTV,

## 1 Introduction

can be offered to a larger extent, it is called an integrated broadband telephone network (IBFN) [ScS1\_84, Thom\_87]. The pilot attempts for the testing of the IBFN are known under the abbreviation BIGFON (wideband integrating fiber-optic telephone network)[Brau\_82, Brau\_83, Brau\_84, Brau\_87]. Since then the name IBFN has gone somewhat out of fashion and the goal of such an extensive monolithic infrastructure, after the abolishment of the net monopoly, has also become less effective. Although there are by now several network carriers that do not promote the goal of IBFN, the technical development continues in this direction. For example, fiber-optics are now positioned in most houses of the new Federal States, but are generally unused, though this last fact can be changed comparatively fast.

Since 1994, the vision of an all services integrating net has been discussed less from the view of the transmission technique, as in the decade before, and more from the view of computer communication: The Internet takes over the hopes, goals, and illusions of IBFN, and is to become *the* medium for mass communication as well. High-quality data compression techniques and adaptive service organization stand as the center of interest instead of gigantic range as with IBFN.

Figure 1.2 depicts the described development in more detail. Not depicted, and first explicitly discussed in §5.7, are radio nets, which have a permanent significance as connection nets for moving participants, as replacement nets in emergencies, and as guarantors of transnational freedom of information.

Beside these digital **public long-distance traffic nets**, there are **local computer networks (LANs)** – without those this script wouldn't have found its journey from the computer to the printer that comfortable. For example, modern hospitals and administrations are, for better or for worse, already completely dependent on their LANs. Someday not only airplanes will be controlled by computer networks (key word: fly by wire; first civilian airplane was the airbus 320, first flight in 1988 or even earlier), but also cars. The latter employ **CAN's (Controller AREA networks<sup>1</sup>)**, which connect computers to instruments, engine and brakes.

Already today many LANs are connected, often via public long-distance traffic nets. This can also be useful for CANs.

In short: Often already today, but in a few years almost comprehensively, technical devices interact through input and output with computer networks. The fact that we interact with a computer network will be progressively less noticeable to us, since the computers take a more behind the scenes role in their outward appearance, though not in regards to their security (problems).

## 1.2 What does security mean?

This second part of the introduction deals with the dangers that generally threaten information-technical systems (IT systems) and in these systems. These include errors

---

<sup>1</sup>Sometimes this name is not used for a class of computer networks, but for a special one: Bosch developed the “CONTROLLER AREA network (CAN)”, which in the meantime has become internationally standardized (ISO 11898).

from humans, losses from integrated circuits, in addition, consciously bad-willing behavior from humans, who naturally can also use the assistance of IT systems for their deeds (computers, communication systems, etc....).

After shortly enumerating *what* is to be protected, it is discussed against *whom* is to be protected and on which *levels* the protective mechanisms must be set.

### 1.2.1 What is to be protected

A typical organization of the *threats* and corresponding *protection goals* for systems of the information technology (IT systems) is as follows [VoKe\_83, ZSI\_89]:

1. *unauthorized gain of information*, i.e., loss of **confidentiality**,
2. *unauthorized modification of information*, i.e., loss of **integrity**, and
3. *unauthorized impairment of functionality*, i.e., loss of **availability**.

Corresponding examples may illustrate this:

1. If case histories (investigations, diagnoses, therapy attempts) are no longer stored on record sheets, but rather in computers, then it is surely undesirable for the computer manufacturer to be able access and read this data during maintenance or repairs.
2. It can be lethal for patients if someone makes unauthorized and *unnoticed* changes to their data, e.g., the instructions for the medicine dosages to be administered.
3. Likewise it can be lethal if a patient's history is stored only in the computer, but it *recognizably* fails just when an inquiry for a therapy measure must take place.

Unfortunately, there is no well-known classification of the threats. In particular the three-divisions above are *not a classification*, even with the attributes I put in italics: If a program from memory that is currently not executing is modified without authorization, this is an unauthorized modification of information. If the unauthorized modified program is executed, this is an unauthorized impairment of functionality.

Because of these demarcation difficulties caused by the interpretation of information, scientists prefer not to differentiate between the 2nd and the 3rd threat.

But from a pragmatic point of view I myself consider this distinction sensible. This is because with the later treated preventive measures it is usually possible to clearly differentiate which of the three protection goals they serve.

Additionally, this distinction has a **correspondence in the correctness term** if one adds the attributes, which are written in italics in the examples, *unnoticed* and/or *recognizably* to the definitions of integrity and/or availability:

## 1 Introduction

Integrity (= no unauthorized and unnoticed change of information) corresponds to the *partial correctness* of an algorithm: If it supplies a result, it is correct.

Integrity and availability together correspond to the *total correctness*<sup>2</sup> of an algorithm: A correct result will be supplied (for sufficient operational funds for the execution of the algorithm presupposed, see below).

For nearly all applications with availability requirements it is characteristic for correct results to be needed not sometime in the future, but at specific times (the next necessary therapy measure is in the above example). Course-sharpened measures ensure safety of integrity for the fact that the correct things happen in time [Hopk\_89]. The proof of total correctness on its own is not sufficient for the general proof of availability. Furthermore, necessary operational funds must at least be analyzed and their sufficient availability proven. Therefore:

Integrity and availability = total correctness and sufficiently operational  
funds available

A few notes for the above classification of threats and protection goals are attached:

The term *unauthorized* in the work [ZSI\_89, ITSEC2d\_91] must be understood in a *very broad* sense, or else the above three threats become incomplete: For example, the loss of a transistor can modify information or impair functionality, and even cause loss of privacy. But it is unclear if this represents an *unauthorized* event with *unauthorized* consequences in the general linguistic usage, because powers will in general only be transferred in instances with responsibility, so in other words humans. Instances without responsibility cannot act unauthorized, and can therefore not cause something unauthorized.

The *impairment of functionality* must refer to the *qualified users*. Otherwise, if an IT system is used intensively by unauthorized users its functionality cannot be impaired, since it works. But depending upon the extent of utilization by the unauthorized users of the IT system, the functionality of the IT system would be detracted from the entitled users [Stel\_90].

In [Stel\_90] it is criticized that in [ZSI\_89] the protection goal, *(legal) commitment*, is not listed. A similar protection goal, called *accountability*, is later introduced to the Canadian safety evaluation criteria [CTCPEC\_92] as the additional fourth. Some protection goals can be more concisely described through this, cp. §1.3. Since one can also seize threats and/or protection goals under integrity and should manage with as few threat types as possible, I will continue with the usual three-division classification.

Positively and as briefly as possible formulated, the three protection goals are therefore:

**confidentiality** = information is only known to entitled users.

**integrity** = all information is correct, complete, and up-to-date, or  
is recognizably not one of these.

**availability** = all information is accessible where and when it is needed by  
entitled users.

Here, in each case, “information” is meant in a broad and comprehensive sense, so that data and programs, as well as hardware structures<sup>3</sup> are encompassed.

So that what is said about “entitled users” makes sense, this term must at least be clarified outside of the IT system being considered: Who is entitled for what and in which situations?

Finally, it is noted that in each case, “correct, complete and up-to-date” can only refer to the inside of a considered system, since accuracy of the system’s view on its environment is not definable within the considered system.

### 1.2.2 Against whom is to be protect

On the one hand the laws of nature take effect in and on every technical system, and if the system is not protected, the forces of nature can also have a big influence. The former means that components age and eventually won’t work as planned. The latter means that precautions against electrical surges (lightning strokes, EMP), voltage drops, flooding (heavy rainfall, broken water pipe), immediate temperature changes etc. need to be employed. Both topics are covered by the special field of fault-tolerance[[Echt\\_90](#)].

On the other hand, people may have an undesirable impact because of incompetence, negligence, or unconsciously unauthorized actions. In accordance with their roles within the IT systems, it is useful to separate them into(cp. Figure [1.3](#))

- externals
- users of the system
- operators of the system
- servicemen
- producers of the systems
- designers of the systems
- producers of design and production items
- designer of design and production items
- producers of design and production items for the producers of design- and production items ...

It is pointed out that all of these producers and designers have users, operators, and maintenance services of their own. If necessary, you have to protect against them as well.

---

<sup>3</sup>Since procedures – in other words: functionality – can be described not only by programs but also by hardware structures, this is necessary.

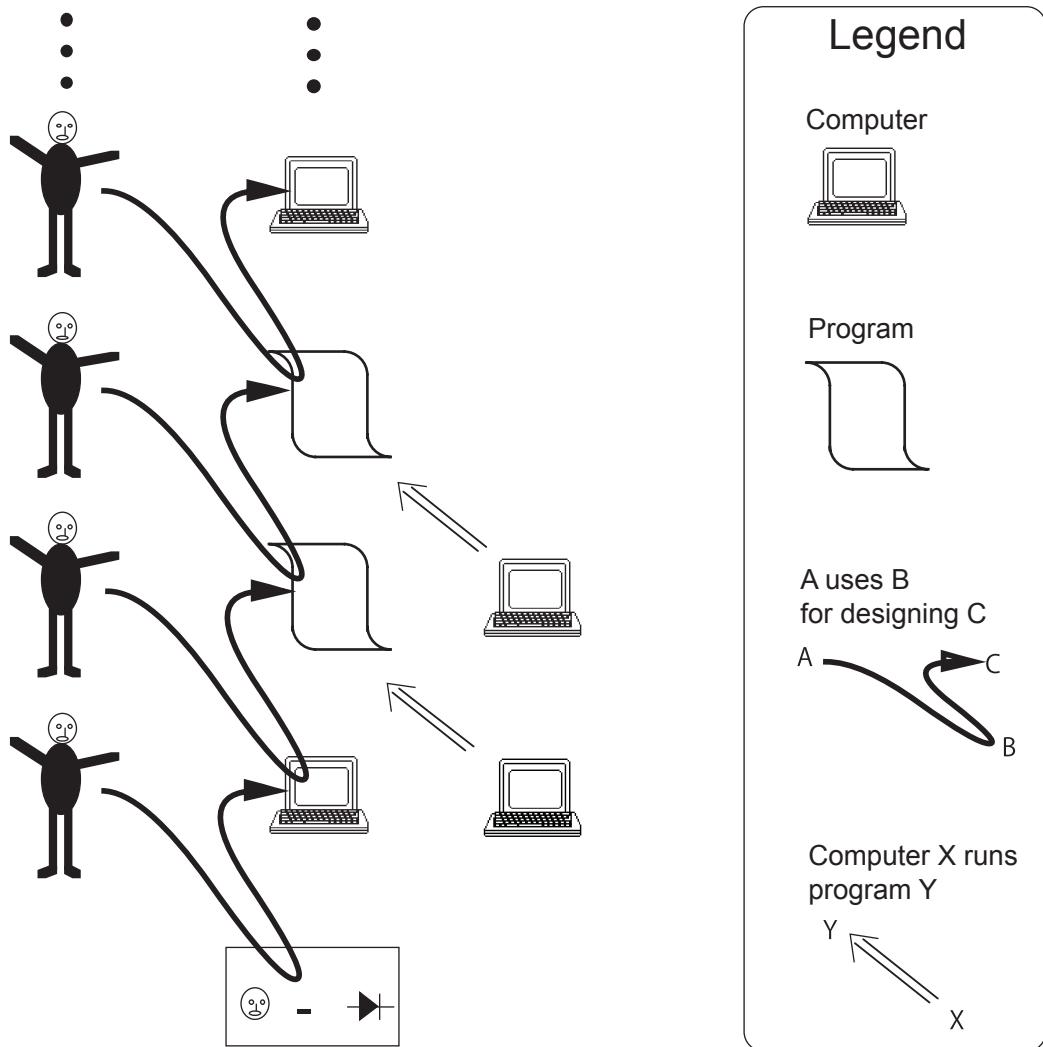


Figure 1.3: People use software programs to develop other software programs and computers

While it is common to distrust externals and users of the IT system, and therefore to protect against their unconscious mistakes and conscious attacks, all the other components (especially the influence of further IT systems) are omitted. This is a serious security breach, since not just unconscious mistakes, but also conscious attacks, can propagate along the design and production line. In Figure 1.4 the human at the top certainly should not be blamed if the computer that was partly produced with a computer is faulty. It could also possibly be the fault of the person at the very bottom, the computer on the bottom right, or anything else at a lower level.

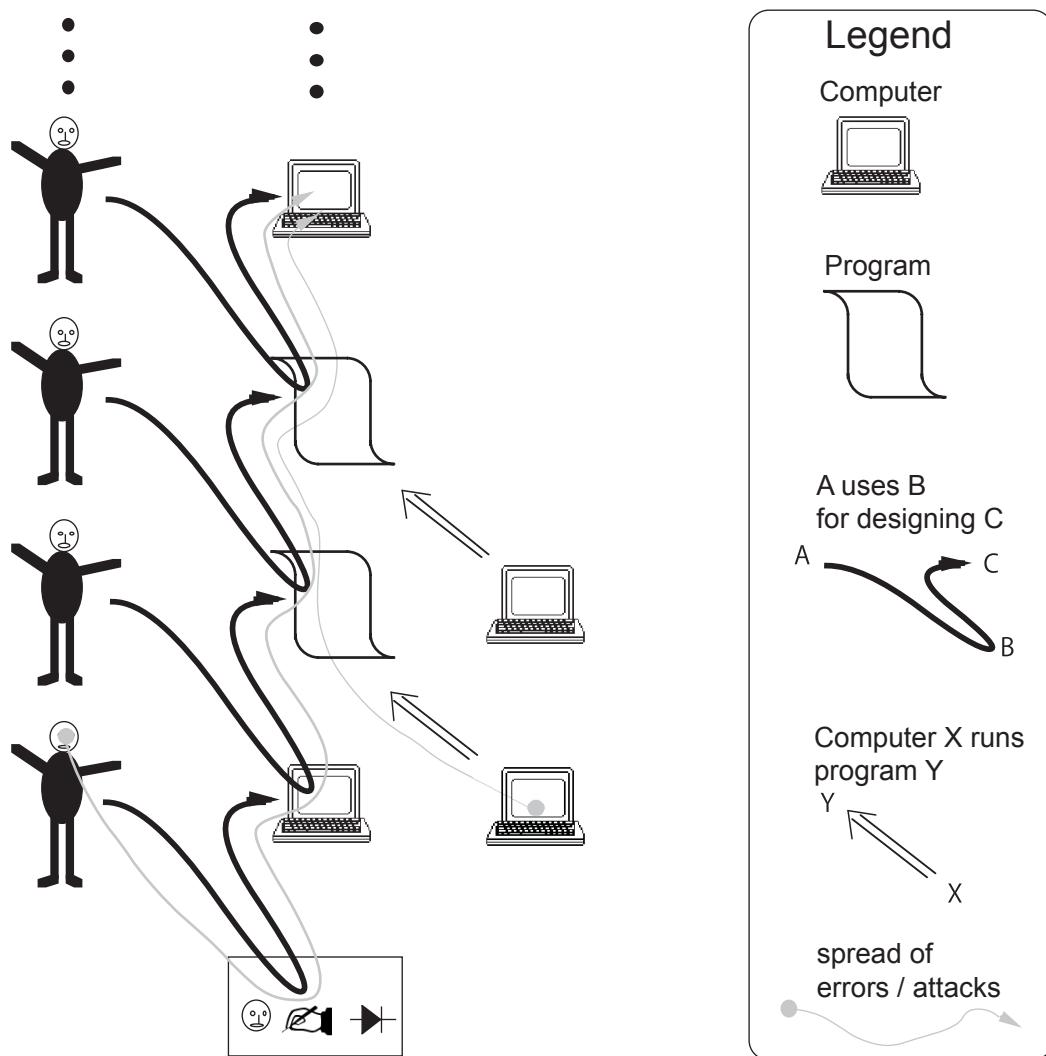


Figure 1.4: Transitive distribution of errors and attacks

The especially high complexity of today's IT systems prevents sufficient monitoring of the systems themselves. That's why their producers and designers, be it the people or the machines (which were at some point man-made) used, have to be considered as potential **attackers**. For example they could hide **Trojan Horses** within parts of the system they designed or produced:

A system part is a **Trojan Horse**, if, using the given data and granted permissions,

## 1 Introduction

it does more than expected<sup>4</sup> or does what is expected *wrong* or *not at all*<sup>5</sup>, cp. fig. 1.5. For instance a text editor could not only save the input to the user's file but also send a copy to the author of the program. For this purpose the Trojan Horse needs a *covert channel*<sup>6</sup>[Cove\_90, Denn\_82, Lamp\_73][Pfit\_89, pages 7,18]. If the editor in our example could not use a second file as covert channel (on many computers it can), it could modulate operating system resources (memory consumption, CPU time) and that could be perceived by other programs.

Unfortunately, there is no practical method for revealing all Trojan Horses or even ruling out their possible existence in an examined system. All in all, you have to rely on watching with the naked eye (with the complexity of today's computer you could even watch almost as well with closed eyes).

The bandwidth, even of known covert channels, cannot be arbitrarily reduced without enormous additional costs. The highest demands in IT security criteria, without limiting the amount of channels, allow a bandwidth of 1 bit/s per hidden channel[ZSI\_89]. Say the attacker can transmit "only" 1 bit/s of a Trojan Horse: Within one year he would receive 4 MBytes of data.

Fortunately, by using diversity, meaning that the design is conducted several times by independent designers with independent aids, you can reduce the bandwidth to 0 bit/s [Cles\_88]. But the expenditures for that would be considerable.

---

<sup>4</sup>In literature, a Trojan Horse is often defined as a “program executing *undocumented additional functions* along with its alleged own function.” With the unnecessary limitation to *programs* and *additional* functions the definition seems to be a bit unsuitable: For every function exists at least one documentation containing *every* detail: the program code itself [Dier3\_91].

<sup>5</sup>At the first glance this definition may look too common, because it demands no *malicious intent* from the author, while according to Homer, the Greeks took use of the wooden horse in front of Troy with such an intent. For the here introduced Trojan Horses, extended insight is given with these two notes:

1. The criteria, whether or not there is *malicious intent* in functions of system parts, is not to be decided - a mathematical definition with respect to those attributes would be quite hard to handle (Moral or legal abjections which would be relevant for the prosecution remain untouched here).
2. For effects of unexpected functionality it does not matter whether or not it was created with malicious intent. Only if potential offenders know of it and can exploit it, does it become relevant.

<sup>6</sup>In the literature both *hidden channel* and *covert channel* are used with the same meaning.

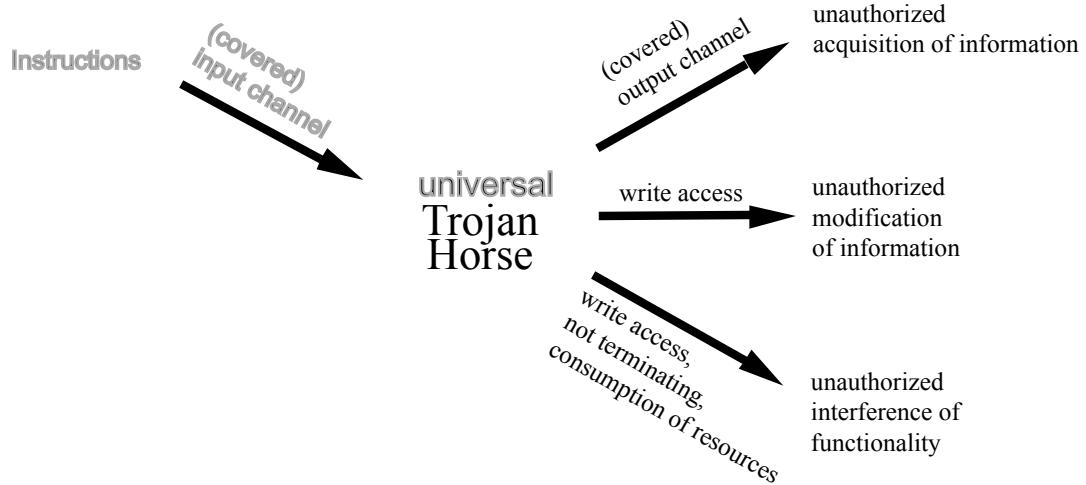


Figure 1.5: Universal Trojan Horse

This very broad definition of Trojan Horse allows many different kinds of interpretations, partly with material names like logical bombs, etc. However, for the principle inspection, only the following attributes “universal” and/or “transitive” are important.

The damage functions of a Trojan Horse are usually fixed during the design stage. With this the Trojan Horse would be a quite inflexible and unaligned attacker, and if discovered, the author’s intent to damage would be detectable. But it is also possible to create an acting frame during the design stage, which would only come to life when the system is in operation. The input for the Trojan Horse would be stashed in the code of a harmless looking part of the system (fig. 1.5). The encryption of the input is chosen so redundantly that the probability that an external person giving input into the infected system part would give input to the Trojan Horse too is minimal. In the most general case you can freely program the Trojan Horse during operation. Dorothy Denning named these **universal**[Denn\_85] Trojan Horses. The original purpose of universal Trojan Horses is of course only possible for people who have access during operation. Particularly in *open* systems, this is possible to nearly everybody. With the help of universal Trojan Horses, very flexible and pointed attacks can be made. The user of a Trojan Horses does not have to wait for years to receive the datagram he has been interested in as of late, but can have it sent immediately.

It is generally thought that the offenders have to directly place Trojan Horses where they can take their subversive effect. But this thought is wrong: If during the design of the system other aids are used (i.e., compilers), both the designer and the production aid can infiltrate a Trojan Horse into the design. Since the production aids are made with other production aids, the Trojan Horse can spread itself recursively within the transitive closure where the guest system was directly or indirectly involved [Thom\_84, Pfit\_89]. That is why I speak about **transitive** Trojan Horses.

The group whose members must be considered as potential attackers is continuously growing because of the transitivity of Trojan Horses, which equals the former recursive characterizing of the set of designers and producers. This growth is unfortunate for two different reasons:

- First of all, the probability that all of them are not attackers decreases just because of the increase in number of people. Additionally, as the size and heterogeneity of the group grows, the sense of partnership and responsibility-for-each-other decreases.
- Secondly, executing controls on these people is increasingly more expensive, unacceptable, (catchword Big Brother) and limited by the cooperation of the nations (which country does not deny the surveillance of its people by others; on the other side: which country denies the import of software?).

### 1.2.3 How and through what can security be achieved?

First of all, we have to agree on what “unauthorized” means. That could be achieved through international conventions, a constitution, laws, cooperation agreements, or by the ethical agreement of professional groups of computer scientists. “Unauthorized” actions are penalized.

Organizational structures should be chosen so as to better hamper or prevent unauthorized actions.

Are the laws, prohibitions, and organizational measures sufficient?

A prohibition is only effective if its adherence can be *verified* and is backed by legal prosecution, and if the original state can be *recovered*. Both conditions don't apply to IT systems.

Generally ”data theft”, especially wiretapping and copying data out of the computers, is almost impossible to detect because the original data is not changed. As mentioned above, it is also almost impossible to detect the installation of Trojan Horses, not to mention the further processing of data acquired legally or illegally.

The recovery of the original state would mean deleting the accrued data. But it is never certain if even more copies exist. Moreover, data can be held inside the human's memory, where deletion had better not be attempted.

Subsequently, more effective measures are needed, as prevention and *technical* precautions are the only well-known ones.

### 1.2.4 Preview of precautions

Fig. 1.6 gives a preview of measures which will be explained in the following chapters. The column “preventing undesirables” (reads: Preventing undesired actions of the system) corresponds essentially to the goal of confidentiality, and the column “rendering

desirables" (reads: Desired actions of the system shall take place) to the goal of integrity and availability.<sup>7</sup>

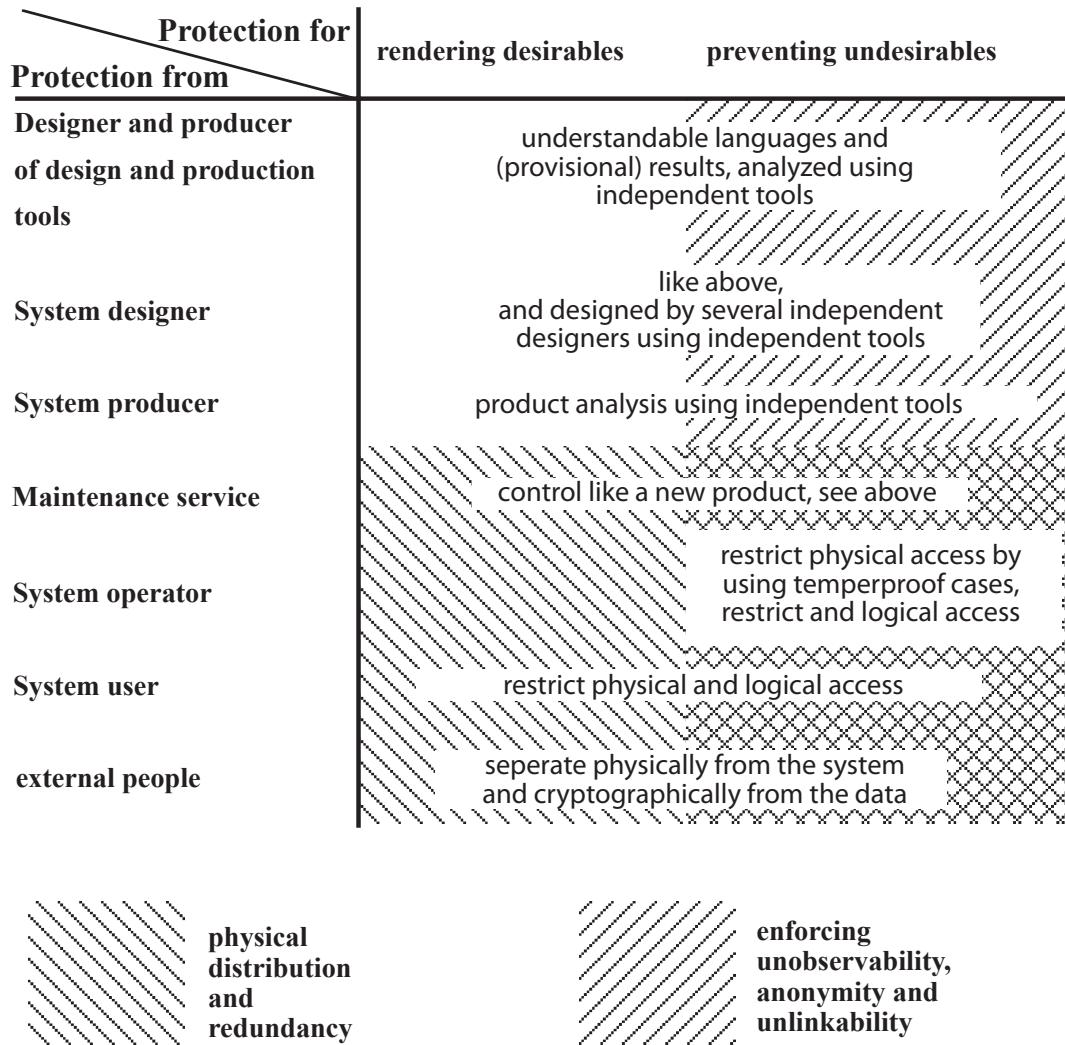


Figure 1.6: Which security measures are safe from which attackers?

<sup>7</sup>unobservability, anonymity, and untraceability may contribute in the following subtle way: If the attacker is not willing to affect all (or at least very many) participants - because he does not want to attract attention or, for example, does not want to hurt the weak members of a society because he is a politically motivated terrorist - unobservability, anonymity, and untraceability prevent a *selective* impairment on the availability of certain groups of participants or individual participants. In such situations you do not need to reckon on damage from offenders with these motivations.

### 1.2.5 Attacker model

Protection from an omnipotent attacker is of course impossible: an omnipotent attacker could

1. trace all data he is interested to the site of their creation (or if necessary collect data from different points and attach them to the ones he is interested in).
2. alter data without drawing notice, because if the user knew what value they must have, he could simply forgo them.
3. affect the whole IT system's functionality through physical demolition.

That is why all of the following measures are only approximations of perfect protection for the participants from all possible attackers. The approximation is determined by the specification of the **maximal considered strength of an attacker** formed into a so-called **attacker model**.

To characterize single mechanisms of protection, the maximal strength of the attacker that is being protecting against, should be accurately described. The attack model encompasses, apart from the possible roles of the attacker (external, user, operator, servicemen, producer, designer ...), the following attributes:

To describe how much the attacker spread, you should quantify his possible roles: Which physical and cryptographic barriers can he surpass? Which subsystem can he use and in respect can he take control of it? Which subsystem could he use as an operator or servicemen, and how? As producer where could he infiltrate Trojan Horses? etc.

When all subsystems are equal in regards to the role of the attacker, it is sufficient to assign a non negative natural number to the system indicating how many subsystems can be controlled by the attacker.

If wires and stations (i.e., relay central, accesspoint, enduser device) are distinguished as the smallest subsystems in computer networks, then it must be specified how many and which stations and/or wires can be observed and/or actively altered by the attacker.

If the attacker acts externally (more precisely: within the whole IT system but outside of his system part<sup>8</sup>; cp. fig. 1.7) only as he is allowed to, does it mean that he is only performing an **observation** attack? He could do this, for example, by listening to wire or radio connections, and evaluating data in a different form or saving it longer than allowed. Or is the attacker risking more by doing external things *prohibited* to him, in other words, by conducting a **modification**<sup>9</sup> attack?

In general, observation attacks can not be recognized - but their success, as we will see later, can in principal, be completely thwarted (in reference to the act of collecting and

---

<sup>8</sup>The attacker's **System parts** are defined as being regions controlled exclusively so that others can't easily look inside. The **zone of control** and therefore the **spreading zone** of the attacker is usually greater, as he usually encompasses big parts of the IT system itself.

<sup>9</sup>I'm not fully happy with the terms **observation** vs. **modification** attacker but I don't know of any better terms. When using **protocol devoted** vs. **protocol violating**, you must assume that everything specified by the protocol is allowed, but this does not hold true for today's protocols.

processing data the attacker would not regularly have access to). Modification attacks can theoretically be spotted, but their success can not completely be prevented. Outside the IT system observing attacks could be possibly detected and outcomes of a modifying attack should be limited.

For further explanation, it might be useful to assign the attributes observing/modifying not to the attacker as a whole, but to his different roles within the IT system.

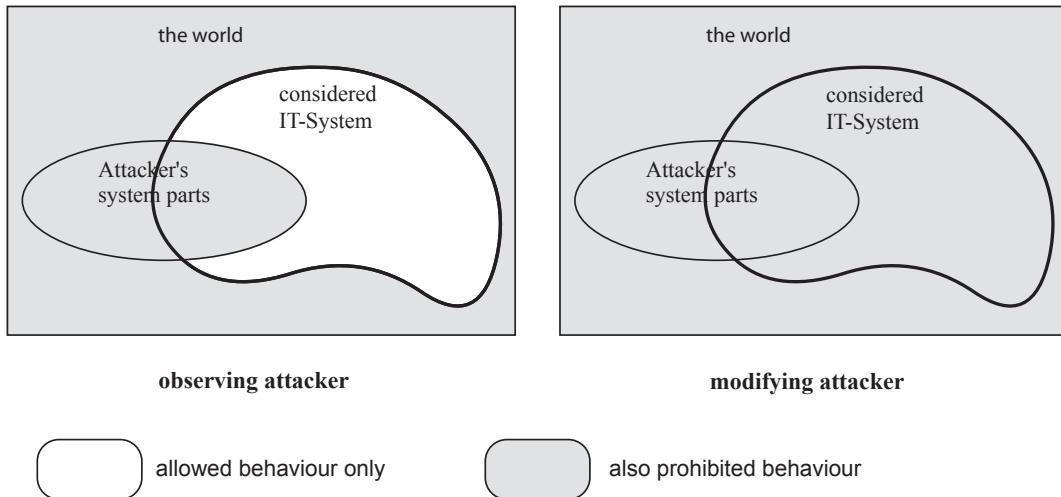


Figure 1.7: Observing vs. modifying attacker

In §1.2.2 we distinguished between stupid, or accidental, mistakes and intelligent, targeted attacks. For the latter it may be useful to distinguish more precisely: How much computing power does the attacker have?

The safe side concedes that the attacker has endless computing power, in other words a complexity-theoretically unrestrained (or: **information-theoretical**) attacker. Therefore, proofs of security have been made in Claude Shannon's computationally unrestricted world model, which is covered more precisely in §3.

It is risky to assume that the attacker only has limited computing power (calculation ability) at hand, so that, for example, he is not able to factor out large primes within a reasonable amount of time, cp. §3. This is a complexity-theoretically limited attacker. Here you can be wrong with respect to his hardware resources and algorithm knowledge. The main risk, as we will see in §3, is in underestimating his knowledge of algorithms.

To substantiate the computationally restricted attacks, the attackers can be distinguished by how much **money** and **time** they have and how much they are willing to spend. As a non-technical measure, the money completely outside the IT system could be considered: Bribing people is the most useful means in many situations, i.e., to make designers install Trojan Horses, operators manipulate the IT system, or users cooperate with the right permissions.

## 1 Introduction

Coming up with a realistic attacker model is an art comparable to estimating mathematical proofs: If the estimation is too rough, the proof fails. If the estimation is too delicate, the proof becomes unnecessarily difficult, fault-prone, and not very convincing. Coming back to attacker models, this means that often all possible points and ways of attack are combined into *one* attacker, though hopefully not all instances of controlling points of attack will conspire together.

A **realistic attack model** should not only cover all attackers known at the moment, but also *all attackers that can be expected during the life of the system*. It should be as simple as possible, so that the remaining risks of an unexpectedly strong attacker are easy to understand. Despite this approximation of the attacker, a system secure against him should, with some effort, be built and operated. A proper proof of all the target's security for this system should also succeed.

The life span of a system has influence not only on the electrical measurements and technical devices of a future attacker but also on his motivation, his will to spend money or time, and the risk assumed. All of this can be increased by political developments, the increase (due to the political or economic reasons) of the processed data's value in the system, or with a growth of dependence on the system from individuals, organizations, or nations. When in doubt, do not try to make a better risk analysis, as that is not possible due to the unpredictable nature of future development, but instead increase the efforts in building a system tough enough to resist stronger attackers.

## 1.3 What does security in computer networks mean?

Some protection goals will here be mentioned, though this list is not exhaustive and there is room for error. These goals are covered more thoroughly in §5 after the necessary formal and cryptographical terms are introduced in §3 .

### Protection goal confidentiality

- The content of messages should be kept private from all instances, except for communication partners.
- The senders and/or recipients of messages should stay anonymous to each other and should not be observable by third parties (incl. the operator of the net).

### Protection goal integrity

- Falsifications of the message content (incl. the sender's address) should be recognized.
- The recipient should be able to prove to third parties that instance  $x$  has sent the message to instance  $y$  (and where applicable, the time when it happened).

- The sender should be able to prove that the message was sent with the correct content and if possible also that it was received (and where applicable, the time when it happened).
- Nobody can withhold compensation for the net operator for services rendered. Conversely, the operator can only demand compensations for correctly rendered services.

Granted, putting the three last protection goals under integrity is forcing the issue a bit, but alternatively, introducing further protection goals could easily drive the matter out of control, cp. end of §1.2.1. The last protection goal, however, could also be put under availability.

### Protection goal availability

- The computer network allows communication between all partners who wish to communicate (and who are allowed to do so).

These protection goals are multilateral and not equally strong for all participants. Depending on the communication subject and circumstances, single protection goals won't have the same meaning for the same user at all times.

## 1.4 Why multilateral security

To generalize:

We do wish that everybody is protected against anybody else, because everybody can, through Trojan Horses, handling errors, their own intentions, or malicious intent, threaten the protection interest of others with devices, and should therefore be considered as potential attackers. If everybody is protected as well as possible against anybody else, we call it **multilateral security**[RaPM\_96].

Multilateral security means the inclusion of *all* participants's protection interests as well as dealing with resulting conflicts, such as the development of a communication connection.

The implementation of multilateral security does not inevitably lead to the fulfilling of all participant's interests. Because protection goals are formulated explicitly, it perhaps reveals incompatible interests of which nobody was previously aware. However, implementation of multilateral security does guarantee that partners will interact with each other within a multilateral communication relationship with a clear ratio of forces.

Technically speaking, it depends on giving the user the possibility to formulate, negotiate, and enforce his protection goals. So multilateral security requires developing accessible methods which give the user the chance to protect himself at least within information technical systems.

Of course the practical employment of those methods will depend on the economical, social, and legal conditions of the situation. Security is an interdisciplinary issue and therefore requires a close dialog with other scientific disciplines.

## **Reading recommendations**

Computer networks: [Tane\_96]

Security: [MüPf\_97], [Denn\_82], [ZSI\_89], [ZSI\_90], [ITSEC2\_91]

Security in computer networks: [VoKe\_83]

Legal basics: [BFD1\_95], [BFD5\_98], [Tele\_98]

Surveillance technology:

<http://www.europarl.eu.int/dg4/stoa/de/publi/166499/execsum.htm>

## 2 Security in Single Computers And Its Limits

After discussing physical security, “logical” protection for isolated computer systems from unauthorized access and computer viruses will now be outlined.

### 2.1 Physical security assumptions

All technical security measures need to be physically embedded in a section of the system where the attacker who is being considered does not have any physical access. In general, we deny reading (Security of Confidentiality) and modifying (Security of Integrity and Availability). Depending on the protection concept and regarding to the single security measures, this encapsulated part of the system can vary in size, graphically from a “data processing center X” to a “chipcard Y”. This part of the system should of course be a different one for the various attackers. We differentiate between several sizes in regards to their dimension and complexity. In both cases, humans not only act as possible attackers but also (in hopefully useful organization structures) as desired “defenders”.

Achieving security would be considered very favorable not only for the IT system as a whole, but especially for each independent user. We can embed the security for user B inside a part of the system that user B exclusively protects, e. g. a chipcard, which he will carry around all day, even while showering, and then certainly put under his pillow at night.

What can we expect from physical security measures at best? Which security measures can we use to at least approximately achieve this? Are chipcards really physically safe? What are useful security assumptions? We will discuss these questions one by one.

#### 2.1.1 What can we expect at best?

The availability of a locally concentrated part of the system cannot be guaranteed if we assume that we have to protect it against thoroughly **conceivable** mighty attackers, such as terrorists or owners of sufficient conventional explosives or even atomic or hydrogen bombs. If we consider the enormous physical costs of security for thick reinforced concrete, emergency power supply, etc., we will be more efficient through the use of distributed systems instead, in the hopes that the attacker is not capable of severe physical attacks on many different places.

Through the distribution of the IT system, the problems of confidentiality and integrity increase rapidly. In regards to these aims of protection, you can expect more from

physical measures, namely protection against all at time conceivable attackers: Certainly you can try, e. g. to construct machine cases in a way, that an attacker who is restricted to the currently known laws of nature can not take unauthorized possession of the information stored inside these cases, nor modify without authorization. If you can achieve this sufficiently, then nothing impedes a physical distribution.

### 2.1.2 Design of security measures

Physical security measures consist of a combination of the following five **basic functions**:

Observation attacks, such as analyzing electromagnetic radiation, must be prevented by **shielding**. (Techniques and norms are known under the abbreviation TEMPTEST [Horg\_85]. Often we forget that even inputs, such as usage of energy, must be independent from the secrets we want to protect [Koch\_98].)

Modification encroachments must be **detected** and **estimated** in relation to their admissibility.

Modification attacks, meaning unauthorized modification encroachments that were detected, must be **delayed**. If necessary, the information must be **deleted** before the end of the delay period.

Depending on dimensions and complexity, it makes sense to design the physical part of the system we want to protect, as **circular layers** [Chau\_84]. We can include several basic functions inside these layers, as we can see in Figure 2.1. Even multiple renderings of the same functions can be implemented, i. e. the detection of different physical dimensions. As an alternative, or a supplement, these functions can occur again in more interior layers. Figure 2.1 shows three examples in which each basic function occurs two times at most.

Even if it is possible to increase the physical security by implementing an appropriate combination of physical security measures, this has two principle disadvantages.

- It is very difficult to validate these measures objectively. There are two reasons:

First, the security of some physical security measures is based on the secrecy of their exact mode of action - it is at least kept secret. This can be because of military reasons or to protect know-how with commercial interest.<sup>1</sup>

Second, it is very important to know the actual state of development of measuring techniques because it is not cost efficient to use physical measures stronger than

---

<sup>1</sup>One of very few exceptions [Wein\_87] is described here shortly: The part of the system which is meant to be secured from physical attacks will be densely wrapped round with thin isolated wire and transfused with an opaque substantiating liquid. The resistance of the wire is measured continuously and the data meant be treated confidential, and will become actively deleted if a suspicious change is detected. If the security layer is physically removed or drilled out, the wire will become interrupted. If someone tries to dismantle the secured item chemically, short circuits will emerge between the wires because the isolation of the chosen wires are more chemically soluble than the sustaining liquid. In both cases the resistance will be changed clearly.

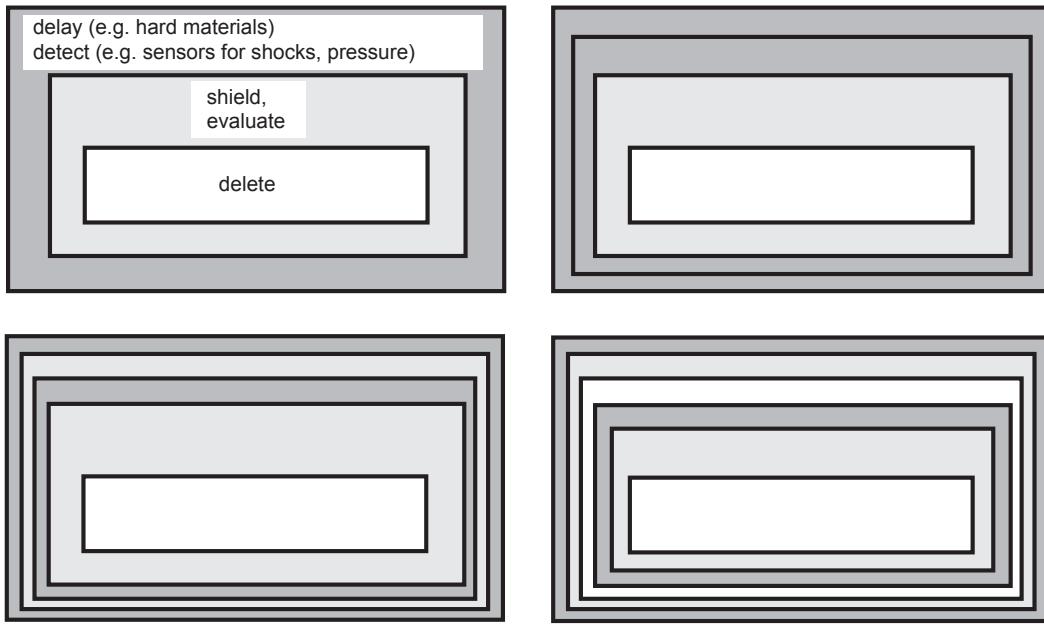


Figure 2.1: The five basic functions arranged in circular layers (upper left figure) and three possibilities of iterated layers.

needed (as it is for cryptography measures, cf. §3). Since in the military field the quality of measuring techniques is kept secret, knowing the actual state of development is rather problematic. For these reasons, the subjective credibility is often fairly low.

- In comparison to other solutions, physical security measures are more expensive and they don't become cheaper as quickly as "normal" digital components. Maintenance is particularly complicated.

### 2.1.3 A negative example: chip cards

Chip cards are physically much more secure than standard magnetic strip cards, but in my opinion they are a bad example of true security:

- It is difficult to shield them since we want them to be thin and elastic. Furthermore, it is possible to immediately measure their energy consumption because that energy is provided externally during the service. In common chip cards the amount of energy consumed depends on the processed instructions and even on the processed information and keys [Koch\_98]. Moreover, nowadays chip cards receive their clock pulse from outside, which made sensational investigations possible [AnKu\_96].

## 2 Security in Single Computers

This could violate confidentiality (at least between the card holder and the devices where the card is put in).

- Normal chip cards are not able to detect encroachments due to the lack of a personal power supply, e. g. no battery.

This could violate confidentiality and, if applicable, the integrity of data processed inside the chip card.

- While they are supplied with energy by the reading device, normal chip cards do not delete their sensible data, not even after a detection of an unauthorized encroachment.

This could violate confidentiality of data processed inside the chip card.

### 2.1.4 Useful physical security measures

According to the reasons named in §2.1.2, physical security measures should be handled sparingly when possible:

Widespread conformity between organizational and information technological structures should be achieved. The ideal case would be if everybody processed and secured their own data, which others are then only able to gain access according to that owner's permission.

All of this means that, if possible, we want to avoid situations where the devices are secure against their users.

Despite all this technology, it is cautious and probably necessary to presume that only in space can data exist where nobody has physical access to it. Fortunately, as it is outlined below, it is possible to achieve security (Confidentiality, Integrity, Availability; according to the specific applications e. g., security by law, protection of personal correlated data from unauthorized access, etc.) even under these defensive presumptions.

## 2.2 Protection of isolated computers from unauthorized access and computer viruses

“**Logical**” protection assumes that physical and organizational protection is given, as was discussed in the previous subchapter.

First we will describe how it is possible to identify humans and IT systems<sup>2</sup>. This is the reasoning for blocking services of the IT system from unauthorized users (admission

---

<sup>2</sup>Some authors distinguish between **identification** (for them: assertion of an identity) and **authentication** (verifying an identity). We understand identification at a global level, meaning assertion and verification of an identity. We handle it like that, because in our case both always appear together. We want to apply the term authentication on messages, e. g. “Does a specific message really come from this alleged originator?”

## 2.2 Protection of isolated computers from unauthorized access and computer viruses

control) and granting access to authorized users, though only to resources which are thought to be used by them (access control). At the end of this chapter we will see how to restrict the threat of “computer viruses” based in “transitive Trojan Horses”.

### 2.2.1 Identification

IT systems are able to recognize a human by what he is<sup>3</sup>, has got, or knows, cp. Figure 2.2. The same terms can be applied to recognition of humans by humans, whereas hand geometry, retina patterns, magnet strip cards, chip cards, and calculators do not play an important role.

Of course, combinations are possible and useful, e. g. a passport combines “What you are” (appearance through a picture and partially unconscious actions through a signature) with “What you have” - the passport itself.

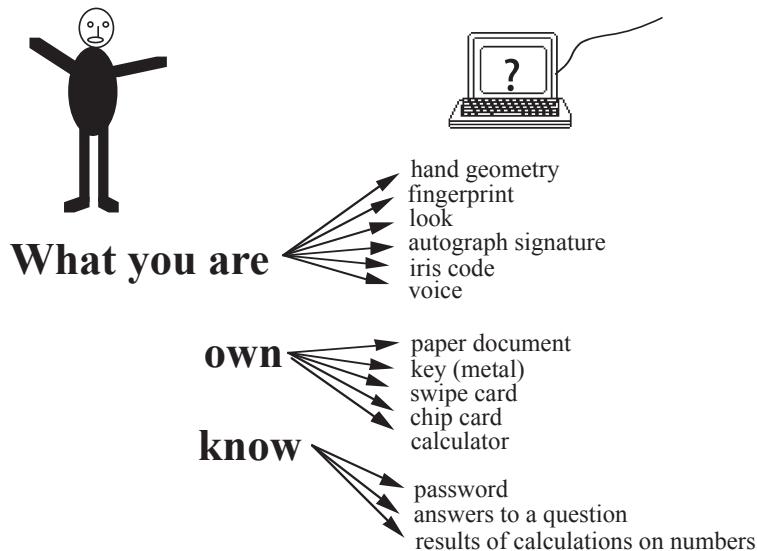


Figure 2.2: Identification of humans by IT systems

Figure 2.2 does not give a complete list:

On the one hand, the given examples can be split up further. For signatures we could “check the results of the signing process” by comparing two static line patterns and “analyze the dynamics of the signing process” by comparing pressure and acceleration values with signature samples.

On the other hand, there exist more examples such as the analysis of typing behavior for keyboard inputs or the style of writing for pen inputs. Both represent the group “What you are”.

<sup>3</sup>The technique to determine what you are is called **biometric technique**. An overview of capacity and costs of biometrics is given in [DaPr\_89], [DuD3\_99], [Eber\_98], [FMSS\_96], [Mih3\_94].

## 2 Security in Single Computers

For each measurement you have to decide if it is performed:

- just at the beginning
- additionally at fixed time intervals or
- permanently

Among other things, for a “metal key” and the analysis of the typing behavior, a permanent controlling action is suitable (the key remains inside the lock and the analyzing process for keyboard input can take place in the background), whereas you cannot demand from anybody to put his hand on a scanner to permanently render a geometric image of the hand.

Humans can recognize an IT system through what it **is**, **knows**, and **where** it is located, cp. Figure 2.3. The example will show that it is not enough if the IT system is capable of recognizing the user, but the user must also be capable of identifying the IT system. The location by itself may be insufficient, as you will see in the following example:

*example:* From Friday night to Saturday morning, a dummy cash machine was installed in front of a real one, covering the original. The dummy one asks for the PINs (personal identification numbers) of the inserted EC cash cards and stores it. The system replies through the display: “Your faulty EC cash card has been impounded, please ask your credit institute for help on Monday!”. In between, money can be withdrawn from a real cash machine using the impounded cash cards and the stored PINs.

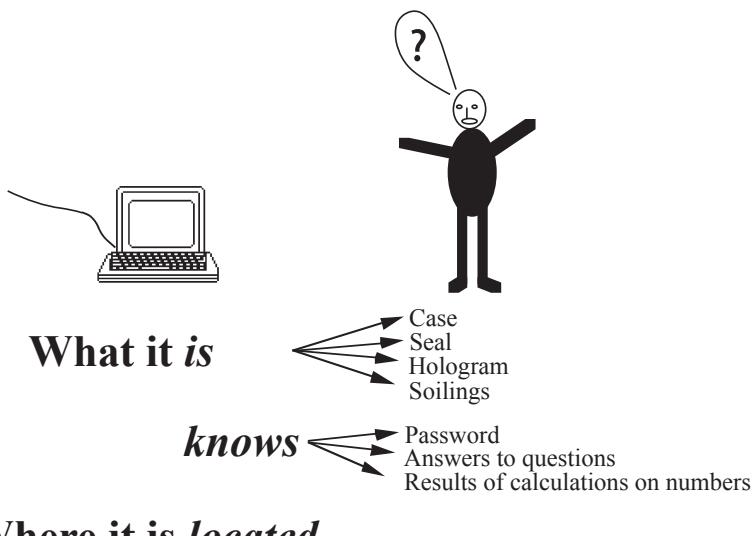


Figure 2.3: Identification of IT systems by humans

## 2.2 Protection of isolated computers from unauthorized access and computer viruses

An IT-System can recognize another IT-system by what it **knows** and **where** the wire comes from, see Figure 2.4. The last one might offer a sufficiently effective identification if we think about components in the same printed circuit board, but would be completely insufficient in open communication networks.

It is important that communication between IT systems is not restricted to comparably insecure algorithms as pass-phrases, Question-Answer-DIALOGUES, and Challenge-Response-Systems like “a number is sent, a calculation result is expected”. Cryptography protocols of high complexity and security must be applicable. Here is an example:

*example:* IT-system 1 generates a random number, sends it to IT-system 2 and expects a ciphertext encrypted with a symmetric block cipher<sup>4</sup> for which only the two systems know the key. If IT-system 1 has received the expected ciphertext, then it **knows** IT-system 2. Inversely, IT-system 2 can identify IT-system 1 (task 3-18a gives tips about attacks which cannot be warded off through this method).

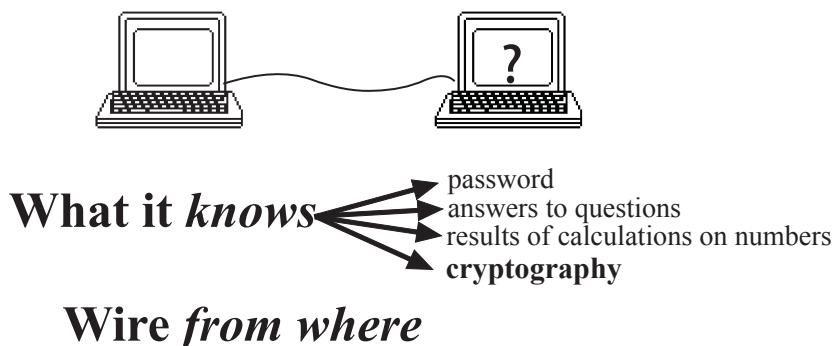


Figure 2.4: Identification of IT systems by IT systems

### 2.2.2 Admission control

**Admission control** describes the behavior of an IT-system, where it requests for the identity of its communication partners and proceeds with the communication process with just the authorized users. These actions impede unauthorized usage of its resources and maybe even much worse things.

It is remarkable that for many services this identity has not to be bound to a real person and pseudonyms can be used instead. [Chau\_87, PWP\_90]. How this happens, is covered in §6.

It should be mentioned that even for identification we cope with a symmetric problem. It is in the interest of the human as well, to not waste his time by spending it with fake

<sup>4</sup>Using a given alphabet, a **block cipher** is capable of en- and decrypting blocks of fix length, whereas a **steam cipher** can do this using strings of **any** length, cp. §3.8. Block ciphers do normally have keys of fixed length (e. g. 128 bit long), which are used to en- and decrypt millions of blocks where even an adaptive active attacker should not be able to break the system.

IT-systems or telling them his secrets.

### 2.2.3 Access control

**Access control** describes the behavior of an IT-system where it does not grant everything to authorized users: Each **subject** (human, IT-system, process) holds only specific rights to run operations on **objects** (processes, data, peripheral devices, etc.). A preferably small and well outlined part of the IT-system checks, before the execution of all processes, if the evoker has the necessary rights. This part of the system is called the **access monitor**, cp. Figure 2.5. The access monitor memorizes submitted or implicitly arising rights and has to be able to detect their annulment.

The allocation of rights itself is usually called **authorization**. Certainly, it must be protected at least as well as the actual access to objects. Because you often can not be aware in advance of the rights to be granted or you do not want to workout a formal specification, you grant rights too easily. Therefore, the access monitor is very often used to **log** all or an appropriate set of operations. When a data security officer possesses suitable programs to evaluate the log files, you can hope that he is able to detect violations of the rights, defined in the world of the IT-system, and induce consequences when applicable.

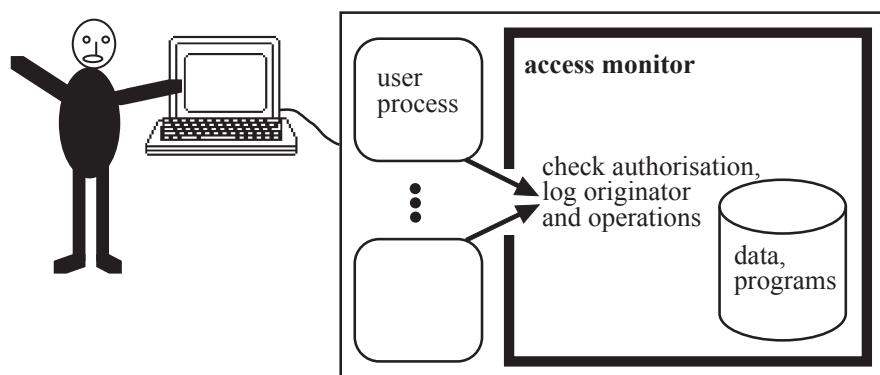


Figure 2.5: Access control by an access monitor

Two facts to be emphasized:

The access monitor controls the access to programs and data (including output) from computer processes that represent people or IT-systems. It is not the human that directly accesses programs, data, or devices.

Running on a specific operating system machine, user processes do not always perform desired actions or even what they seem to do - because they can contain Trojan Horses. Therefore, the access monitor sits at the right place to gather its information. It is located between the user and the user process on the one side, and the data and

## 2.2 Protection of isolated computers from unauthorized access and computer viruses

programs which can be owned by others, on the other side. This will be explained more in detail in the “Computer viruses” example in the following paragraph.

If we demand help from access control, not only with confidentiality and integrity, but also with the execution of **availability**, then it is not enough when the access monitor checks only the presence of rights. It has to also watch the frequency of their usage and analogical restrictions. In general: To achieve **availability**, admission control is not enough. The usage of resources has to be monitored too, and certainly the usage of objects underlying the admission control.

### 2.2.4 Limitation of the threat through “computer viruses” towards the one of “transitive Trojan Horses”

Those “**Computer viruses**” that are widely discussed and published about, are special Trojan Horses which are categorized by the mechanisms that they disperse:

*example:* If an executed program which contains a computer virus has the write access to an arbitrary program, it can alter the program in a way that if executed, a copy of the virus (optional changed) is executed together with this program.

Computer viruses can spread within the transitive shell of (mostly generously granted) write access rights.

General security measures to avoid the propagation of computer viruses already existed before viruses were invented [Denn\_82] (page 317f):

*example:* Protection can be achieved by limiting the access rights of a program during execution to the necessary minimum (principle of least privilege).

To protect a program from unknown manipulation outside the area of physical security, all programs are digitally signed by the creator, cp. §3 and [CIPf\_91, Cles\_88]. Before a program is executed, this digital signature has to be checked to ensure that the program has not been changed in an unauthorized way.

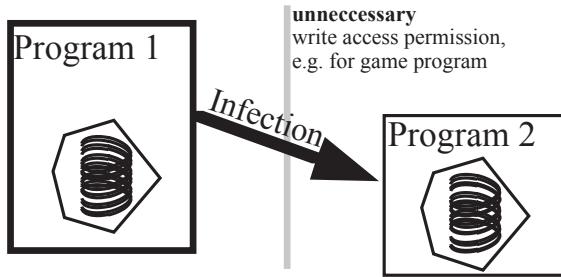
As the threat of computer viruses can be reduced to just transitive Trojan Horses through these security measures, computer viruses are not a research topic in the field of Computer Science. Unfortunately, these general security measures were not instated, partly because of mental laziness, and to a smaller part, because of the cost which such measures require. The result is a serious problem, as a huge amount of computers are not protected against computer viruses.

Annotation with an explanation: It is foreseeable that all attempts of coping with computer viruses without enforcing the principle of least privilege will fail:

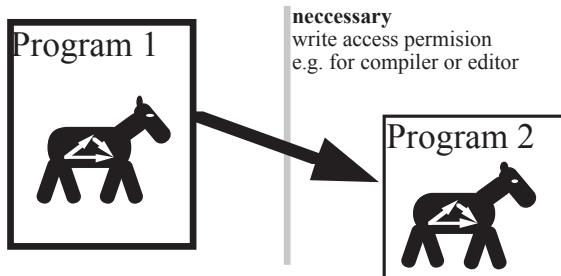
1. lemma: It **can not be decided**, whether a program contains a computer virus or not.

Indirect proof (cp. [Cohe\_84, Cohe\_87] see also [Cohe1\_89, Cohe2\_92]). Assuming we have a decision procedure, decide(), which returns TRUE for each program

## Computer Virus



## transitive Trojan Horse



restriction of attack distribution by least possible privileges:

No unnecessary access permissions!

--> No computer viruses,  
only transitive Troj. Horses

Figure 2.6: Computer virus, transitive Trojan Horse and restriction of security threats caused by computer viruses to threats caused by transitive Trojan Horses

that is a computer virus<sup>5</sup>, and FALSE otherwise. Then we get a false result for the following program:

```
example: program showOpposite;
    if decide(showOpposite) then doNotExecuteVirusDefinition
    else executeVirusDefinition;
```

---

<sup>5</sup>Here (as in many explanations) we have to pay attention to the exact wording of the definition and the statement. The above definition of computer viruses says (one word highlighted):

*example:* If an executed program which contains a computer virus has write access to an arbitrary program, it **can** alter the program in a way that if executed, a copy of the virus (optionally changed) is executed together with this program.

Hence we do not ask if a program **contains** a virus definition, we ask if it is **capable of executing** it. Only for such a case does the the above program **example** gives an opposing example.

## 2.2 Protection of isolated computers from unauthorized access and computer viruses

Hence, no such procedure as `decide()` can exist.

2. Because computer viruses are special Trojan Horses, you can say:  
It can not be decided, whether a program is a computer virus or not.

Therefore, generally it is not possible to detect computer viruses or Trojan horses. There is only one way to protect oneself: Restrict the access of a program during execution and/or do not execute suspicious programs. In short: Instead of exact determination, it is better to treat a good natured program as potentially dangerous!

Unfortunately, two further modest aims (detection of **known** computer viruses and the determination of the damage done to data after identification of a virus attack) that we would like to achieve, are not within reach:

3. statement: Even for **known** computer viruses there exist no effective detection mechanisms!

Explanation: computer viruses can modify themselves using events which occur during runtime and that are therefore unknown to the detection mechanism. The detection mechanism would have to analyze **all** possible self modifications of the computer virus. Given an intelligent programming of the **known** virus, this is not feasible. If we now use a virus scanner to estimate and search for short patterns, there will barely be a program, that is not suspected of containing a computer virus. Prototype tools (which are not very effective yet) are known between relevant people, e. g. Mutation Engine [[Adam2\\_92](#), [DuDR2\\_92](#)].

4. statement: Even for **known** Trojan Horses there exist no effective detection mechanisms!

explanation: same as in 3.

5. statement: It is not possible to determine with certainty, if, after a computer became infected by a computer virus, unauthorized changes were made to the data and/or if it was passed on without permission.

explanation: After an execution of the so-called damage function, the computer virus can modify the damage function and the modification mechanism as well.

### 2.2.5 Remaining problems

1. You have to specify what an IT System has to do and what it **should refrain from**. The latter one is often forgotten.
2. Next the **complete correctness of the implementation** has to be proved. Due to the matters of complexity this is often not feasible.
3. If you could achieve these things, you still have a problem left: you do not know if you have been able to **detect all hidden channels**.

## 2 Security in Single Computers

The first and the third problem cannot be solved with certainty because it is always possible to miss something. Also the second problem has not been solved satisfactorily yet. Therefore, it is recommended that:

*example:* IT-systems should be generally designed and implemented as distributed systems in a way, that an attacker, who has the control of a limited number of computers (as a provider or a user of an universal Trojan Horse), is not capable of serious harm.

## Recommended readings

Physical security measures: [Chau\_84, Clar\_88, Wein\_87]

Identification: [DaPr\_89, §7] [Mill3\_94, DuD3\_99]

Admission control: [Chau\_87, PWP\_90]

Access control: [Denn\_82, ZSI\_89, ZSI\_90]

### Actual research literature:

Computers & Security, Elsevier

Congress series IEEE Symposium on Security and Privacy, EDORICS, IFIP/Sec, VIS; Congress volume published by IEEE, as part of the series LNCS published by Springer-Verlag Heidelberg, at Chapman & Hall London, at the series “Datenschutz Fachberichte” published by Vieweg-Verlag Wiesbaden

### Reference to actual research literature and comments on recent developments:

<http://www.ieee-security.org/cipher.html> as newsletter at cipher@ieee-security.org (with subject line “subscribe”)

Datenschutz und Datensicherheit DuD, Vieweg-Verlag

# 3 Cryptography Basics

If an IT-system can not (or should not<sup>1</sup>) be completely physically protected, then cryptography is the efficient addition for accomplishing confidentiality and integrity. Cryptography does not contribute to availability, or just indirectly at best.

If efficiency is not overly important, then an even further-reaching – but by far not comparably well researched – aid is available through steganography (see §4).

Conceptually, **cryptology**, i.e., the science (in the Grecian meaning of the word: the wisdom) of the cryptographs, can be divided into **cryptography** and **cryptanalysis**: Cryptography describes how cryptographs work. Cryptanalysis tries to analyze cryptographs – in other words, decipher them.

## 3.1 Systematics

We differentiate the most important cryptographic systems with the following three criteria:

1. The **purpose**.

Here we consider mainly

- **encryption systems**, i.e., systems to assure the nondisclosure of messages (simply called: cryptosystems). Hence these systems serve the protection goal of confidentiality.
- **authentication systems**, i.e., systems that protect messages from being tampered with without the recipient noticing. This is similar to error detection codes, only against malicious (intentional) errors. Hence the protection goal these systems serve is integrity. A particularly special case would be
  - **digital signature systems**, which assure that the recipient of a correctly “coded” message can not only be sure that the message was actually sent by the alleged sender, but can also prove this to an outsider, e.g., a court.

---

<sup>1</sup>This is especially the case for computer-networks: Let us assume that to accomplish physical protection we will need one guard every 100m to keep a view on the line. Let us further assume that we would like to have security around-the-clock, requiring a 4-shift-operation. Let us further assume that guards would work for the very low gross wage of 25,000 Euro/year. Assuming all of this, physical protection would cost 500,000 Euro per year and kilometer. This means that working through this chapter and purchasing a few cryptographic devices (or even only software) would quickly pay for itself. You can of course compute the reduction of costs gained by replacing the human guards by dogs and by replacing continuous watching with physical obstacles such as subterranean placement of the cable (and then only hourly visual inspection), but as much as you compute, you will never something as economical as cryptography - that is my promise as a cryptographer

### 3 Cryptography Basics

Apart from these two there are e.g., pseudo-random-bitstream-generators (see §3.4) and collision-resistant hash-functions (see §3.6.4 and §3.8.3) [Damg\_88] and much more.

#### 2. The necessary key distribution.

This differentiation mainly applies to those cryptographic systems that have two kinds of parties, senders and recipients (like encryption and authentication systems). Here one distinguishes between **symmetric** and **asymmetric** systems, depending on whether both parties use the same or different keys (see below).

#### 3. The security-level.

The main differentiation here is between **information-theoretical** and **cryptographic** security. The former is absolute (as far as the cryptographic system is concerned, so for example, not with key theft). The latter is weaker, but can be reasonably subdivided (see below).

Two limitations of *encryption systems* should be mentioned right at the start:

- They only protect the confidentiality of the *content* of the message, not information about the communication itself such as who communicates from where, when, and with whom. If this needs to be protected as well, further protection measures are necessary (see §5).
- Also, often the message content is not protected completely, but some attributes of the content, for instance its length, are simply ignored<sup>2</sup>. This becomes especially obvious when proving perfect security of the Vernam cipher (see exercise 3-10b).

What might have caused these widespread – and by most of the specialists unnoticed – failures (stated nicely: inaccuracies)? Maybe the following procedure:

On the highest design level – the requirement-definition – the following was determined: Messages (atomic objects on this level) need to remain confidential. Later the atomic type “message” gets sophisticated on lower levels, by deciding on the representation of messages and thereby implicitly introducing, for instance, the length of messages. Eventually – on the lower level – the requirement “confidentiality of messages” is only based on the corresponding main-attribute, implicitly introduced auxiliary-attributes – e.g., “length” – either do not come to the picture or are considered unimportant.

What do we learn from this? All sophisticated, and all implicitly introduced, attributes of an object that needs to remain confidential, also need to remain confidential.<sup>3</sup> Thus

---

<sup>2</sup>If some attributes of the content of a message are ignored, one needs to ensure that those attributes do not give away “too much” information about the contents of the message. A drastic example would be unary coded message-content together with the unprotected attribute message-length. Then the message-length gives away everything about the message-content, so that encryption for instance with the Vernam cipher would be completely useless (**Unary coding** means: There is only one code-sign – for example 17 would be represented by writing this code-sign 17 times).

<sup>3</sup>This is also true for derived attributes of other objects, as the following two examples show:

one confidentiality-requirement becomes *several* in the course of sophistication. If this seems too costly for the designer (i.e., the sophisticator), then he needs to ask the originators of the requirement-definition if they accept the non-implementation of the confidentiality-requirement concerning some attributes. However this sounds much easier, than it probably will be, as the designer first needs to explain his (sophisticated) world to the originators of the requirement-definition. And normally the originators of the requirement-definition do not want to occupy themselves with this “world”.

The most important – in the following considered – cryptographic systems consist of three algorithms (e.g., key generation, encryption, decryption in Figure 3.1), which may be publicly known. Next the functions of the different systems are illustrated together with the required key generation and -distribution.

### 3.1.1 Cryptographic systems and their key-distribution

Initially a survey of symmetric and asymmetric encryption systems is given, and subsequently, a survey of symmetric and asymmetric authentication-systems.

All four images are highlighted by the following basic structure (see Figure 3.1): To the left and to the right resides the **domain of trust** of the respective participant<sup>4</sup>. In the lower middle, there is the **area of attack**, i.e., the region where attacks will be warded off with the cryptographic system, as they are not prevented (or at least defeated) there by physical and/or organizational measures.

- 
1. For the in §3.2 described Vernam cipher, the presentation of the ciphertext that the attacker discovers is not allowed to depend on the composition of the modulo-sums. So for the binary Vernam cipher  $0 \oplus 1$  needs to look exactly like  $1 \oplus 0$  concerning all properties – analog signal-levels, timing etc. (the same applies for  $1 \oplus 1$  and  $0 \oplus 0$ ). In practice this is easily achievable, for instance by temporarily storing the result before outputting it – in a software-implementation this happens automatically in every computer. But in an hardware-implementation one needs to remember this.
  2. Also, timing is crucial for the software implementation of cryptographic systems – e.g., the execution-time of cryptographic operations must not depend on the value of the secret key. This can be achieved – while programming cryptographic software – by making sure that the value (or even parts of the value) of the secret key do not appear in any jump-condition. Alternatively it can be ensured via time-queries that all cryptographic operations take the same time.

Both examples have been known for a long time – but seemingly not continuously heeded in practice. Otherwise the publication of P. Kocher about *Timing Attacks* would not have been met with such a broad response. Find out more about these attacks in [EnHa\_96].

<sup>4</sup>This confidence level needs to be assured respectively by non-cryptographic measures, in particular it needs to be delimited (and if necessary to be defended) by physical and organizational measures. In the confidence level, only things that are authorized by the respective participant should occur. And only information authorized by the respective partner should leave the confidence level. This means that the confidence level needs to be shielded (see §2.1): No analyzable electromagnetic radiation may leave it, and also its input – for instance its energy consumption – may not depend on the things that happen in it. If this is not respected, then, for instance, the varying energy consumption of individual transistors can possibly be used to access secret keys [Koch\_98]. Hence, confidence levels should be decoupled as much as possible from their environment. A confidence level is generally realized through a – (hopefully) for the respectable participant trustworthy – end device [PPSW\_95, PPSW\_97].

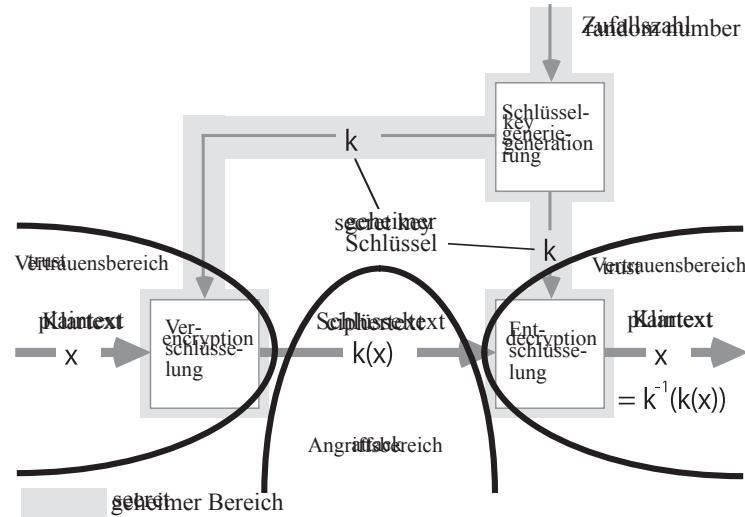


Figure 3.1: Basic scheme of cryptographic systems shown with the example of a symmetric encryption system

### 3.1.1.1 Cryptographic Systems, Overview

The oldest and best known type of cryptographic systems is the **symmetric encryption system** (see Figure 3.2).

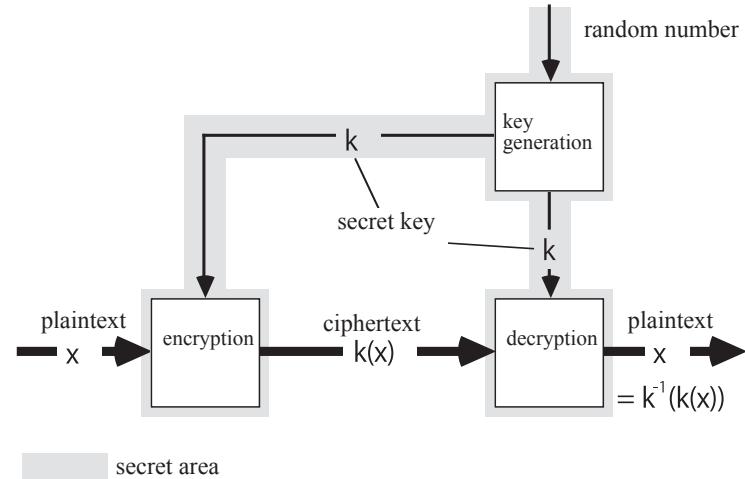


Figure 3.2: Symmetric encryption system (black box with a lock; two identical keys)

*Annotation* The notation  $k(x)$  is just an abbreviation because the  $k$  that is generated by the key generation is normally not a whole algorithm. It is just a key, i.e., a

parameter which is put into some default en- and decoding-algorithms, e.g.,  $enc$  and  $dec$  (hence, in reality  $enc$  and  $dec$  would be programmed once and for all or implemented in hardware). Then the ciphertext is

$$S := enc(k, x)$$

and the decrypted text is

$$dec(k, S) = dec(k, enc(k, x)) = x.$$

The problem with symmetric encryption systems is that if a message needs to be encrypted before it is sent over an insecure net, the key  $k$  first need to be presented to the sender and recipient.

If sender and recipient have previously met each other in real life, they could have exchanged  $k$  on that occasion.

But if one just takes the net-address of the other one (e.g., the service-provider in a public system) from a directory and both want to communicate confidentially now, it becomes difficult. In practice one usually assumes the existence of a trustworthy “**key exchange center**”  $X$  to solve this problem. Every participant  $A$  exchanges a key  $k_{AX}$  with  $X$  upon registration with the open system. If  $A$  now wants to communicate with  $B$  and does not yet have a common key with  $B$ , then  $A$  requests a key from  $X$ . Then  $X$  generates a key  $k$  and sends it to both  $A$  and  $B$ , encrypted with  $k_{AX}$  and  $k_{BZ}$  respectively. From then on,  $A$  and  $B$  can use the key  $k$  to send messages both ways. Of course this is not very confidential: Apart from  $A$  and  $B$ ,  $X$  can also decrypt all messages (and it is even very likely that  $X$  can read all messages, as the operator of the key exchange center and the network operator are often the same person).

This can be improved by utilizing *several* key exchange centers in such a way, that they can only read the messages if they all work together (see Figure 3.3). Again, every participant  $A$  exchanges a key  $k_{AC}$  with every center  $C$  in the beginning. If  $A$  wants to communicate with  $B$ , then  $A$  asks every center to generate a fragment-key. That center sends the fragment-keys  $k_i$  again confidentially to  $A$  and  $B$ . Then  $A$  and  $B$  use the *sum*  $k$  of all  $k_i$  (in an appropriate group) as an actual key. To finally get, for example, a 128 bit long key, every  $k_i$  needs to have length 128 bit, and  $k$  could be the bit-by-bit XOR of all  $k_i$  (or also the sum modulo  $2^{128}$ ). As long as at least one of the centers keeps its fragment-key  $k_i$  secret, the other centers have no information about  $k$ , as by adding an appropriate  $k_i$ , still all  $k$  can be reached (This means that  $k$  is still perfectly encrypted with a Vernam cipher from the point of view of the other centers, see §3.2).

**Asymmetric encryption systems** were invented to simplify this key-distribution (see Figure 3.4).

Thus different keys  $c$  and  $d$  are necessary for en- and decoding, and only  $d$  needs to be kept secret.

*Remark 1:* Again, the notations  $c(x)$  and  $d(c(x))$  are just abbreviations. If  $enc$  and  $dec$  are again en- and decoding-algorithms, then  $c$  and  $d$  are actually input-parameters for these algorithms. The ciphertext is  $S := enc(c, x)$  and the decrypted text is  $dec(d, S) = dec(d, enc(c, x)) = x$ .

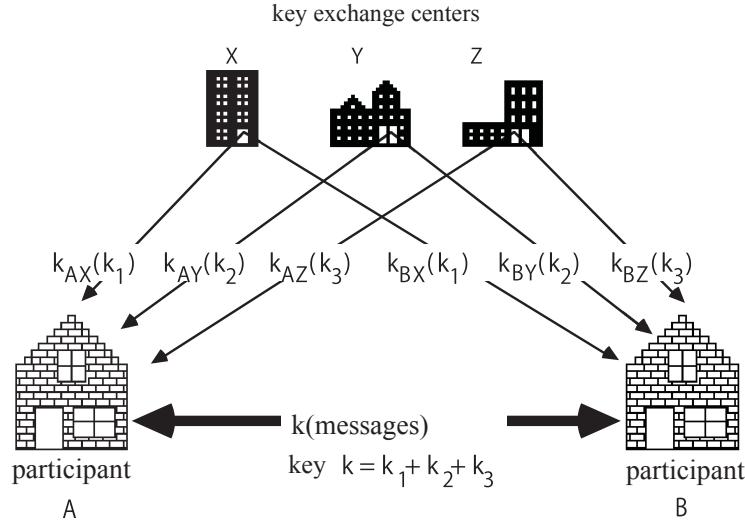


Figure 3.3: Key exchange for symmetric encryption systems

*Remark 2:* The random number in the lower left of the Figure 3.4 has the following meaning: Let us assume that the encryption is deterministic. If an attacker then sees a ciphertext  $S$  and knows some plaintexts  $x_i$  (of which probably one is the correct plaintext for  $S$ ), then he can check this himself by creating all  $c(x_i)$  and comparing them to  $S$  (This is certainly hardly possible for long private letters, but it is possible for standard-steps from protocols, e.g., for payment transactions). For this reason, the encryption must be made **indeterministic**, i.e.,  $S$  must also depend on a random number that can not be guessed by the attacker (this is also known as **probabilistic encryption**).

This random number can be understood as part of the plaintext. Because the decryption-algorithm acquires the complete plaintext, it also obtains the contained random-number, but does not give it to the outside.

To assure that  $c$  really does not need to be kept secret,  $d$  may not be gained from  $c$  with sensible effort (this will be specified more in detail later, in security concepts and in §3.6).

Now, every user  $A$  can generate (in any case) his own pair of keys  $(c_A, d_A)$  and must not communicate  $d_A$  to anyone else. Further on,  $c_A$  just needs to be distributed in such a way, that every other user  $B$ , who wants to send a confidential message to  $A$ , has  $c_A$  or can find it. If  $A$  and  $B$  previously knew each other, they can exchange  $c_A$  privately;  $B$  can now write  $c_A$  openly in his address-book. Also, users can pass  $c_A$  on to other users.

For connections with unknown persons,  $c_A$  could also stand in the same directory in which  $B$  also looks up the net-address of  $A$ . If there exists no such directory outside of the net, then a key register inside the net can take the place of the directory (see Figure 3.5).

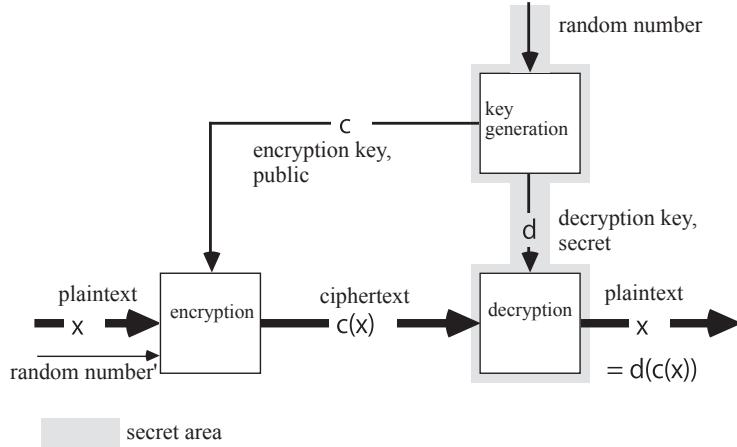


Figure 3.4: Asymmetric encryption system  $\approx$  Black box with a spring lock, there is only one key: Everyone can put something in, but only the key owner can take it out.

Thereby, the analogy between an asymmetric encryption system and a black box with a spring lock reaches its limits when the lock has clicked. After that, nothing can be put into the box without the assistance of the key owner. With a asymmetric encryption system this is always possible, i.e., after encryption of one message additional messages can be encrypted. So, an analogy with a black box with a lock, for which only one key exists, and where the box has a slot for putting things in, would be better. The slot must be properly designed, i.e., it must be impossible to take things out through the slot, and one must not be able to see the box's contents through the slot. But this analogy needs one additional mechanical security feature)

Be aware, that in the case of Figure 3.5, the key register  $R$  (or its operator respectively) can not decipher the message. Indeed there is a changing attack-possibility for  $R$  (see exercise 3-6c)).

In order that participant  $B$  can be sure that  $c_A$  really belongs to participant  $A$ , the public-key register does not just authenticate  $c_A$  with its signature but also the *correlation* between  $A$  and  $c_A$ . Otherwise an attacker could intercept the response message (step 3. in Figure 3.5) from the key register and send another “authenticated” key to  $B$ , for example a key to which the attacker knows the accompanying decryption-key.

### 3.1.1.2 Authentication-systems, overview

A **symmetric authentication-system** is illustrated in Figure 3.6.

So the message is not being obliterated by the left cryptographic algorithm in Figure 3.6 here, but a verification part is annexed to the message. This verification part is often called MAC. The receiver can also create the correct MAC (using  $x$ ) and check, if the MAC that came with the message is the same.

*Annotation* Again, the notation  $k(x)$  is just an abbreviation. If the cryptographic algorithm is called *code*, then  $MAC:=code(k,x)$  and the receiver of a message  $(x,MAC)$  tests, if  $MAC=code(k,x)$ .

### 3 Cryptography Basics

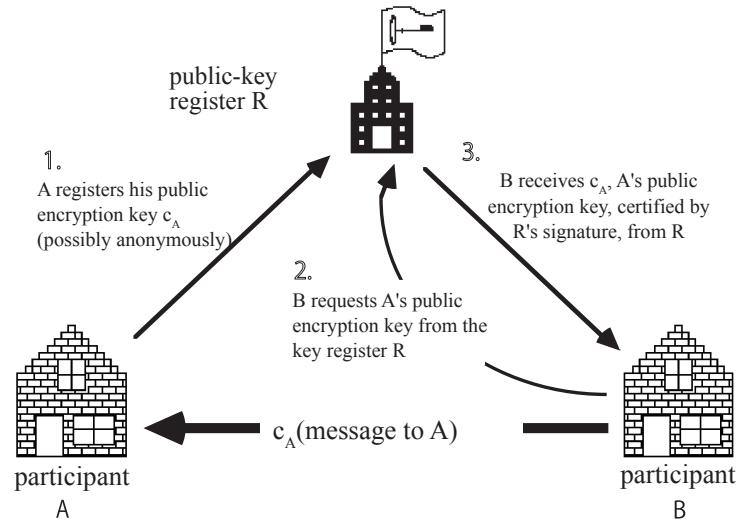


Figure 3.5: Key distribution for asymmetric encryption systems by a public-key register

The key-distribution can be handled in the same way as the key-distribution for symmetric encryption systems. Also, there are the analog problems, e.g., a single key exchange center could now sign faked messages.

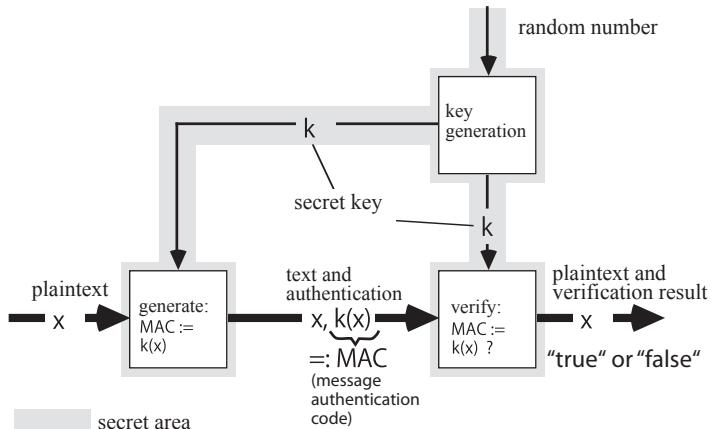


Figure 3.6: Symmetric authentication system ( $\approx$  Glass box with a lock, there are two identical keys for putting things in)

Asymmetric authentication-systems, also called **digital signature systems**, next simplify the key-distribution analog to the asymmetric encryption systems (see Figures 3.7 and 3.8).

But their **main advantage** is different: Namely that the receiver  $B$  of a signed message from  $A$  can now prove to anyone else (who also knows  $A$ 's key  $t_A$ ), that this

message came from  $A$ . This is not possible with a symmetric authentication-system, even if the key exchange center confirmed the keys that  $A$  and  $B$  to a court, for example: In a symmetric authentication-system,  $B$  could as well have created the MAC himself. But in digital signature systems,  $A$  is the only one who can create the signature.

Therefore, digital signature systems are inevitable, when it comes to relevant legal things that need to be handled digitally in a secure manner, e.g., in digital payment systems. There, they correspond to the role of the handwritten autograph in today's legal transactions (hence the name of course). In the following, the notions "autograph" and "signature" are widely interchangeable.

The just described main advantage of digital signature systems, concerning integrity, is always compared to a disadvantage in regards to confidentiality: The receiver of a digital signature can show the message and the signature around – and if the key to test the signature is public, *everyone* can test the validity of the signature. Depending on the content of the message, this can be much more unpleasant for the signer of the message, than if the message could just be shown around behind his back – anybody can make up arbitrary messages and spread them as non-verifiable rumors. This verification of the integrity of a message behind the sender's back is not possible in symmetric authentication-systems: In such systems, the receiver of the message could also have created the MAC, so that third persons can not verify the authenticity of the message in symmetric authentication-systems. Please note that there exist authentication systems that combine both advantages: In **undeniable signatures**, the assistance of the signer is necessary to verify the signature, so that verification behind his back is impossible. But on the other hand, it then needs to be stated under which circumstances the (professed) signer is obliged to actively assist the verification of "his" signature. Because otherwise the receiver of the signed message could (in case of a conflict with the sender) not take any advantage of the fact that the message is signed, because he could not convince anyone of this fact (find more information about this type of authentication-systems in §3.9.4).

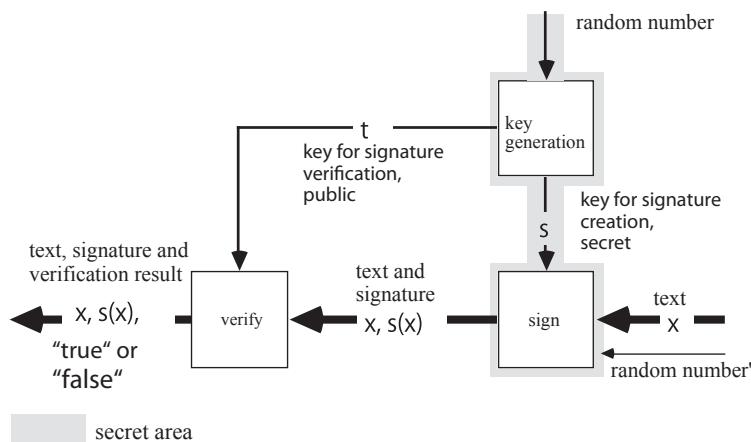


Figure 3.7: Digital signature system (Glass box with a lock; there is only one key for putting things in)

### 3 Cryptography Basics

*Remark 1:* In detailed notation, signing- and testing-algorithms could be called *sign* and *test*. The signature is  $Sig := sign(s, x)$  then, and the receiver of a message  $(x, Sig)$  computes  $test(t, x, Sig) \in \{true, false\}$ .

*Remark 2:* The random number in the lower right of Figure 3.7 is chosen independently from the random number above (the reason why this random number is necessary will not be understood until §3.5).

Please be aware that, in contrast to symmetric authentication, a separate testing-algorithm is needed here, as the receiver has the key  $k$  there, with which he can compute the correct value  $MAC$  from  $x$ ; But here, the receiver does not have  $s$ .

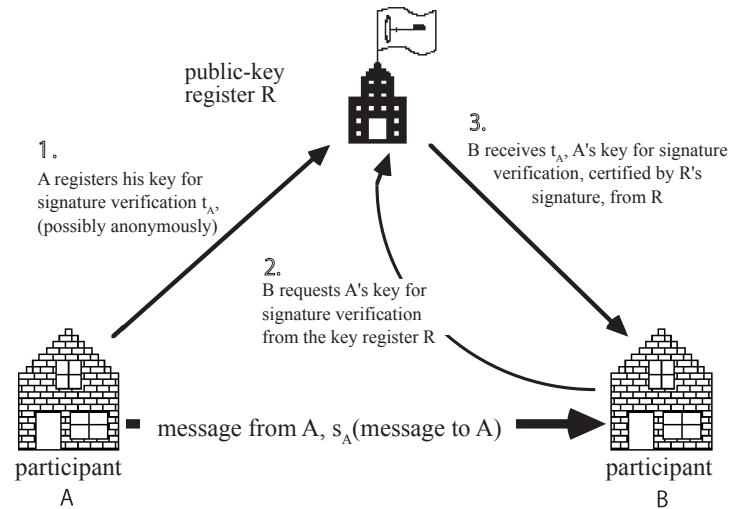


Figure 3.8: Key distribution for digital signature system by a public-key register

Further be aware, that the possibility of exchanging the testing-key privately with a communication-partner only suffices in the case that one trusts this partner, i.e., that one just uses the signature system as a convenient form of mutual authentication. But if one wants to be sure that a signature is accepted by a court (if necessary), then one needs to assure that one has the correct testing-key, i.e., at least ask a key register to control this.

Again, a single key register would have opportunities for changing attacks (compare with exercise 3-6c).

The verification of the public testing-key refers – as with the public decryption-keys – not to the key alone, but to the *correlation* between the key and the participant. If we ignore other information, such as the time-stamp (cp. with exercise 3-6f), then the attestation from the key register  $R$  for  $t_A$  (step 3. in Figure 3.8) is as follows:  $A, t_A, s_R(A, t_A)$ . This attestation of the public key is called **certification of the public key**, the instance that executes it is called **certification authority (CA)**.

Recapitulating, there exists one more requirement for the key-distribution here than for asymmetric encryption systems: It must be **consistent**. This is a common target of several receivers who do not trust the certification authority concerning the fulfillment of this target: Even if the certification authority cheats, all receivers need to get the same value  $t$ . Imagine, a receiver would check with a key  $t$ , but the court would check with an entirely different key  $t'$  later on! Further on, the integrity of the distribution must be assured, i.e., if the certification authority does not cheat, everybody receives the certified  $t$ .

The consistency of the key-distribution should be supported by the fact that the user who wants a public key to be certified, proves the knowledge of the associated secret key. Otherwise, an attacker could present foreign public keys to the certification authority and let them be certified as his own public keys. This would not allow him to execute the secret operation, but the certification of the same public key for several different participants could cause confusion and disorientation. An elegant possibility to prevent all this for digital signature systems is, that in the example,  $A$  does not present  $t_A$  for certification but first of all **certifies in person** the correlation between his name  $A$  and his public key  $t_A$  using the secret key  $s_A$ . So he presents  $A, t_A, s_A(A, t_A)$  to the certification authority and receives as attestation  $A, t_A, s_A(A, t_A), s_R(A, t_A, s_A(A, t_A))$ . This is often called the key-certificate.

Recapitulating, the **certification authority** has the following tasks:

1. Verification of the identity of a participant (How thoroughly this is done is defined by the certification authority in a so called *certification policy*. Often this is done by the submission of an official identity card).
2. Verification of the application of the participant (How thoroughly this is done is once again defined by the certification authority in the *certification policy*. Often this is done by a written, personally signed contract, and if necessary, the signature of the participant is even notarized).
3. Verification of the uniqueness of the name (that is about to appear in the key-certificate) of the participant.
4. Verification that the participant can apply the associated secret key.
5. Digital signing of the name of the participant and the public key. If necessary, the addition of the public (itself again certified) key of the certification authority.

### 3.1.2 Further annotations to the key-distribution

The important classes of cryptographic systems were presented together with their key-distribution in §3.1.1. Three facets of the key-distribution are going to be illustrated here, before we will discuss the denotation of security in cryptographic systems in §3.1.3.

### 3 Cryptography Basics

#### 3.1.2.1 To whom are keys assigned?

Here the distinction between three possible answers is interesting:

1. single participants,
2. pair-relations, or
3. groups of participants.

In *asymmetric* cryptographic systems, pairs of keys are usually assigned to *single* participants. The secret key ( $d$  or  $s$ ) is then only known by the single participant, the public key ( $e$  or  $t$ ) is potentially known by everybody (see Figure 3.9).

In *symmetric* cryptographic systems, keys are usually assigned to *two* participants, i.e., to a *pair-relation*: If we disregard the (for the key-distribution unimportant) case in which someone sends messages to himself, (see exercise 3-9c), then always at least two participants need to know the key in symmetric systems. As symmetric keys need to remain confident, they should only be known by the least possible number of persons. From this follows that every symmetric key is known by exactly two participants.

Thus, (except for special circumstances) keys are not assigned to groups, but either to single persons or to pair-relations.

knowledge about the key crypto-graphic systems and their goals	one person (generator)	two persons (both partners)	everyone (publicly known)
asymmetric concealment system: confidentiality	$d$ decryption key		$C$ encryption key
symmetric cryptographic system: confidentiality and / or integrity		$k$ secret key	
digital signature system: integrity	$s$ signature key		$t$ verification key

Figure 3.9: Goals and key distribution of asymmetric encryption system, symmetric cryptographic system and digital signature system

#### 3.1.2.2 How many keys need to be exchanged?

If **n** participants communicate confidentially with each other, they need  $n$  pairs of keys when an **asymmetric** cryptographic system is used. If they want to digitally sign their messages,  $n$  additional pairs of keys are needed. Hence, altogether **2n pairs of keys** are needed (If participants shall be able to stay anonymous and if their public keys shall

not be person-tags (see §6), then every participant does not only need two, but many digital aliases, i.e., public keys).

If  $n$  participants use **symmetric** cryptographic systems and if everyone wants protected communication with everyone, then they need  **$n \cdot (n-1)$  keys**. It is assumed that keys are assigned to *pairs of participants* (see §3.1.2.1) and that a separate key is used for every direction of the communication (compare with the solution of exercise 3-10e). Thus, the participants will often exchange two symmetric keys in one step, so that the **key-exchange** will only occur  **$n \cdot (n-1)/2$**  times.

In particular for IT-systems with many participants, the following question comes up: **When** shall the (pairs of) keys be generated and exchanged?

In asymmetric cryptographic systems, the generation can happen without problems before the actual operation of the IT-system or – if the participants exchange – it can happen before the registration of the single participant. The distribution of the public keys can take place on demand.

In symmetric cryptographic systems, the generation or even the distribution can (at least in IT-systems with many participants) not happen for every possible communication connection because of the substantially higher number of necessary keys. Hence, only the keys between the participant and the key exchange center(s) are created and exchanged (when he registers) in such systems. Then all other keys can and must be created and exchanged on demand. This saves considerable effort, as in IT-systems with many participants, generally just an infinitesimal fraction of all possible communication-connections is ever seized.

### 3.1.2.3 Security of the key-distribution limits the cryptographically reachable security

Because a secure primary-key-distribution outside the communication-system that is to be protected, is assumed in the following, its importance shall be emphasized here:

The – through the use of a cryptographic system – reachable security is *at most* as good as the security of the primary-key-distribution.

Therefore it is reasonable and – as shown in §3.1.1.1 and in the exercises 3-6c) and 3-6d) – possible, not to assume just a single primary-key-distribution, but several. Thereby, the reachable security can be significantly improved.

**Remark:** The security of the key generation also limits the security reached through cryptographics:

A secret key ( $k$ ,  $d$  or  $s$ ) can be at most as secret as the *random number* from which it was created. This random number must therefore be – if possible – created for the future owner of the key and must not be guessable by an attacker. Date or time may in no case be chosen as a *random number* and one must not use the simple function *random* in ones programming-environment. Generation of good random numbers is not a trivial matter.

One option is **physical phenomena**. For this, special hardware, which not every personal computer possesses, is needed. Examples for this are noise-diodes, radioactive processes or turbulences in the fan of an ordinary computer [GIVi\_88, DaIF\_94].

If only a few real random numbers are needed, then they can also come from the **user**. Firstly, he could try this with dice. This might be a bit exhausting, but for the creation of a signing-key for really important things, it is possibly the best idea. Secondly, one can just let the user type in characters "randomly" and then compress the input inside the computer. Thirdly, sometimes the most low-order bits of the distance between the keystrokes is used instead of the entered characters, as they are more randomly. But this is problematic, at least in combination with the much too low sampling rate of some operating systems.

If one does not completely trust any of the existing methods, one should create several random numbers using different methods and then use their **XOR** (see Figure 3.10). If only one of the numbers is random and secret, then it is proven that also the XOR is random and secret. One should proceed in such a way, particularly if the government is interested in being involved in the creation of random numbers because it fears that some users could enter "bad" random numbers, since the government can not demand that users trust in foreign random numbers.

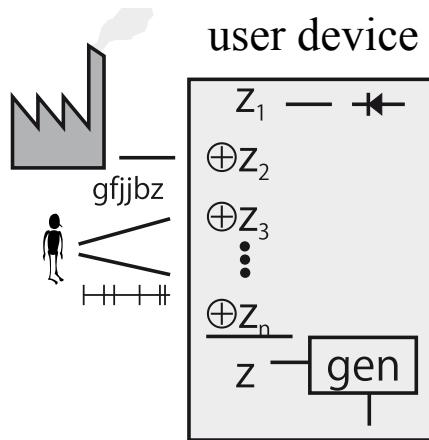


Figure 3.10: Generation of a random number  $z$  used for key generation: XOR of random numbers generated by a device, received from a manufacturer, rolled by the user, or calculated with time intervals

### 3.1.3 Basics of security

#### 3.1.3.1 Attack-aims and -successes

During the establishment of a cryptographic system, four phases can be distinguished:

1. Definition of the cryptographic system

2. Choice of the security-parameter
3. Choice of the key(-pair)
4. Processing of messages

While the break can already be investigated after the first phase with general complexity-parameters, the investigation for concrete effort (that can be yielded by an attacker) is only profitable after the second phase.

For the following attacks, it is assumed that they occur after the third, or even after the fourth phase. The following are differentiated *in distinguished order of attack-successes*:

- a)** Obtaining the key (**total break**)
- b)** Obtaining a procedure that is equivalent to the key (**universal break**)
- c)** Breaking (i.e., faking or decoding) for only some single messages<sup>5</sup> (**message-dependent breaking**).
  - c1)** for a self-chosen message<sup>6</sup> (**selective break**)<sup>7</sup> or
  - c2)** for a random message (**existential break**)<sup>8</sup>.

It is obvious that breaking in terms of a), b) and c1) is not acceptable. If c2) is also not allowed, one is certainly secure enough. But there are weaker cryptographic systems where one just hopes that the reasonable (and for the attacker, interesting) messages are so rare in the total amount of messages, that the attacker will never obtain any of them with his success in terms of c2).

In *encryption systems*, one can further distinguish b) and c):

1. **Complete decoding**, i.e., the attacker obtains the whole plaintext.
2. **Partial information:** The attacker only obtains some characteristics of the plaintext, e.g., single bits or the cross sum of the plaintext.

### 3.1.3.2 Types of attacks

Ordered by *increasing strength*:

---

<sup>5</sup>For “faking”, a single message usually is a plaintext and for “decoding”, it is a ciphertext.

<sup>6</sup>In encryption systems, this is typically an intercepted, encrypted message, of which the plaintext is interesting to the attacker (so the attacker selects the message). If the attacker creates the ciphertext himself, then he would learn nothing about others from the corresponding plaintext. But in authentication-systems he can freely chose a plaintext for authentication. This authentication can be useful for him when he uses it against others.

<sup>7</sup>A clear difference to universal breaking only exists if the selective break includes an active attack, see §3.1.3.2.

<sup>8</sup>In encryption systems, only *selective* message-dependent breaking is reasonable – it is futile to decode random ciphertexts that were sent by nobody. Existential breaking does exactly this.

### 3 Cryptography Basics

**a) Passive.** In encryption systems, this means that the attacker just observes all the time. In authentication-systems, this means that the attacker just observes until he tries to fake a message.

In encryption systems, one further distinguishes:

**a1) ciphertext-only attack:** The attacker only sees ciphertexts all the time.

**a2) known-plaintext attack:** The attacker sees the corresponding plaintexts for some ciphertexts, e.g., because some message are made public after some time. In a reduced manner, this occurs when the attacker can guess parts of the plaintext with the utmost probability (e.g., on the basis of the reaction of the receiver, or on the basis of events that just happened near the sender, or because those parts are typical letter-openings); many classic encryption systems were broken like this.

**b) Active.** The attacker already interferes with the system before the actual attack, e.g., he first makes a signer sign some harmless-looking messages.

One can further distinguish: Which communication-partner is attacked by the attacker? In asymmetric systems, only an attack on the partner with the secret is profitable, as the attacker can encode or test signatures himself.

In *symmetric encryption systems*, one distinguishes:

**b1) plaintext → ciphertext (chosen plaintext attack):** The attacker can choose plaintexts to be encoded and sent by the sender. Example: He visits a bank-agency as a client and executes certain transactions, which are then encoded and sent to the main office.

**b2) ciphertext → plaintext (chosen-ciphertext attack):** The attacker sends ciphertexts to the receiver<sup>9</sup> and see the decryptions. Example: The attacker is an employee in the bank's main office, but is not allowed to know the key. Now he sends weird ciphertexts (with feigned sender-address) and observes what the decryption-computer does with them.

According to the annotation above, only b2) is relevant in *asymmetric encryption systems*.

In *symmetric authentication-systems*, one distinguishes:

**b1) plaintext → MAC (chosen plaintext attack<sup>10</sup>) and**

**b2) MAC → (a possible<sup>11</sup>) plaintext (chosen MAC attack<sup>12</sup>).**

<sup>9</sup>So that the receiver uses the corresponding key while decoding, the attacker usually needs to (be able to) state a (faked) sender-address that fits. If the receiver (in symmetric encryption systems) would use the key that he exchanged with the attacker while decoding, the attacker would not obtain new information – he would know the key and could execute the operation himself.

<sup>10</sup>Until now, this term was only used for encryption systems in literature.

<sup>11</sup>For short MACs, but potentially long plaintexts, single MACs must in average correspond to several plaintexts.

<sup>12</sup>Admittedly, a chosen MAC attack is a bit theoretical: A stolen authentication-device would expect

Only b1) plaintext → signature (**chosen plaintext attack**) is relevant in *digital signature systems*.

### Adaptivity:

- non-adaptive (i.e., the attacker has to choose all his messages in the beginning)
- adaptive (i.e., the attacker can first choose some messages and later some more messages depending on the result, etcetera)

An example may clarify the reason for this distinction: An attacker goes with a crate of beer to a televiser which is not averse to the odd drink and gives him beer until the televiser needs to use the bathroom. As a general rule, the televiser will – contrary to his orders – not take the cryptographic device with him to the bathroom. Now the attacker has the opportunity to process several texts with the cryptographic device – but he will certainly not have the calmness and the time to imagine new clever inputs depending on the outputs of the cryptographic device. So our beer-crate-bathroom-attack is a non-adaptive active attack. If we replace the beer with sleep-tablets however, then we get an adaptive active attack – at least for attackers with good nerves that are able to concentrate in uncomfortable environments.

It is desirable that a cryptographic system allows no or only minimal attack-successes, even for very strong attackers. So the best option is a cryptographic system that can not even be existentially breached even by adaptive active attacks.

#### 3.1.3.3 Connection between observing/modifying and passive/active attacker

While the *authorization* for actions is crucial in the informal definition of the attacker-attributes observing/modifying in §1.2.5, passive/active distinguishes actions independent of whether or not they allowed or forbidden for the attacker. Hence, observing and passive attackers – as well as modifying and active attackers – both do *not* mean exactly the same thing.

For example, an observing attacker can be a passive attacker if he is interested in the contents of communications of a communication-connection which is protected by a symmetric encryption system, in which he is not allowed to interfere. Or he can be an active attacker if he – as in the example above – visits a bank as a client and executes allowed, but self-chosen, transactions, which are then encoded and sent to the main office.

Conversely, a modifying attacker can be a passive attacker if he illicitly obtains copies of the encrypted password-file by bypassing and/or breaking the access-control in another IT-system (active attack) and tries to obtain the plaintext (passive attack)<sup>13</sup>. But if the modifying attacker is also a legal user of the IT-system and if it is also forbidden for him

---

the plaintext as input and it would generate the corresponding MAC – and not the other way around.  
But there is nothing like persevering systematics...

<sup>13</sup>In a certain sense, this is cheating as "changing" and "passive" refer to different things: "changing" refers to the retrieval of the password-file and "passive" refers to the decryption of the encrypted

### 3 Cryptography Basics

to copy the password-file (changing attack), so he is to be considered active, as he may change his password randomly.

The examples are illustrated in Figure 3.11.

Attacker	passive	active
observing	Interested in the contents of foreign communication relationships without interfering that communication	Allowed selection of cleartexts, which are then decrypted with a key unknown to the attacker and transmitted. These ciphertexts are used by the attacker to break the cryptographic system.
modifying	Breaks access control and tries to decrypt foreign encrypted files.	Breaks access control and tries to decrypt encrypted files which enclose cleartext partially choosen by the attacker.

Figure 3.11: Combinations of attacker properties observing/modifying and passive/active

#### 3.1.3.4 Basics of “cryptographically strong”

When possible, one should use *information-theoretical secure* systems in terms of the security of the cryptographic system. That is the symmetric encryption system “Vernam cipher” (one-time pad) (described in §3.2) or the authentication-codes (described in §3.3). The attacker can compute as much as he wants – it will not help him<sup>14</sup>.

Three reasons could force a switch to systems that are only *complexity-theoretical secure*:

- The necessary security-service is not reachable by information-theoretical secure systems, for example a digital signature in the narrow sense (see §3.1.1.2).
- The risk of unauthorized notice of the exchanged key is higher than the additional security gained by an information-theoretical secure cryptographic system compared to an alternative asymmetric (and therefore only complexity-theoretical secure, see below) system.
- The effort of exchanging the necessary keys for information-theoretical secure systems is too big. This effort increases without limit, with the total length of the messages that need to be protected.

---

password-file. If this should be prevented, then passive changing attacks would be those attacks, where something should happen (changing), but does not happen (passive). For example a Trojan horse inside a component could refuse the service of the component to its environment (denial of service attack).

<sup>14</sup>So the security is (as far as the algorithmic parts of the system are concerned) absolute

The next secure systems after information-theoretical systems are called “cryptographically strong”. Independent of the system’s type, one can say several things about the security in terms of cryptography:

1. **Many cryptographic systems are, in principle, breakable:** If keys of fixed length  $l$  are used, and enough information is available to make the key unique, then an attacker-algorithm could in principle always try all the  $2^l$  keys and thereby completely break the system (This affects most asymmetric systems, concerning encryption as well as authentication, as there is generally only one possible  $d$  or  $s$  for every known  $c$  or  $t$ . This also affects most symmetric encryption systems (except the Vernam cipher) for chosen plaintext attacks with enough “material”. Moreover, the same thing holds e.g., for the faking of a signature for a specific message: The attacker knows that his faked signature  $Sig$  for the message  $x$  is good enough, if it passes the test  $test(t,x,Sig)$ . As there generally is an upper limit to the length of signatures, he once again must only try a finite amount of  $Sigs$ ).

This exhausting trying requires an **exponential** amount of operations though, so it is not realistic for e.g.,  $l > 100$ .

But this means, that the best that the developer of cryptographic system may hope for, in terms of complexity-theory, is complexity for the attacker-algorithm which is exponential in  $l$ .

2. **Not only asymptotic “worst case”-complexity:**

Complexity-theory provides mostly asymptotic results.

Complexity-theory deals mostly with “worst-case”-complexity.

It is nice to know how the security acts when the security-parameter  $l$  (generalization of the key-length; applicatory useful) is sufficiently big. But in the practical application of cryptographic systems, the values of all parameters must be fixed. So the interesting question is: How secure is the system for specific, fixed values of the parameters? Or asked differently: Where does “sufficiently big” start?

The “worst-case”-complexity is insufficient for security, and so is the “average-case”-complexity (For there should not be just a few keys that can not be broken, but most keys should not be breakable).

One wishes: The problem must be difficult almost everywhere, i.e., it may only be not difficult in an infinitesimal fraction of all cases (i.e., if an honest user chooses his key randomly, the likelihood that the key can not be broken must be overwhelmingly big).

If the security-parameter is  $l$ , then one demands:

If  $l \rightarrow \infty$ , then likelihood of breaking  $\rightarrow 0$ .

One hopes: The likelihood of breaking decreases rapidly, so that  $l$  does not need to be overly big.

### 3 Cryptography Basics

3. **How does one define easy and difficult?** Essentially, two complexity-classes are needed in cryptographic systems: The things that the normal users must do must be easy, whereas the breaking must be difficult. Formally, this is mostly represented by the difference between polynomial and non-polynomial, thus:

key generation, en- and decoding: easy = polynomial in  $l$   
breaking: difficult = non-polynomial in  $l \approx$  exponential in  $l$

Why this special separation? (articles b and c do also apply to general complexity-theory)

- a) More difficult than exponential is not possible, see 1).
- b) Closure: Inserting polynomials into polynomials produces polynomials, i.e., reasonable combinations of polynomial algorithms are polynomial too (without needing to prove it).
- c) Reasonable computation models (Turing-, RAM-machine) are polynomially equivalent, so one does not need to appoint a specific machine model.

A polynomial of high order would suffice as a lower bound for the attacker-algorithm in practical use (and the simple algorithms are for the most part only up to  $l^3$  or less).

*Remark to:* polynomial in  $l \approx$  exponential in 1

Here stands no equal sign as there exist more functions between polynomial ( $l^k$  with fixed  $k$ ) and exponential ( $2^l$ ), e.g.,  $2^{\sqrt{l}}$ .

4. **Why are complexity-theoretical assumptions necessary?** (e.g., “factorization is difficult” (see §3.4.1))

Until now, the complexity-theory can not prove any usable lower boundaries.

For experts: More exactly, the breaking of many of such systems obviously lies in NP (in principle, NP means here: If a solution has been guessed, one can rapidly test if it is correct. It has been like this e.g., in 1) with the signature that was to be faked). Therefore, it is obvious in such cases, that proof that the breaking of such a system is exponential, would imply P≠NP. But as many scientists have tried to prove or refute P≠NP for many years, we can not expect such proof in the near future.

Inside of NP, no appropriate lower boundaries are known, so it would also be of no use here to be content with a lower boundary of  $l^{100}$ , for example.

5. **What kind of assumptions are made willingly?**

The more compactly and the longer (and by more people) examined, the more trustworthy. Thus in certain respects, cryptographic systems that are not information-theoretical secure either break or become more trustworthy as they age. Nevertheless one needs to gradually increase the parameters  $l$  because of the advancements

in computer performance. (Be careful when things need to remain secret for a long time, or if signatures need to remain valid i.e., fake-proof for years!)

Of course, an NP-complete problem would be especially nice, but no such problem – as difficult as in the sense of 2) – has been found yet.

## 6. What, if the assumption turns out to be wrong?

- a) Make other assumptions.
- b) More detailed analysis, i.e., appoint a specific computation-model (RAM- or Turing-machine? How many tapes with the Turing-machine? etcetera) and then analyze if the order of the polynomial is high enough.

## 7. Proof-intention:

If the attacker-algorithm can break the cryptographic system, then it can also solve the problem that is accepted as difficult.

### 3.1.3.5 Overview of the cryptographic systems presented in the following

About the crossed-out fields in Figure 3.12: Such systems do not exist with the specifications from Figures 3.4 and 3.7, as justified in article 1) above (Actually there now exists such a system, which nearly has the characteristics of digital signatures and is information-theoretical secure. But in means of efficiency, it presently seems to not be usable. Compare §3.9.3 and [ChRo\_91]).

Since 1998, field 3 is not entirely empty any more. Until then, there was no system known that was practical against active attacks, and there was no system against adaptive active attacks known at all [BIFM\_88, NaYu\_90, Damg\_92]. But the – in 1998 published by Ronald Cramer and Victor Shoup – system CS [CrSh\_98] has not been proven equivalent to a true standard-problem like the factorization of large numbers (see §3.4.1) or the computation of discrete logarithms (see §3.9.1). Its security is only as strong as the Diffie-Hellman-decision-problem is difficult. Though this assumption is also not unusual, it is a stronger assumption than the Diffie-Hellman-assumption which itself is a stronger assumption than the discrete-logarithm-assumption. For these reasons, and because systems which are based on the discrete logarithm are not a main-topic in this lecture, CS will not be explained in further detail in the following, but a system which is based on the factorization-assumption will be described by introducing the  $s^2\text{-mod-}n$ -generator.

Fields 10 and 11 are empty because no such systems are known. But in contrast to fields 1 and 2, an invention is not impossible.

About the subset-signs:

Horizontally: Every asymmetric system can also be used as a symmetric system if the creator of the secret key communicates it to his partner.

Vertically: Every information-theoretical secure system is also cryptographically strong, and the latter are automatically well-researched. Also every system that is secure against active attackers is certainly secure against passive attackers (But pay attention: This does not mean that *every* single cryptographically secure system must

### 3 Cryptography Basics

be better than every single well-researched system; i.e., one could imagine that the  $s^2$ -mod- $n$ -generator can be broken and that DES can not be broken because it is based on a completely different assumption.).

That explains the remaining empty fields: There one can use the system from the field to the right, or the system from the field above, e.g., GMR for 4, 6, 7, 9 and 11 or authentication-codes for 4, 6 and 9. Indeed there are sometimes more efficient system for weaker requirements. In fact, this is the only reason why it might pay off to use, for example, a pseudo-one-time pad instead of a real one-time pad or RSA as a signature system instead of GMR, or DES instead of an authentication-code.

Cryptographic systems which are not even well-researched were not included here. The term is a bit controversial of course. In particular, those systems for which already several similar systems were broken, are not considered well-researched here (Nevertheless, some people use nonlinear shifting-registers for symmetric encryption as these are even faster than DES).

Systems which were not even published, but which were only analyzed by their creator and his “intimates”, can be trusted even less.

#### 3.1.4 Hybrid cryptographic systems

Because symmetric cryptographic systems can be implemented between two and three times more efficiently than asymmetric systems in hardware as well as in software, asymmetric encryption systems (and if necessary digital signature systems) are mostly used only to submit a confidential (and if necessary digitally signed) key for symmetric cryptographic systems. With this submitted key, the actual data can be encrypted and/or authenticated much more efficiently i.e., much faster. This cryptographic system, which consists of an asymmetric encryption system, if necessary of a digital signature system and of a symmetric cryptographic system, is called a **hybrid cryptographic system**. It combines the comfortable key-management of the former cryptographic systems and the efficiency of the latter. This hybrid cryptographic system is of course only as strong as the weakest system that was used to build the hybrid.

A hybrid cryptographic system may not only be used because of its higher efficiency, but also because it uses the transfer-channel more efficiently: long blocks could create too much overhead; the error-propagation of the symmetric cryptographic system could be more adjusted to the application and to the channel (see §3.8.2)(The latter is only reasonable when authenticity is not important for the application.).

## 3.2 One-Time Pad

This section deals with encryption systems that are symmetric and information-theoretically secure i.e., with those systems that consist of the components represented in Figure 3.2. We will see that the simplest example, the one-time pad, is already perfect and there is no need for anything else.

System type		Concealment	Authentikation	
Security level		sym. asym.	sym. asym.	sym. asym.
information theoretical		Vernam-Chiffre (one-time pad)	1	2
cryptographically strong against... active attack	passive attack	Pseudo-one-time-pad with $s^2\text{-mod-}n$ -Generator	3 CS	4 GMR
well researched	mathematical chaos	5	System with $s^2\text{-mod-}n$ -Generator	6 7
		8	RSA	9 RSA
		DES	10	DES 11

X system is impossible        dominated by the known system

Figure 3.12: Overview of the following cryptographic systems

Security here means, roughly speaking, that an attacker who sees a ciphertext  $S$  (of random length), cannot deduce anything about the plaintext. This is achieved by requiring that there is at least one key  $k$  for every possible plaintext  $x$ , such that  $k(x) = S$ . (That means:  $x$  is actually a possible content of  $S$ , provided the secret key is  $k$ .) In order for the different plaintexts to have the same probability, it is required, in the most common case in which all keys have the same probability, that for each  $x$  there must be equally many such  $k$ , usually exactly 1.

Through the example of the one-time pad, one sees that this definition is very natural.  
*example:* We assume that we want to encode 2 bits and have 2 key bits. The encoding function is addition  $(\text{mod } 2)$  (= XOR bit by bit). Now the attacker e.g., sees the ciphertext  $S=01$ . In this case it could have been either the plaintext  $x = 00$  with the key  $k = 01$ , or  $x = 01$  and  $k = 00$ , or  $x = 10$  and  $k = 11$ , or  $x = 11$  and  $k = 10$ , c.p.

### 3 Cryptography Basics

Figure 3.13. Thus every plaintext is just as likely as every other plaintext.

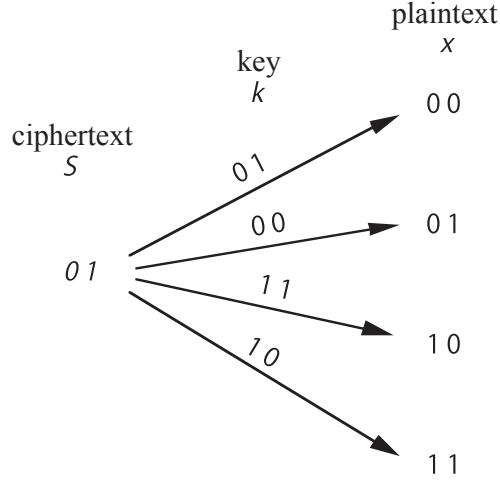


Figure 3.13: ciphertext can correspond to all possible plaintexts

This was the binary one-time pad for 2 bits. One can extend<sup>15</sup> this to a random amount of bits, and to other kinds of addition instead of addition ( $\text{mod } 2$ ). (for example ( $\text{mod } 26$ ) for letter-wise encoding by hand.)

The left side of Figure 3.14 illustrates the issue shown in Figure 3.13, representing all possible ciphertexts, omitting the key values though, for lack of space.

The right side of Figure 3.14 represents an insecure cipher. This can be seen because not every ciphertext can be mapped from every plaintext. An attacker who e.g., sees the ciphertext 10, thus knows that e.g., the plaintext 11 is not transferred. (of course even the insecure cipher in Figure 3.14 is better than nothing. If a 1-bit-keysection is only used once in each case for en- resp. decoding of a non-redundantly coded<sup>16</sup> 2-bit-textsection, the attacker will never find out, precisely which plaintext was transferred, no matter for how long and what he calculates. In this *limited* sense, even this cipher is information-theoretically secure: The attacker is able to infer information about the plaintext, but not enough to determine it exactly. In his revolutionary work about

<sup>15</sup>Remember the warning from §3.1: perfect secrecy (and in a certain sense the Vernam cipher is the epitome of secrecy) should be rather independent of the probability distribution of the plaintext domain and of the (source) encoding *which is selected for the plaintext*. Now, the binary Vernam cipher shows very strikingly that this is not the case in a terrifying way: if the plaintext messages are coded *unary*, then the *binary* Vernam cipher is totally ineffective. Therefore, in addition to the description of the Vernam cipher, the message length should be specified a-priori (more exactly: the ciphertext length), so that it cannot leak anything about the contents of the message. But then the problem remains that if too short a length is selected, long messages are divided into several short ones and then the number of messages could tell the observer something about their contents. With a sufficiently large chosen ciphertext length, this problem should really be negligible...

<sup>16</sup>If the 2-bit-text sequence contains sufficient redundancy, then the insecure cipher can be broken completely. Given that the plaintexts 00 and 10 do not appear we know that the plaintext 11 is always encoded the ciphertext 00, 01 corresponds to 01, 10 to 01 and 11 to 11.

information-theoretically secure secrecy [Sha1\_49], Claude Shannon calls this limited information-theoretical security *ideal secrecy* - a very unfortunate choice of term. Claude Shannon refers to information-theoretical security, as used and defined in this work, as *perfect secrecy*. In newer literature this kind of perfect secrecy is called *unconditional secrecy*.)

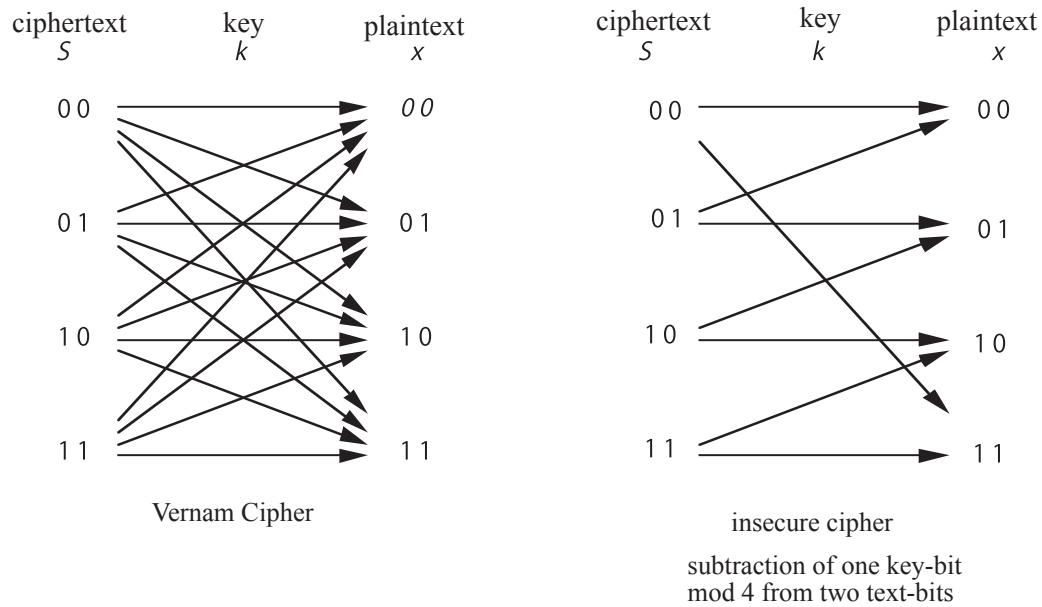


Figure 3.14: Every ciphertext should correspond to every possible plaintext

### Vernam cipher in general:

If someone wants to use the Vernam cipher over a group  $G$  to encode a character string of the length  $n$  (of characters from  $G$ ), then one has to choose randomly and exchange confidentially<sup>17</sup> a character string of the length  $n$  as key in advance, e.g.,  $k = (k_1, \dots, k_n)$ . The  $i^{\text{th}}$  plaintext character  $x_i$  coded as

$$S_i := x_i + k_i$$

It can be decoded by

$$x_i := S_i - k_i.$$

<sup>17</sup>To directly answer the question "what good is the Vernam cipher anyway if in order to be able to exchange a message confidentially I have to exchange a key of the same length as the to-be transmitted message? I might as well just transmit the message confidentially?": There is no benefit in the Vernam cipher concerning the length of the material that has to be transmitted secretly, but it can be exchanged at any time, i.e., whenever it suits both communication partners, long before the message has to be exchanged confidentially. In general, there will be no message to transmit just when the time would be good to do so. Thus the favorable situation for confidential communication "in advance", i.e. key exchange, is used.

### 3 Cryptography Basics

For the security proof see exercise 3-10 b). (Obviously one can regard each plaintext character individually, because they are all coded independently.) It is also plain to see that this cipher is safe against active attacks (see 3-10 ): Even if the attacker receives some pairs of plaintext – ciphertext of his choice, he learns only something about the key bits used there, which later (when he really wants to decode something) will never occur again<sup>18</sup>.

In regards to computation, Vernam cipher is simple and fast, and it causes no message expansion. Its only disadvantage is the length of the key: It is just as long as the message. Therefore Vernam cipher is impractical for many applications (however, with increasing availability of storage media it becomes more and more doable, at least in cases where the participants can e. g., exchange disks in advance).

One can easily understand that for information-theoretical security the keys must be as long as described in the following: Let  $\mathcal{K}$  be the set of keys,  $\mathcal{X}$  the set of plaintexts, and  $\mathcal{S}$  the set of ciphertexts that appear at least once. First  $|\mathcal{S}| \geq |\mathcal{X}|$  has to apply, because with a fixed key all plaintexts must be encoded differently in order to be decodable. Now we require of security that for all  $s$  and  $x$  there is a  $k$  with  $k(x) = s$ . With a fixed plaintext  $x$  for each ciphertext  $s$ , the key  $k$  must be different. Thus we get  $|\mathcal{K}| \geq |\mathcal{S}|$ , and overall  $|\mathcal{K}| \geq |\mathcal{X}|$ .

This shows us that at least as many keys as plaintexts must exist. If the plaintexts are coded reasonably<sup>19</sup>, the keys must be of the same length.

*Remark 1:* In the terms just introduced, the definition for information-theoretical security in the case of equally distributed keys reads:

$$\forall s \in \mathcal{S} \exists const \in \mathbb{N} \forall x \in \mathcal{X} : |\{k \in \mathcal{K} | k(x) = s\}| = const. \quad (3.1)$$

Let us discuss this definition a bit more in detail, as it is vital in this context:

From  $\mathbb{N} = \{1, 2, 3, 4, \dots\}$  follows:  $const \geq 1$

The constant  $const$  may depend on the concrete ciphertext  $s$  in the above definition — however in many information-theoretically secure encryption systems, e. g., the one-time pad, it does not. There, as was mentioned previously,  $const = 1$ . Therefore one may be tempted to change the order of quantors in the definition:

$$\exists const \in \mathbb{N} \forall s \in \mathcal{S} \forall x \in \mathcal{X} : |\{k \in \mathcal{K} | k(x) = s\}| = const$$

But the definition becomes more restrictive through this, without those systems that are satisfying it, becoming more secure. Therefore it is wise to stick with the weaker

---

<sup>18</sup>“again” refers to the concrete key bits, not their values. Of course every bit can and has to occur in a randomly selected, infinite bit sequence again and again if only its value is regarded as the characteristic. The same applies not only to individual bits, but also to each finite bit string.

<sup>19</sup>In order to prevent a misunderstanding: Here “encoding” does not have anything to do with privacy or authentication, it only means that with bit strings of the length  $l$  overall  $2^l$  values can be differentiated, that means  $2^l$  of plaintexts can be skillfully coded — or said in a different word: represented. Appropriate applies to the values of the keys, so that the appropriate length of the (encoding of) keys results.

definition. For the same reasons one does not use the following even more simplified definition:

$$\forall s \in \mathcal{S} \quad \forall x \in \mathcal{X} : |\{k \in \mathcal{K} | k(x) = s\}| = 1$$

What follows are some – admittedly rather exotic – encryption systems, which satisfy the general definition (1), but not the narrower one:

Example for  $const = 2$ : We extend the set of keys of the one-time pad by a bit, which is not used at all during the en- and decoding. Then each plaintext can be assigned to each ciphertext in each case by 2 keys, which differ exactly in the unused bit.

Example of  $const$  dependent on  $s$ : We double the key length of the one-time pad but ignore every second key bit while en- and decoding. Then each ciphertext of the length  $l$  can be assigned to each plaintext through  $2^l$  keys, which differ exactly in the unused bits. Thus  $const$  depends on the ciphertext, more exactly: its length. Then this applies:  $const = 2^l$ .

*Remark 2:* Security being ‘information-theoretical’, or ‘absolute’, refers to the fact that in its discussion we did not argue about how difficult something might be to calculate: We assumed the worst case i.e., we assumed the attacker has an exact table that tells him which  $k$ ,  $x$ , and  $s$  belong together. The encryption systems with short keys discussed later on will not be safe in this sense; there are for example many asymmetrical encryption systems where there is for each  $c$  just one exact possible  $d$  (see Figure 3.4), and therefore there is for every ciphertext  $s$  just one exact possible plaintext  $x$ , because the attacker knows  $c$ . These systems rely completely on the assumption that  $x$  is very hard to calculate.

*Remark 3:* The definition of information-theoretically secure encryption systems which is commonly used, [Sha1\_49], strikes out a bit further, but ours can be proved to be equivalent. (And in concrete proofs usually other authors use the simpler formulation above (3.1)) What it says is: No matter what information the attacker has a-priori about the plaintext, he cannot gain any additional information about it from the ciphertext.

The a-priori-information is represented by a probability distribution of plaintexts (e.g., in letter openings  $W(\text{'Dear'}) = 0.4$ ,  $W(\text{'Hello'}) = 0.4$ ,  $W(xrq, q) = 0.0001$ ). Also, certain keys have certain probabilities and are chosen independent of the plaintext. Therefore certain ciphertexts also occur with certain probabilities. The a-posteriori-probability of a plaintext  $x$ , if the attacker has seen the ciphertext  $s$ , is the conditioned probability of  $x$ , if one knows  $s$ ,  $W(x|s)$ . The security requirement is then

$$\forall s \in \mathcal{S} \quad \forall x \in \mathcal{X} : W(x|s) = W(x). \quad (3.2)$$

This means that for someone who does not know the key, plaintext  $x$  and ciphertext  $s$  are stochastically independent. To the equivalence of the definitions 3.1 and 3.2:

$$W(x|s) = \frac{W(x) * W(s|x)}{W(s)}$$

### 3 Cryptography Basics

[3.2](#) is equivalent to<sup>20</sup>

$$\forall s \in \mathcal{S} \quad \forall x \in \mathcal{X} : W(s|x) = W(s). \quad (3.3)$$

We show that this is equivalent to

$$\forall s \in \mathcal{S} \quad const' \in \mathbb{R} \quad \forall x \in \mathcal{X} : W(s|x) = const'. \quad (3.4)$$

[\(3.4\)](#)  $\Rightarrow$  [\(3.2\)](#) is obviously with  $const' := W(s)$ . Inversely we show  $const' = W(s)$ :

$$W(s) = \sum_x W(x) * W(s|x) = \sum_x W(x) * const' = const' * \sum_x W(x) = const'.$$

[\(3.2\)](#) already looks very similar to [\(3.1\)](#): In general  $W(s|x) = W(\{k|k(x) = s\})$ , and when all keys are equally probable,  $W(s|x) = |\{k|k(x) = s\}|/|\mathcal{K}|$ . Then [\(3.2\)](#) is equivalent to [\(3.1\)](#) with  $const = const' * |\mathcal{K}|$ .

## 3.3 Authentication Codes

Authentication codes are symmetric systems of authentication that are information-theoretically secure. They consist of the components shown in Figure [3.6](#). Before we give an example (Figure [3.15](#)), we will discuss informally what is meant by information-theoretical – thus absolute – security in this case.

Security here means that the receiver of a message can tell by the (invalid) MAC, whether it is a genuine message of the sender or forged by an attacker.

Forgery of course cannot be detected with a probability of 1: There must be a correct MAC for the forged message, and the attacker can always be lucky and guess the right one. More precisely, if the domain for MACs has cardinality  $W$  one has a  $\frac{1}{W}$  chance of just guessing the correct MAC. (In particular with  $\sigma$ -bit MACs with a probability of  $2^{-\sigma}$ .)

Thus the best security we can hope for is achieved when an attacker can do nothing better than randomly guessing.

*Remark:* Note however that the attacker cannot locally check whether or not his guess was right, i.e., this is different from the situation in [§3.1.3.4](#). Here we will prove that it is almost always possible to detect a forged message. This is in contrast to a situation where an attacker with unlimited computation power could, with a digital system like in Fig. [3.7](#), produce many such signatures that look perfectly valid because he knows the test that they have to pass. (see however [§3.9.2](#) and [§3.9.3](#))

Thus authentication codes provide better safety properties than digital systems, in principle, but worse usability in practice. (see [§3.1.1.2](#))

### Example 1:

Before we formalize the safety statement, we will look at Fig. [3.15](#), which represents a small authentication code (as seen by the attacker). Here 1 message bit  $b \in \{H, T\}$  (short

---

<sup>20</sup>We regard only those ciphertexts  $s$  that can occur, so that means:  $W(s) > 0$

for Head, Tail) is to be authenticated; the key length is 2 bits, and the MAC is 1 bit. As it is the case for all following systems, all keys are selected with the same probability.

The table head lists the possible results; the table body shows which plaintext yields what MAC. (This code can be represented much shorter, see ex. 3-8 b)) As with the Vernam cipher, each key section (here in each case the two bits) is only used once.

		text with MAC				
		H,0	H,1	T,0	T,1	
		00	H	-	T	-
key	01	H	-	-	T	
		10	-	H	T	-
		11	-	H	-	T

Figure 3.15: A simple authentication code

The MAC-length tells us that falsification can be detected with a probability of at most  $1/2$  in this example. Now we want to show that it actually is detected with a probability of  $1/2$ .

We regard only the case where the attacker wants to forge the message  $T$ . If the correct sender has not yet sent anything, all 4 keys are equally probable; for two of those,  $T$  has 0 as MAC resp. 1. Thus the attacker selects the wrong value with a probability of  $1/2$ .

However, the interesting thing about an authentication code is that security is still valid if the correct sender sends the message  $H$  with the correct MAC, and the attacker intercepts it and tries to replace it with  $T$ .

case 1: The attacker intercepts  $H,0$ . Now  $k = 00$  and  $k = 01$  are still possible keys. The attacker would have to select 0 as the MAC for  $T$  if the key is  $k = 00$ , but 1 if it is  $k = 01$ . Again the attacker's guess is wrong with probability of  $1/2$ .

case 2: The attacker intercepts  $H, 1$ . Similarly... (for more see ex. 3-8 b)).

#### A more precise definition of security:

A more precise definition of when an authentication code is secure with error probability  $\varepsilon$ , describes exactly what we did in the example above:

"Even if an attacker sees a correct message  $(x, \text{MAC})$ , eliminates it and tries to send the plaintext  $y$  instead, no matter how skillfully he chooses  $\text{MAC}'$ , the falsification will

### 3 Cryptography Basics

be detected with a probability  $\geq 1 - \varepsilon$ , i.e., the probability for  $MAC' = \text{code}(k, y)$  is at most  $\varepsilon$  (where  $k$  is the correct secret key)."

We now just have to indicate precisely, for what set the probability is actually calculated. This is exactly the set of those keys that are still possible from the attacker's view (in the above example, in case 1, the set consisting of 00 and 01). The attacker's view implies that only those keys  $k$  are applicable, for which  $MAC = \text{code}(k, x)$  holds.

To be completely formal: An authentication code  $\text{code}$  with key set  $\mathcal{K}$  (and probability distribution  $W$  to  $\mathcal{K}$ ) is called secure with error probability  $\varepsilon$ , if the following holds:  $\forall x \forall MAC \in \text{code}(\mathcal{K}, x) \forall y \neq x \forall MAC'$ :

$$W(MAC') = \text{code}(k, y) | MAC = \text{code}(k, x)) \leq \varepsilon.$$

#### **Example 2 (Improvement of the code of Fig. 3.15):**

In order to reduce the error probability of the code from Fig. 3.15 from  $\frac{1}{2}$  to  $(\frac{1}{2})^\sigma$ , one can attach  $\sigma$  MACs to one message bit as before, i.e.,

$$x, MAC_1, \dots, MAC_\sigma.$$

For this,  $2 \times \sigma$  key bits have to have been exchanged. The attacker guesses every individual MAC correctly with probability  $\frac{1}{2}$ . Since they are stochastically independent from each other, all MACs are guessed correctly with a probability of only  $(\frac{1}{2})^\sigma$ .

If one wants to use the code for messages of length  $l$  bits, one needs  $l * 2\sigma$  key bits, and each bit is authenticated individually in the above sense. (The total probability of errors changes somewhat!) A substantially more efficient code is discussed in task 3-8 f).

#### **Outlook: Limitations of authentication codes.**

Just like information-theoretically secure encryption systems, authentication codes need keys of a certain minimum length.

We have almost shown that for an error probability  $\varepsilon$  a key set containing at least  $\lceil 1/\varepsilon \rceil$  elements is needed: It was clear to begin with that the number of possible MACs for the message  $y$  to be counterfeited must be  $\lceil 1/\varepsilon \rceil$ . In addition, for every possible MAC there must be a possible key.

It is also evident that there must be even more keys, because on the basis of the correct message  $(x, MAC)$  the attacker could rule out many keys. It can be shown (easily) that there have to be  $\lceil 1/\varepsilon^2 \rceil$  keys [GiMS\_74].

If one chooses  $\varepsilon = 2^{-\sigma}$ , then one needs keys of at least length  $2\sigma$ . This is not as bad as the lower barrier of encryption systems: There the key has to be as long as the message. But in practice, a value of approximately  $2^{-100}$  would be sufficient for the error probability. (Otherwise the probability that the computer which does the MAC authentication will unnoticed fail and let falsified messages pass as valid, etc., or all risks of practical life in general become greater than the one risk of a single message being forged successfully.)

Therefore the code from Fig. 3.15 is optimal for 1-bit-messages, and so is its first extension from example 2. The key length does not necessarily have to grow with the

length of the message as in the 2<sup>nd</sup> extension of example 2 though. With the substantially more efficient code from task 3-8 f) it is possible e.g., to authenticate messages of length 100 with 200 key bits and an error probability of  $2^{-100}$ . In [WeCa\_81, §3] this code has been extended in such a way that one is able to authenticate a message of length  $l$  with an error probability of  $2^{-\sigma+1}$  with a key of a length about  $4\sigma ld(l)$ . That is not quite optimal, but practicable.

When authenticating  $n$  messages consecutively with one key and aiming for an error probability of  $\leq 2^{-\sigma}$ , at least  $\sigma$  key bits per message are needed — less than  $2\sigma$  bits per message as before: There is a procedure which functions with  $(n + 2)\sigma$  bits for  $n$  short messages [WeCa\_81, §4].

The idea is that for each one of the short messages, a secret  $MAC$  is computed with the same  $2\sigma$  key bits every time. These are never put out in plaintext (otherwise an attacker could falsify arbitrary messages after having observed 2 MACs), but encrypted using a one-time pad. For the encryption of the  $n$  MACs with the one-time pad,  $n\sigma$ -bit keys are needed, thus  $(n + 2)\sigma$  bits altogether.

## 3.4 The $s^2 - \text{mod} - n$ -pseudo-random-bitsequence-generator: cryptographically strong secrecy

What we have learned so far: First — for information-theoretically secure secrecy only symmetrical methods can be used, which means one has problems with key distribution. Second — the keys must be at least as long as the message. Therefore one is almost always contents with cryptographic security in practice. (red telephones between Moscow and Washington allegedly use the Vernam cipher.)

In this section we regard the second best safety level, i.e., cryptographically strong symmetric and asymmetric secrecy. However, the asymmetric system has the disadvantage that it is definitely insecure against active attacks. In many cases that are relevant in practice, one would therefore prefer RSA although it is not even cryptographically strong (§3.6).

The symmetric and the asymmetric encryption systems are based on the same cryptographically strong PRNG. Cryptographically strong PRNG are of interest in other contexts too, because the generation of genuine random numbers is in practice a considerable problem.

Furthermore the mathematical and cryptographic bases for this system are the same ones needed for GMR and RSA, since all these systems are based on the assumption of factorizing.

### 3.4.1 Basics for systems with factorizing assumptions

In every asymmetric cryptographic system a secret (a decoding or signature key) must be generated, about which some information (the encoding or test key) can be made

### 3 Cryptography Basics

public. This information must make it possible on the one hand to examine what is done using the secret, but on the other hand it may not permit the deduction of the secret itself with reasonable effort. In 1976, when [DiHe\_76] was published, the mere idea of such things being possible appeared to be rather sensational.

Nearly all asymmetric systems that are seriously discussed at present are based on only two different ideas for such secrets. We will focus on the ones that are based on the factorizing assumption. (The other one is called discrete logarithm, see §3.9.1, and is based on similar elementary number theory.)

#### Idea for the secrets:

The secret essentially consists of two large, independently selected, random

prime numbers  $p$  and  $q$ .

('Large' here means at present circa 300 to 1000 bits long.) The public information is essentially the product

$$n = p * q.$$

So that it produces something meaningful, one also has to be able to:

1. select large prime numbers in a reasonable time (multiplying them can be done anyway),
2. but it may not be possible to factorize such products with feasible effort.
3. Moreover it is not enough, of course, to have a secret, but there must be functions over messages, e. g., for decoding or signing, which can only be computed efficiently with knowledge of  $p$  and  $q$ , while  $n$  is sufficient for the reverse operations.

For some of the systems, a few special characteristics of  $p$  and  $q$  are required, e. g., in the one from §3.4 we must have

$$p \equiv q \equiv 3 \pmod{4}.$$

#### The Factoring Assumption:

As laid out in §3.1.3.4, it has not yet been proven that factoring is difficult. We therefore just have to rely on the assumption that for every fast (factoring) algorithm  $\mathcal{F}$  the probability that  $\mathcal{F}$  is actually able to factorize a number of the form  $p * q$ , quickly decreases with increasing length  $l$  of the factors  $p$  and  $q$ .

$l$  is then called the safety parameter. It is easy to see that worst- or average-case complexity would not be sufficient: Even if  $\mathcal{F}$ , for example, could only efficiently factorize every 1000<sup>th</sup> number of any given length  $l$ , this would mean that it could find out the secrets for  $\frac{1}{1000}$  of the participants. Put more precisely, it is therefore required that the remaining probability decrease faster than 1 divided by any polynomial.

### 3.4 The $s^2 - \text{mod} - n$ -pseudo-random-bitsequence-generator

*Assumption:* For every (probabilistic) polynomial algorithm  $\mathcal{F}$  and every polynomial  $Q$ , the following is true: There exists an  $L$ , such that for all  $l \geq L$  the following holds: If  $p, q$  are independently chosen random prime numbers of length  $l$  and  $n := p * q$ , then

$$W(\mathcal{F}(n) = (p; q)) \leq \frac{1}{Q(l)}.$$

This is the assumption with which one works in proofs; in practice, the definition for the assumption for concrete systems, of course, is more like: There is no algorithm  $\mathcal{F}$  that factorizes more than a certain proportion (e.g.,  $2^{-100}$ ) of numbers  $n$  of the concrete length  $l$  (e.g., 1024) used in the given system, in less than a hundred years, even on the fastest available computer.

While no "fast" algorithms are known that factorize all numbers  $n$ , there are known fast algorithms, for the special case where  $p+1, p-1, q+1$  or  $q-1$  only have small prime factors [DaPr\_89, p.232].

In order to avoid this, one demands:

- $p + 1$  must have (at least) a large prime factor.
- $p - 1$  must have (at least) a large prime factor.
- $q + 1$  must have (at least) a large prime factor.
- $q - 1$  must have (at least) a large prime factor.

Here the large prime factors should each be at least 100 bits long [BrDL\_91, p.268].

For very large  $l$  (and therefore asymptotically),  $p$  and  $q$  will satisfy these requirements "automatically" with very large probability. For practical use, this is sufficiently probable for prime numbers which are substantially longer than 500 bits [BrDL\_91, p.268].

A short account of the state of factorizing is given here for a concrete hint for the dimensioning of cryptographic systems that are based on the factorizing assumption. By means of

- single "supercomputers" (array processor),
- one hundred workstations connected by a local area network and
- world-wide distributed UNIX systems connected by electronic mail, whose spare time was used,

numbers with 100 to 130 decimal places could be factorized in 1989 within a month [LeMa1\_89, LeMa1\_90, Silv\_91, Schn\_96, p.257].

In 1999 numbers with 155 decimal places could be factorized [Riel\_99].

According to [Schn\_96, p.256], the *quadratic sieve*, which is the factorizing algorithm used to break the factorizing record in 1989, is the fastest known algorithm for numbers  $n$  of up to 110 decimal places. Its run time is

$$L_{\text{quadratic sieve}}(n) = \exp(\sqrt{\ln(n) \ln \ln(n)}).$$

### 3 Cryptography Basics

The run time of factoring algorithms is, as usual, given in time units of one multiplication of two numbers of half the length of  $n$ , for example for the computation of  $n = p * q$ .

For numbers  $n$  with more than 110 decimal places<sup>21</sup>, the factoring algorithm *general number field sieve* which has been in use only since 1993, is faster and the fastest known so far. Its run time is

$$L_{\text{general number field sieve}}(n) = \exp(1.923 * \sqrt[3]{\ln(n)(\ln(\ln(n)))^2}).$$

For factoring numbers of length of approximately 130 decimal places, it takes about twice as much time for 4 additional decimal places using the algorithm *quadratic sieve* and 5 additional decimal places using *general numbers field sieve*. In practice,  $l$  should nowadays be selected within the range of lengths 350 to 2048 bits (see task 3-11 a) ) so that the number  $n$  that is supposed to be factorized, has a length between 700 and 4096 bits.

#### Generating prime numbers:

- First we note that even between very large numbers there are enough prime numbers to choose from: According to the theorem concerning prime numbers [Kran\_86, §1.15], the following is true for the number of the prime numbers  $\pi(x)$  that are less than or equal to a certain maximum  $x$ :

$$\frac{\pi(x)}{x} \approx \frac{1}{\ln(x)}.$$

When considering numbers up to a length of  $l$ -bit, then on average every  $(l * \ln(2))^{th}$  is a prime number.

- According to the Dirichlet theorem [Kran\_86, §1.15], on average half of all prime numbers are  $\equiv 3 \pmod{4}$  (and a quarter is  $\equiv 3$  resp.  $7 \pmod{8}$ ).

First, these two theorems tell us that we do not have to fear that someone factors all numbers  $n$  of the form  $p * q$  and a fixed length  $l$  by making a list of all applicable prime numbers. Second, they are needed for the prime number production.

- Fast random prime number generation is in principle as simple as generating prime numbers with additional characteristics, e. g.,  $p \equiv 3 \pmod{4}$ :

WHILE no prime number is found, yet  
Do select random number  $p$  of suitable size<sup>22</sup>

---

<sup>21</sup>In the case of equating the specified running times of *quadratic sieve* and *general number field sieve* and solving after  $\ln(n)$ , one gets:  $\ln(n) \approx 286$ . This corresponds to 124 decimal places. The difference to the 110 decimal places specified by Bruce Scheier may explain itself by the fact that with a *quadratic sieve*, a factor also stands in front of the root, which, since it lies quite near 1, was set to 1 in the formula of run-time.

(if necessary e.g., with  $p \equiv 3 \pmod{4}$ )  
 Test, whether  $p$  is a prime number  
 OD.

The algorithm will succeed after  $l * \ln(2)$  loops.

- Testing whether  $p$  is prime number cannot be done just by factoring  $p$  and thus testing whether it has any divisors except 1, because we just assumed that factoring is difficult.

Fortunately there are faster tests, although these have an exponentially small error probability. The best method is the RABIN-MILLER-Test. It is particularly simple, especially for numbers  $p$  that satisfy  $p \equiv 3 \pmod{4}$ . §3.4.1.5 yields: For every prime number  $p$  we have:

$$p \text{ prime} \Rightarrow \forall a \in \mathbb{Z}_p^*: a^{\frac{p-1}{2}} \equiv \pm 1 \pmod{p}$$

which, in turn, for a non-prime number  $p$  is only true for  $\frac{1}{4}$  of all  $a \in \mathbb{Z}$ . (Here  $\mathbb{Z}_p^* := \{1, 2, \dots, p-1\}$  for a prime number  $p$ . This notation will be introduced more in detail below.) One thus checks  $p$  against this formula for  $\sigma$  random values  $a$ . If it does not hold at least once, it is obvious that  $p$  is not prime. Otherwise one can say in a certain way that  $p$  is a prime number at least with probability  $1 - 4^{-\sigma}$ .

The error probability is irrelevant at least with respect to the factoring assumption, since there is a similar error probability anyways: the probability that  $\mathcal{F}$  has the ability to factor big numbers.

Before applying the Rabin-Miller-Test though, one checks in practice whether  $p$  contains very small prime factors. To do this, a table of the first prime numbers – e.g., of length  $l \leq$  the computer's word length – is made up (once and for all).  $p$  is then divided by all these prime numbers to see whether they are factors of  $p$ . If they are not, it is worth a try to check whether  $p$  also satisfies the equation above. (see, e.g., [Kran\_86](§1, §2))

### Calculating with and without knowledge of $p$ and $q$ :

Almost all operations on  $p$ ,  $q$ , and  $n$  are coset operations. Therefore we will first briefly cover the most important facts about operations in cosets. In particular, we will consider their respective computational complexity. (For more information see e.g., [Lips\_81, Lüne\_87, ScSe\_73, Knut\_97] or any other introduction to elementary number theory.)

#### 3.4.1.1 Computation modulo n

Here,  $n$  is some random number. The operations below can therefore be executed both by participants who know the secret and by participants who do not.

### 3 Cryptography Basics

- $\mathbb{Z}_n$  denotes the ring of residue classes ( $\text{mod } n$ ). It is usually represented by the numbers  $\{0, \dots, n-1\}$ ; calculating in this structure should be intuitively clear for most readers. Nevertheless the fundamental definitions will be revised briefly:  
The coset ring ( $\text{mod } n$ ) can more formally be defined by means of a congruence relation “ $\equiv$ ”:

Let  $a, b \in \mathbb{Z}$ . Then

$$a \equiv b \pmod{n} \Leftrightarrow n \mid (a - b).$$

(The sign ‘|’ stands for ‘divides’, i.e.,  $x \mid y \Leftrightarrow \exists z \in \mathbb{Z} : y = x * z$ .) We have:  $a \equiv b \pmod{n}$  is true if their remainders with resp. to division by  $n$  are equal.

Now, for every  $a \in \mathbb{Z}$ , a residue class is defined by  $\bar{a} := \{b \in \mathbb{Z} | a \equiv b \pmod{n}\}$ . It is possible to define meaningful operations with these classes, because

$$\begin{aligned} a_1 \equiv a_2 \pmod{n} \wedge b_1 \equiv b_2 \pmod{n} \Rightarrow \\ a_1 \pm b_1 \equiv a_2 \pm b_2 \pmod{n} \wedge a_1 * b_1 \equiv a_2 * b_2 \pmod{n} \end{aligned}$$

which can be seen quite easily. (That is, addition (or sim.) can be done with random elements of  $\bar{a}, \bar{b}$ ; the result will always be the same.) In general, elements of  $\{0, \dots, n-1\}$  are used as arguments of the operations and the result is usually represented by the remainder of division of their outcome by  $n$ . This is also the most usual representation in computers. For “pen-and-paper” computation usually the symmetric representation around zero is used, i.e., for odd numbers  $n$ ,  $\{-(n-1)/2, \dots, 0, \dots, (n-1)/2\}$  and for even numbers  $n$  – somewhat less symmetric –  $\{-n/(2+1), \dots, 0, \dots, n/2\}$ .

Most normal arithmetic rules apply. More precisely:  $\mathbb{Z}_n$  is a commutative ring with 1.

- Addition, subtraction, and multiplication in  $\mathbb{Z}_n$  is ‘easy’. Long binary numbers that span several computer words can in principle be treated the same way as regular numbers with several decimal places. (Let  $l$  be the length of  $n$ , then the complexity of addition and subtraction is  $O(l)$ , the complexity of multiplication is  $O(l^2)$  – it does not matter whether the calculation is carried out using regular integer arithmetic or arithmetic ( $\text{mod } n$ ) [Lips.81]. There are even faster algorithms, but for numbers of the order of magnitude common in cryptography, the difference is not yet big.)

Exponentiation  $a^b \pmod{n}$  with an exponent  $b$  of the same order of magnitude is ‘easy’<sup>23</sup> as well. It is often used (e.g., in the prime number test above). However,  $b$ -fold multiplication of  $a$  takes too long here. Therefore “square and multiply” algorithms are used, processing the exponent bit-per-bit; Let  $b = (b_{l-1} \dots b_0)_2$  be

---

<sup>23</sup>Hint: If all three parameters are of length 512 bits, exponentiation on software and computers available in 1997 would take about 0.1s. It depends of course on the exact performance of the computer and the quality of the implementation.

### 3.4 The $s^2 - \text{mod} - n$ -pseudo-random-bitsequence-generator

the binary representation of  $b$ . We will do the calculation going from left to right (though it works in both directions):

$a^{(b_{l-1})_2}, a^{(b_{l-1}b_{l-2})_2}$  etc. are calculated one after another. For every  $a^{(b_{l-1}b_{l-2}\dots b_{l-i})_2}$ , the next intermediate value is calculated as:

$$a^{(b_{l-1}b_{l-2}\dots b_{l-i}0)_2} = a^{2*(b_{l-1}b_{l-2}\dots b_{l-i})_2} = (a^{(b_{l-1}b_{l-2}\dots b_{l-i})_2})^2$$

resp.

$$a^{(b_{l-1}b_{l-2}\dots b_{l-i}1)_2} = a^{2*(b_{l-1}b_{l-2}\dots b_{l-i})_2+1} = (a^{(b_{l-1}b_{l-2}\dots b_{l-i})_2})^2 * a.$$

depending on the next bit of the exponent. Summing up, we square for every binary digit of the exponent and – if it is a 1 – we multiply by the base. each  $(\text{mod } n)$ .

Therefore the complexity of an exponentiation  $a^b \pmod{n}$  with an exponent  $b$  of the same order of magnitude as  $a$  is at most  $O(l^3)$ , because squaring (and multiplying where necessary) is done  $l$  times. For every bit the exponent  $b$  is shorter, it is done once less – leading zeros are just canceled out. Let  $|b|$  be the length of the exponent  $b$ , then the complexity of calculating  $a^b \pmod{n}$  is at most  $O(l^2 * |b|)$ .

example:  $7^{22} \pmod{11}$ . 22 is  $(10110)_2$ . We get:

$$\begin{aligned} 7^{(1)_2} &:= 7; \\ 7^{(10)_2} &:= 7^2 \equiv 5; \\ 7^{(101)_2} &:= 5^2 * 7 \equiv 3 * 7 \equiv -1; \\ 7^{(1011)_2} &:= (-1)^2 * 7 \equiv 7; \\ 7^{(10110)_2} &:= 7^2 \equiv 5. \end{aligned}$$

- $\mathbb{Z}_n^*$  describes the multiplicative group of  $\mathbb{Z}_n$ , i.e., the set of all elements that have an inverse. (It can be easily shown that this is really a group). The following holds

$$\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid \gcd(a, n) = 1\}.$$

The calculation of the inverse is 'easy' as well. To do this, the so-called extended algorithm of Euclid is used. Its output for two integers  $a, b \in \mathbb{Z}$  is their gcd and two integers  $u, v$  such that

$$u * a + v * b = \gcd(a, b)$$

The following example is to illustrate this algorithm: Let  $a = 11, b = 47$ . Next we will use the regular algorithm of Euclid, iteratively dividing the previous divisor by the previous remainder:

$$47 = 4 * 11 + 3;$$

$$11 = 3 * 3 + 2;$$

$$3 = 1 * 2 + 1.$$

### 3 Cryptography Basics

The extended algorithm now calculates the gcd, represented as a linear combination of divisor and dividend, starting with the last line:

$$\begin{aligned}
 1 &= 3 - 1 * 2 && \text{(Now substitution of 2 by 3 and 11)} \\
 &= 3 - 1 * (11 - 3 * 3) \\
 &= -11 + 4 * 3 && \text{(Now substitution of 3 by 11 and 47)} \\
 &= -11 + 4 * (47 - 4 * 11) \\
 &= 4 * 47 - 17 * 11 && \text{(sample: } 4 * 47 = 188, 17 * 11 = 187\text{)}
 \end{aligned}$$

(This can be optimized especially w.r.t. storage, s. literature.)

Getting back to inverting  $a$ , we calculate integers  $u, v$  with the extended algorithm of Euclid:

$$u * a + v * n = 1.$$

Then, obviously

$$u * a \equiv 1 \pmod{n}$$

and this just tells us that  $u$  is the inverse of  $a$ . In our example we get

$$11^{-1} \equiv -17 \equiv 30 \pmod{47}$$

(and  $47^{-1} \equiv 4 \pmod{11}$  as well as  $11^{-1} \equiv -17 \equiv 3 \pmod{4}$  and  $47^{-1} \equiv 4 \pmod{17}$ , too.)<sup>24</sup>

#### 3.4.1.2 The number of elements in $\mathbb{Z}_n^*$ and its relevance for exponentiation

The following are a few characteristics of  $\mathbb{Z}_n$  with which knowledge of the secret  $(p, q)$  helps:

- The  $\phi$  function by Euler is defined as

$$\phi(n) := |\{a \in \{0, \dots, n-1\} \mid \gcd(a, n) = 1\}|,$$

whereby we get for random integers  $n \neq 0 : \gcd(0, n) = |n|$ , see e.g., [Lüne\_87, p.12].

From these two definitions it follows immediately that

$$|\mathbb{Z}_n^*| = \phi(n)$$

Specifically for  $n = p * q$ ,  $p, q$  (being prime numbers), and  $p \neq q$ ,  $\phi(n)$  one can easily calculate:

$$\phi(p * q) = (p - 1)(q - 1). \quad ^{25}$$

---

<sup>24</sup>We also see that the equation  $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid \gcd(a, n) = 1\}$  shown above is correct: We just calculated the inverse  $n$  for each  $a$  s.t.  $\gcd(a, n) = 1$ . Conversely: If there is an inverse, thus  $a * a^{-1} \equiv 1 \pmod{n}$ ,  $n|(a * a^{-1} - 1)$ , there is a  $v$  s.t.  $a * a^{-1} - v * n = 1$ . If  $a$  and  $n$  had a common divisor  $d$ , then it would also divide 1.

<sup>25</sup>Those numbers that are not coprime to  $n$  (i.e., with  $\gcd \neq 1$ ) are  $0, p, 2p, \dots, (q-1)p$ , and  $q, 2q, \dots, (p-1)q$ . These  $1 + (q - 1) + (p - 1) = p + q - 1$  numbers all differ for  $p \neq q$ .

### 3.4 The $s^2 - \text{mod} - n$ -pseudo-random-bitsequence-generator

With given  $n (= p * q)$  calculating  $\phi(n)$  is as difficult as finding  $p$  and  $q$  [Hors\_85, S.187]:

$$\phi(p * q) = (p - 1)(q - 1) = p * q - (p + q) + 1 = n - (p + q) + 1 \text{ Summing up:}$$

$$p + q = n - \phi(p * q) + 1 \quad (**)$$

$$(p - q)^2 = p^2 - 2pq + q^2 - 4pq = (p + q)^2 - 4n$$

The square root yields:

$$p - q = \sqrt{(p + q)^2 - 4n}$$

Inserting  $(**)$  gives us:  $p - q = \sqrt{n - \phi(p * q) + 1}^2 - 4n$

Adding  $(**)$  yields  $2p = n - \phi(p * q) + 1 + \sqrt{n - \phi(p * q) + 1}^2 - 4n$

- For some cryptographic systems (e.g., RSA), the following implication of the number of elements for exponentiation is important: According to the small lemma by Fermat we have: For all  $a \in \mathbb{Z}_n^*$ :

$$a^{\phi(n)} \equiv 1 \pmod{n}.^{26}$$

#### 3.4.1.3 The relation between $\mathbb{Z}_n$ and $\mathbb{Z}_p$ , $\mathbb{Z}_q$

**( $p \neq q$ ,  $p, q$  being prime numbers,  $n := p * q$ ..)**

The relation between  $\mathbb{Z}_n$  and  $\mathbb{Z}_p$ ,  $\mathbb{Z}_q$  will become important later, making it possible for a participant to compute secret things first  $(\pmod{p})$  and then  $(\pmod{q})$  using the secret  $(p, q)$ , and then use the result in the context of  $(\pmod{n})$ , which is what everybody else works with.

The basis for this relation is the following formula (a special case of the “Chinese remainder theorem”):

$$a \equiv b \pmod{n} \Leftrightarrow a \equiv b \pmod{p} \wedge a \equiv b \pmod{q} \quad (*)$$

This can be seen easily: According to the definition of ‘ $\equiv$ ’ this is equivalent to

$$n|(a - b) \Leftrightarrow p|(a - b) \wedge q|(a - b),$$

and this is quite clear since  $p * q$  is the factorization of  $n$ .<sup>27</sup>

---

<sup>26</sup>This is a general theorem from the theory of groups [Lips\_81, Waer\_71]: It is valid for every element  $a$  in all finite groups  $G$ :  $a^{|G|} = 1$ . The proof is not that difficult. Both of the basics are:

1. it is valid for every subgroup  $H$ :  $|H||G|$ .
2. the powers of an element, thus  $a, a^2, \dots$  are forming a subgroup. It exists a first power  $a^\nu = 1$ , and just from there the sequence repeats itself cyclically.  $\nu$  is called ‘the order of  $a$ ’. This subgroup has thus exactly  $\nu$  elements, and from (1.) it follows  $\nu||G|$

<sup>27</sup>It is easy to see, that the Chinese remainder theorem is not only valid for  $n = \text{product of two different prime numbers}$ , but also for  $n = \text{product of as many as desired, in pairs divisor-strange natural numbers}$ .

### 3 Cryptography Basics

If, for some function  $f(x)$ , calculating  $y := f(x) \pmod{\pi}$  is only easy for prime numbers  $\pi$ , then, in order to get  $y := f(x) \pmod{m}$ , the owner of the key can first calculate  $y_p := f(x) \pmod{p}$  and  $y_q := f(x) \pmod{q}$

Then, the following holds:

$$y \equiv f(x) \pmod{n} \Leftrightarrow y \equiv y_p \pmod{p} \wedge y \equiv y_q \pmod{q}.$$

Now it would be great if we could efficiently get  $y$  from  $y_p$  and  $y_q$ . This is exactly what the “Chinese remainder algorithm (CRA)” does (the actual Chinese remainder theorem just states that such a  $y$  exists for random  $y_p, y_q$ ):

First one determines integers  $u, v$  using the extended Euclidean algorithm such that

$$up + vq = 1. \text{<sup>28</sup>}$$

Claim:

$$y = \text{CRA}(y_p, y_q) := u * p * y_q + v * q * y_p \pmod{n}.$$

So we have to show that  $y \equiv y_p \pmod{p} \wedge y \equiv y_q \pmod{q}$ . This can bee seen best in the following table:

	$\pmod{p}$	$\pmod{q}$
$up$	0	1
$vq$	1	0
$up \ vq + vq \ y_p$	$0 * y_q + 1 * v_p$ $\equiv y_p$	$1 * y_q + 0 * v_p$ $\equiv y_q$

(The upper two lines result from  $up + vq = 1$  and imply that  $up$  and  $vq$  in  $\mathbb{Z}_n$  are representing something like two “base vectors” if one compounds  $\mathbb{Z}_n$  from  $\mathbb{Z}_p$  and  $\mathbb{Z}_q$ . The computation of  $up$  and  $vq$  can happen once and for all with production of the secret.)

#### 3.4.1.4 Squares and roots in general

From §3.4.1.3 we know, how a secret  $p, q$  can help to calculate a function  $f \pmod{n}$ , that can only be computed efficiently modulo prime numbers. Extracting a root  $\pmod{n}$  will be such a function. It will turn out to be necessary to know some secret about  $n$  to efficiently extract roots  $\pmod{n}$ , while everyone is able to calculate the square  $\pmod{n}$  knowing only  $n$ , thus executing the inverse operation. As an introduction we will first take a closer look at roots and squares in coset rings.

- They are defined exactly as one would imagine, except that one worries mostly only about invertible elements:

$$\text{QR}_n := \{x \in \mathbb{Z}_n^* \mid \exists y \in \mathbb{Z}_n^* : y^2 \equiv x \pmod{n}\}.$$

The  $x$ s in this set are called quadratic residues (similar to square numbers in  $\mathbb{N}$ ), and an appropriate  $y$  is called root of  $x$ .

---

<sup>28</sup>Two different prime numbers have  $\gcd = 1$  of course. It is easy to see, that the CRA works too if  $p$  and  $q$  are not prime numbers, but furthermore divisor-strange. By reapplication of CRA, the Chinese remainder theorem can be covered for products that consist of more than two factors, too.

- $y$  is root of  $x \Rightarrow -y$  is also a root of  $x$  is valid as usual, because  $(-1)^2 = 1$ .
- Other laws, which are known from  $\mathbb{N}$  or  $\mathbb{R}$ , are generally not valid. For instance, a number can actually have more than two roots  $(\text{mod } n) \neq p * q$ , as can be seen here:  $(\text{mod } 8) : 1^2 \equiv 1 \wedge 3^2 \equiv 1 \wedge 5^2 \equiv 1 \wedge 7^2 \equiv 1$
- The quadratic residues make up at least one (multiplicative) group: Let  $x_1, x_2 \in \text{QR}_n$  and let  $y_1, y_2$  be their roots, then the roots of  $x_1 * x_2$  and  $x_1^{-1}$  are exactly  $y_1 * y_2$  and  $y_1^{-1} : (y_1 * y_2)^2 = y_1^2 * y_2^2 = x_1 * x_2$  and  $(y_1^{-1})^2 = (y_1^2)^{-1} = x_1^{-1}$

### 3.4.1.5 Squares and roots $(\text{mod } p)$

While little can be said about roots and squares  $(\text{mod } n)$  for random  $n$  (see §3.4.1.4), everything is nicely arranged, especially because  $\mathbb{F}_p$  is a field.<sup>29</sup>

- Most importantly, every number has at most two roots (because in groups a polynomial of degree 2 has at the most 2 zeros)<sup>30</sup> We also know that both  $\pm y$  are roots. In general  $y \neq -y$  holds, too, except for  $y \equiv 0$  or  $p \equiv 2$ . (For even or compound numbers  $p$ , this assumption holds for  $y \equiv p/2$ .)

Thus all quadratic residues have exactly 2 roots for  $p > 2$ .

- From this it follows that for  $p > 2$  exactly half of all numbers  $(\text{mod } p)$  are quadratic residues, i.e.:

$$|\text{QR}_p| = \frac{p-1}{2}.$$

because the squaring function is exactly  $2 \rightarrow 1$ . This can be illustrated in the following way:

$x$	$ $	0	$\parallel$	1 2 ... $\frac{p-1}{2}$	$ $	$-\frac{p-1}{2}$	...	-2	-1
$x^2$	$ $	0	$\parallel$	1 4 ... $(\frac{p-1}{2})^2$	$ $	$(\frac{p-1}{2})^2$	...	4	1

(Reminder: Each of the lower cells of the table contains pairwise distinct values. A number contained twice in either of the two cells would have to have more than two squares — which it cannot.)

<sup>29</sup>Obvious: Every element except 0 has a multiplicative inverse, as shown above.

<sup>30</sup>Indirect proof for those prefer it spelled out explicitly: Assume there is a number which has more than 2 roots. Then there are, in particular, two substantially different roots, i.e., two roots, which differ in more than the sign. (If there were no two substantially different roots, then there would be at most two roots anyway.) Let  $x$  and  $y$  now be substantially different roots. Then we have:

$$x^2 \equiv y^2 \pmod{p} \text{ and } x \neq \pm y \pmod{p}.$$

According to the definition of congruence:

$$p | (x^2 - y^2).$$

This is equivalent to  $p | (x+y) * (x-y)$ . Since  $p$  is a prime number,  $p$  must either divide  $(x+y)$  or  $(x-y)$ . Then however  $x \equiv \pm y \pmod{p}$  must be true. Contradiction!

### 3 Cryptography Basics

- For the time being, we define for all  $x \in \mathbb{Z}_p^*$  the Legendre-Symbol just as a denotation to distinguish between squares and non-squares (later called Jacobi-Symbol in a generalized form) as

$$\frac{x}{p} := +1 \text{ if } x \in \text{QR}_p \text{ and } \frac{x}{p} := -1 \text{ else.}$$

- The Euler criterion helps to quickly check whether  $p$  is a square for  $p > 2$ :

$$\frac{x}{p} \equiv x^{\frac{p-1}{2}} \pmod{p}. \quad ^{31}$$

(so far we would have had to try out  $(p-1)/2$  possible roots in order to determine whether  $x$  is a square, which would be exponential in the length  $l$  of  $p$ ; however the exponentiation can be done in  $O(l^3)$ , see §3.4.1.1).

- Sometimes the multiplicativity of the Legendre-symbol comes in handy, i.e.,

$$\frac{x * y}{p} = \frac{x}{p} * \frac{y}{p}. \quad (3.5)$$

This follows immediately from the Euler criterion.

#### 3.4.1.6 Squares and roots $(\text{mod } p)$ for $p \equiv 3 \pmod{4}$

Some things become even easier, if  $p$  satisfies  $p \equiv 3 \pmod{4}$ , which is required in most of the following cryptographic systems.

- To begin with, there is a simple algorithm for root extraction:<sup>32</sup> (so far we would have to try  $(p-1)/2$  possible roots one after the other.) For every  $x \in \text{QR}_p$  it is true that:

$$w := x^{\frac{p+1}{4}} \text{ is a root from } x \pmod{p}.$$

First of all, this formula makes sense because  $p \equiv 3 \pmod{4} \Rightarrow (p+1)/4 \in \mathbb{N}$ . Second of all, we have to prove that  $w^2 \equiv x \pmod{p}$ :

$$w^2 \equiv (x^{\frac{p+1}{4}})^2 \equiv x^{\frac{p+1}{2}} \equiv x^{\frac{p-1}{2}} * x \equiv 1 * x,$$

---

<sup>31</sup>If one accepts the small Fermat theorem (s. §3.4.1.2), then one can easily prove that:  $\phi(p) = p-1$  is for prime numbers, thus  $x^{p-1} \equiv 1$  is valid for  $\pmod{p}$ . Then  $x^{(p-1)/2}$  is a root from 1, thus  $\pm 1$  (it lies then at least in the range of values of the Jacobi symbol).

a) if  $x$  is a square, e.g.,  $x \equiv y^2$ , then  $x^{(p-1)/2} \equiv y^{p-1} \equiv 1$ .

b) the equation  $x^{(p-1)/2} \equiv 1$  can have at most  $(p-1)/2$  solutions as a polynomial equation in the field  $\mathbb{Z}_p$ .

Because of a) we already know that all quadratic residues are solutions, and we know that there are  $(p-1)/2$  quadratic residues. Thus there cannot be further solutions. Thus  $x^{(p-1)/2} \not\equiv 1$  is valid for the square not-remainders; it remains only  $x^{(p-1)/2} \equiv -1$ .

<sup>32</sup>Root extraction is easy in the sense of 'fast', even modulo other prime numbers, but the algorithm is much more complicated [Kran\_86, p.22]

### 3.4 The $s^2 - \text{mod} - n$ -pseudo-random-bitsequence-generator

where the last equation follows directly from the Euler criterion. Moreover, we know that the root  $w$  obtained this way is again a square: It is a Potenz of some  $x^2$ . This will be important when we want to extract roots repeatedly, e.g., in order to extract the  $4^{th}$ ,  $8^{th}$ , etc. root.

- It is true that

$$-1 \notin \text{QR}_p,$$

because if e.g.,  $p = 4 * r + 3$  (with  $r \in \mathbb{N}_0$ ), then, according to the Euler criterion:

$$\frac{-1}{p} \equiv -1^{\frac{p-1}{2}} \equiv (-1)^{2 * r + 1} \equiv -1 \pmod{p}. \text{<sup>33</sup>}$$

- This has the following implication for the two roots  $\pm w$  of some single  $x$ : Let  $w$  be the root found using the above method, thus  $w \in \text{QR}_p$ , then:

$$-w \notin \text{QR}_p,$$

because if both roots were quadratic residues then the following could be deduced:  $-1 \equiv (-w) * w^{-1} \in \text{QR}_p$  (because  $\text{QR}_p$  is a group); contradiction! Thus  $x$  has exactly two  $4^{th}$  roots, namely the  $\pm w$  and two  $8^{th}$  roots, etc.

#### 3.4.1.7 Squares and roots $(\text{mod } n)$ with knowledge of $p, q$

$(n = p * q, p, q$  prime numbers and  $p \neq q$ .)

Now we combine the results from §3.4.1.3, §3.4.1.5, and §3.4.1.6, in order to receive possible operations which only owners of the secret can execute: Knowing the secret, it is possible to find out if a given  $x$  is a square  $(\text{mod } n)$  and, if necessary, extract a root:

- square test: It is valid

$$x \in \text{QR}_n \Leftrightarrow x \in \text{QR}_p \wedge x \in \text{QR}_q.$$

(The owner of  $p, q$  can evaluate the right hand conditions using the Euler criterion.)  
Proof:

(B1) Let, for example,  $w$  be a root of  $x \in \text{QR}_n$ , i.e.,  $w^2 \equiv x \pmod{n}$ . Then

$$w^2 \equiv x \pmod{p}$$

and

$$w^2 \equiv x \pmod{q} \quad (\text{because of } *)$$

---

<sup>33</sup>therefore the choice of  $p \equiv 3 \pmod{4}$  is essential. Because otherwise for large  $p$ ,  $p$  would be  $\equiv 1 \pmod{4}$  and therewith  $-1 \in \text{QR}_p$  according to the Euler criterion.

### 3 Cryptography Basics

- (B2) Conversely, let  $x \in \text{QR}_p \wedge x \in \text{QR}_q$ , e.g.,  $w_p^2 \equiv x \pmod{p}$  and  $w_q^2 \equiv x \pmod{q}$ . According to the Chinese remainder theorem, there is a  $w$ , such that

$$w \equiv w_p \pmod{p} \wedge w \equiv w_q \pmod{q}$$

Therefore, again with (\*), we get

$$w^2 \equiv w_p^2 \equiv x \pmod{p} \wedge w^2 \equiv w_q^2 \equiv x \pmod{q} \Leftrightarrow w^2 \equiv x \pmod{n}$$

- Every  $x \in \text{QR}_n$  has exactly 4 roots: It is an element of  $\text{QR}_p$  as well as of  $\text{QR}_q$  and thus has 2 roots  $\pm w_p$  and  $\pm w_q$  respectively. Each of the 4 combinations can be put together to form one root according to (B2):  $w := \text{CRA}(\pm w_p, \pm w_q)$ .
- Now we will extend the Legendre Symbol to the Jacobi Symbol modulo  $n$ , first again as pure means of denotation:

$$\left(\frac{x}{n}\right) := \left(\frac{x}{p}\right) * \left(\frac{x}{q}\right).$$

The Jacobi Symbol of  $x \pmod{n}$  is not +1 if and only if  $x \in \text{QR}_n$ , as with prime numbers. It evaluates to +1 either if  $(x \in \text{QR}_p \wedge x \in \text{QR}_q)$  or  $(x \notin \text{QR}_p \wedge x \notin \text{QR}_q)$ . (The first condition is equivalent to  $x \in \text{QR}_n$ .) Thus all squares have Jacobi Symbol 1, but not the other way round.<sup>34</sup>

The Jacobi Symbol is multiplicative, i.e.,

$$\left(\frac{xy}{n}\right) = \left(\frac{x}{n}\right) * \left(\frac{y}{n}\right)$$

This follows immediately from its definition and the multiplicative properties of the Legendre Symbol.

#### 3.4.1.8 Squares and roots $\pmod{n}$ with knowledge of $p, q \equiv 3 \pmod{4}$

$(n$  is  $p * q$  with  $p$  being a prime number and  $p \neq q$ .)

- Extracting roots is easy: The roots  $\pm w_p$  and  $\pm w_q$  can be determined with the method explained in §3.4.1.6, and  $w$  can be calculated from them as

$$w := \text{CRA}(\pm w_p, \pm w_q).$$

- -1 is a non-square with Jacobi-Symbol 1:

$$\left(\frac{-1}{n}\right) := \left(\frac{-1}{p}\right) * \left(\frac{-1}{q}\right) = (-1) * (-1) = +1.$$

---

<sup>34</sup>Since  $\pmod{p}$  and  $\pmod{q}$  half of all numbers  $\neq 0$  are quadratic residues, it follows that  $\pmod{n}$  the same is true for only  $\frac{1}{4}$  of all numbers, but exactly half of them have Jacobi-Symbol 1.

### 3.4.1.9 Squares and roots $(\bmod n)$ without knowledge of $p, q$

So that extracting roots and square testing will make good secret operations, we still have to prove or assume that these operations are difficult without knowledge of  $p$  and  $q$ . It turns out that this can be proven for extracting roots. Therefore the following useful theorem may help:

- If someone knows two substantially different roots  $w, w'$  of the same number  $x$   $(\bmod n)$ , i.e.,  $w \not\equiv \pm w' \pmod{n}$ , then he can definitely factor  $n$ :

$w^2 \equiv w'^2 \equiv x \pmod{n} \Rightarrow n|(w^2 - w'^2) = (w + w') * (w - w')$ . From  $w \not\equiv \pm w' \pmod{n}$ , on the other hand, it follows that  $n$  is not a factor of  $(w + w')$  or  $(w - w')$ . This means that each of the two factors must be divisible by at least one prime factor of  $n$ . Thus, one factor can be calculated with

$$\gcd(n, w + w').$$

- The following proof sketch is to show that under the factoring assumption, extracting a root  $(\bmod n)$  is also difficult.<sup>35</sup> This can be used as a sample for all proofs of this kind.

Our aim is to show that “Factoring is difficult  $\Rightarrow$  Extracting roots is difficult”. One shows the equivalent inversion “Extracting roots is easy  $\Rightarrow$  Factoring is easy”. Thus one assumes that there is a root extracting algorithm  $\mathcal{W}$  and uses it to devise a good factoring algorithm  $\mathcal{F}$ . For this,  $\mathcal{F}$  is constructed explicitly such that  $\mathcal{F}$  uses  $\mathcal{W}$  as a subprogram, c.p. Figure 3.16.<sup>36</sup>

---

<sup>35</sup>More formally, this says (similar to the factoring assumption): For each (probabilistic) polynomial algorithm  $\mathcal{W}$  and each polynomial  $Q$  there exists an  $L$  such that for all  $l \geq L$  the following holds: If  $p, q$  are chosen as random prime numbers of length  $l$ ;  $n := p * q$ , and  $x$  is randomly chosen from  $\text{QR}_n$ , then:

$$\mathcal{W}(\mathcal{W}(n, x)) \text{ is a root from } x \leq \frac{1}{Q(l)}.$$

<sup>36</sup>The difference for the so-called Karp reduction [GaJo\_80, p.118], [Paul\_78, p.60], as known from complexity theory, is that here  $\mathcal{F}$  can call  $\mathcal{W}$  a random (of course only polynomially) number of times, while there, only one call is permitted

```

program  $\mathcal{F}$ 
...
    subprogram  $\mathcal{W}$ 
    [black box];
...
begin  $\mathcal{F}$ 
...
    call  $\mathcal{W}$ 
    ...
    call  $\mathcal{W}$ 
}
multiple times, but polynomial
...
end  $\mathcal{F}$ .

```

Figure 3.16: Structure of a Turing reduction from factorization  $\mathcal{F}$  to extracting roots  $\mathcal{W}$

Our specific  $\mathcal{F}$  looks like this:

```

begin  $\mathcal{F}$ 
    choose  $w \in \mathbb{Z}_n^*$  randomly; set  $x := w^2$ ;
    extract root:  $w' := \mathcal{W}(x, n)$ ;
    if  $w \neq \pm w'$  then begin
         $p := \gcd(n, w + w')$ ;
         $q := n \text{div } p$ 
    end
end.

```

It is clear that with  $\mathcal{W}$   $\mathcal{F}$  is polynomial too.

The other claim is: If  $\mathcal{W}$  finds a root  $(\bmod n)$  for a fixed  $n$  with probability  $\varepsilon$ , then  $\mathcal{F}$  factors the same  $n$  with probability  $\varepsilon/2$ .<sup>37</sup> From the point mentioned earlier, it follows that  $\mathcal{F}$  can factor definitely, if

1.  $\mathcal{W}$  actually finds a root and
2.  $w' \neq \pm w$ .

The first point is valid with probability  $\varepsilon$ . Thus we have to show that the root  $w'$  that was found is significantly different from the  $w$  one knows before with probability  $1/2$ . Intuitively, this is true because of the fact that  $\mathcal{W}$  does not know, which root of  $x$  the program already knows. More precisely: The result  $w'$  of  $\mathcal{W}$

---

<sup>37</sup>We can improve the probability by executing  $\mathcal{F}$   $\sigma$ -times; then the probability is  $\varepsilon(1 - 2^{-\sigma})$ , and the algorithm is still polynomial with constant  $\sigma$ .

### 3.4 The $s^2 - \text{mod} - n$ -pseudo-random-bitsequence-generator

depends only on the input to  $\mathcal{W}$ , i.e.,  $(n, x)$  and possibly internal random decisions of  $\mathcal{W}$ . Let the four roots  $x$  be  $\pm w'$  and  $\pm w''$ , then  $\mathcal{W}$  would have  $w'$  as output, regardless of whether  $w = \pm w'$  or  $w = \pm w''$  has been selected outside the subroutine, because that can't be seen from  $x$ . Since  $w$  is chosen randomly, it will take all 4 values with the same probability.  $w \neq \pm w'$  with probability  $1/2$ , which was to be shown.<sup>38</sup>

- It is generally assumed that it is also difficult to test whether  $x \in \mathbb{Z}_n^*$  is a square, but so far this could not be proven. Therefore, one makes a quadratic-residuosity-assumption, similar to the factoring assumption. Two restrictions apply:
  1. There is a fast algorithm for computation of Jacobi symbols, even if  $p$  and  $q$  are not known (with the so-called square reciprocity law). This is not used in the following. But it means that someone who is presented with  $x$  such that  $(\frac{x}{n}) = -1$ , is able to determine that  $x \notin \text{QR}_n$ . Therefore the assumption is limited to those  $x$  with  $(\frac{x}{n}) = +1$ .
  2. Just guessing whether  $x \in \text{QR}_n$ , will give the correct result with a probability of  $1/2$  for a random  $x$  with  $(\frac{x}{n}) = +1$ . Therefore one requires only that no polynomial (guessing) algorithm  $\mathcal{G}$  does substantially better than pure guessing.

$qr$  is the predicate whether something is a quadratic residue, i.e.,

$$qr(n, x) := \text{true, if } x \in \text{QR}_n \quad qr(n, x) := \text{false, otherwise.}$$

The assumption is as follows

*Assumption* For each (probabilistic) algorithm  $\mathcal{G}$  and each polynomial  $\mathcal{Q}$  the following holds: There exists an  $\mathcal{L}$ , such that for all  $l \geq L$ : If  $p, q$  are chosen as

---

<sup>38</sup>Formally: The assumption that  $\mathcal{W}$  is a ‘good’ algorithm for extracting roots is just the opposite of the footnote “extracting root is difficult”. That means that there is nevertheless a polynomial  $Q$  such that for a infinite number of  $l$ : If  $p, q$  are randomly chosen prime numbers of length  $l$ ,  $n = p * q$  and  $x$  is randomly chosen from  $\text{QR}_n$  then the following is true:

$$W(\mathcal{W}(n, x) \text{ is a root of } x) > \frac{1}{Q(l)}. \quad (3.6)$$

We have to show that the existence of  $\mathcal{F}$  contradicts the factoring assumption., i.e., that there is a polynomial  $\mathcal{R}$  such that for infinitely many  $l$  the following is true: If  $p, q$  are chosen as random prime numbers of the length  $l$ ,  $n = p * q$ , then

$$W(\mathcal{F}(n) = (p; q)) > \frac{1}{R(l)}. \quad (3.7)$$

This is shown for  $R := 2Q$  and the same  $l$ 's as in (3.6): Let  $l$  be given. From the declaration of  $\mathcal{F}$  you can see that  $x$  is chosen randomly from  $\text{QR}_n$  (because each of the  $x$ 's have exactly 4 roots and occur at 4 of the  $w$ 's with the same probability) When calling  $\mathcal{W}$ , all assumptions of (3.6) hold, which means that (3.6) is valid. Now it follows, as in the main text above, that in half of all cases  $w' \neq w$  and thus  $\mathcal{F}$  is successful. Altogether, this is valid with a probability of  $(l/(2 * Q(l)))$ , as claimed. So we have shown formally: “Extracting root is easy  $\Rightarrow$  Factorizing is easy”.

random prime numbers of the length  $l$ ,  $n := p * q$ , and  $x$  is randomly chosen from those residue classes with Jacobi symbol +1, then it is valid that:

$$W(G(n, x)) = qr(n, x) \leq \frac{1}{2} + \frac{1}{Q(l)}.$$

- Choosing a random square is easy: One simply chooses a random  $w \in \mathbb{Z}_n^*$  and squares it. Since every  $x$  has the same number of roots (for specialists: exactly 4), every square is chosen with the same probability.

### 3.4.2 Requirements concerning the Pseudo-Random Bit Generator

The following is described more in detail in [BIBS\_86].

#### What is a Pseudo-Random Bit Generator (PRBG)

?

A Pseudo-Random Bit Generator (PRBG) does the following: It generates a *long* Pseudo-Random Bit Sequence from a randomly chosen initial value (seed). That means (c.p. Figure 3.17):

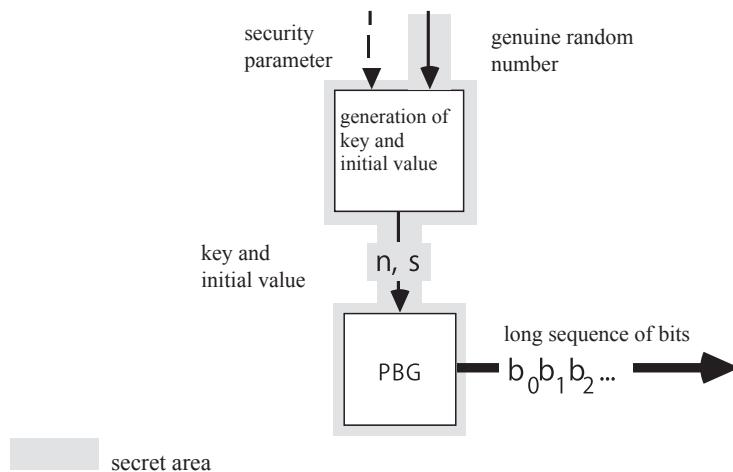


Figure 3.17: Pseudo random number generator

- a key – and seed-generating algorithm belong to it. It creates a key  $n$  and a seed  $s$  of the length  $l$  from the security parameter  $l$ . (A key *or* a seed would be enough at this point. But it is interesting in the later use of asymmetric systems, where the key  $n$  is allowed to be public while the seed  $s$  has to be a secret for the whole time.)

- The actual PRBG is an algorithm which generates a bit sequence  $b_0 b_1 b_2 \dots$  (which can be of arbitrary length) with the key  $n$  and the short seed  $s$ . Though only a polynomial long start-piece of this sequence may be used.
- PRBG is deterministic, i.e., from the same initial value and key the same bit sequence is generated every time.
- PRBG is efficient, thus calculable in polynomial time.
- PRBG is secure, i.e., the results are not distinguishable from genuine random sequences through a polynomial test. (See below for more detail.)

### Usage as pseudo-one-time pad

An important application, is to use the pseudo random bits instead of a one-time pad (see §3.2) as a symmetrical encryption system (see Figure 3.18): Instead of a genuine random-sequence, a pseudo-random-sequence is added bitwise (mod 2) to the plaintext. This idea comes from Vigenére (1523-1596) and in his honor, the pseudo-one-time pad is sometimes called Vigenére's cipher.

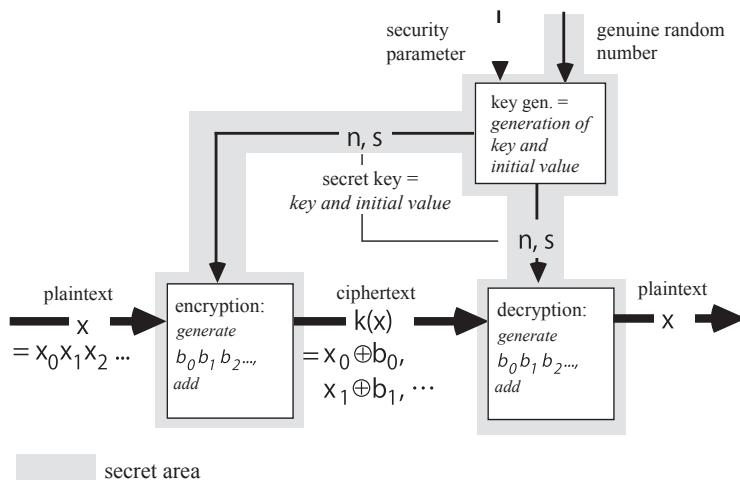


Figure 3.18: Pseudo-one-time pad

With this, instead of the long one-time pad, only the short key and initial value of the PRBG must be exchanged.

**Demands on the pseudo-one-time pad:** For every polynomial limited attacker, the pseudo-one-time pad is supposed to be as safe as a one-time pad, up to an infinitesimal portion of cases.

The idea is that if no polynomial test (thus also no polynomial attacker) can differentiate between pseudo-one-time pads and the genuine one-time pad, then there cannot be a

### 3 Cryptography Basics

difference in regards to a polynomial attacker, whether one encrypts with a pseudo-one-time pad or a genuine one-time pad.

#### Demands on PRBG

The “strongest” demand: PRBG passes *each* probabilistic test  $T$  in polynomial runtime.

Here ‘**passes**’ means: The test cannot differentiate between sequences which the PRBG produces and genuine random-sequences, with a significant probability.

A probabilistic **test**  $T$  with a polynomial runtime is a probabilistic, polynomial time-limited algorithm that assigns to each input out of  $0, 1^*$  a real number out of  $[0, 1]$  (thus a finite sequence it has to test). (The value depends generally on the random decisions in  $T$ ).

The basic idea is now that every individual sequence is not regarded (because each individual finite sequence can be produced with the same probability by a genuine random generator, i.e., to talk of the eventuality of an individual sequence is senseless for cryptography). Instead of this, one compares the average value of the test for the PRBG sequences with the average value for genuine sequences. If the same results are achieved on average with both kinds of sequences, then the test cannot help in differentiating the PRBG sequences from the genuine random sequences.

Therefore the following is specified

- $\alpha_m$  is the average value that  $T$  assigns to a genuine random  $m$ -bit-sequence (i.e., the expectancy value with uniform distribution over absolutely all  $m$ -bit-sequences), and
- $\beta_{l,m}$  is the average value that  $T$  assigns to a  $m$ -bit-sequence generated by PRBG if the security parameter is  $l$  (whereas the average is created over the keys and initial values, which are both created with the security parameter  $l$ ).

Then it is more formally valid (i.e., at a transposition of the formal parameter  $m$  to a arbitrary polynomial  $Q(l)$ ):

A PRBG passes  $T$  if and only if:

$Q$  and all  $t > 0$  are valid for all polynomials: For all sufficiently large  $l$ 's,  $\beta_{l,Q(l)}$  lies in the interval  $\alpha_{Q(l)} \pm 1/l^t$

The following three demands are equivalent to this ‘strongest’ demand (but easier to prove): For each produced finite initial bit sequence, in which an arbitrary (the right or left) bit is missing, every polynomial time-limited algorithm  $P$  (Predictor) is only able to guess the missing bit (i.e., it cannot better predict than with a probability of  $1/2 \pm 1/l^t$ ). Proof ideas (indirect proofs):

1. (Simple Part) Each Predictor is also to be used as a test: One takes all bits from the sequence which is to be tested, except one, gives them as input to the predictor and compares the result with the correct bit from the sequence. The test result is 1, if the predictor has estimated correctly, otherwise the result is 0. The value  $\alpha_m$

### 3.4 The $s^2 - \text{mod} - n$ -pseudo-random-bitsequence-generator

is  $1/2$ , because each predictor can estimate correctly only with a probability of  $1/2$  according genuine random-sequences. So if the predictor estimates substantially different (better) on PRBG sequences, then the PRBG has not passed this test.

2. (More difficult part) To show: From each of the 3 weaker demands follows the 'strongest'.

Let us say that there exists this test  $T$ , which PRBG does not pass, and we show that there is a predictor which forecasts a bit from PRBG too well. This is shown for a left bit here:

For a  $t > 0$ , a polynomial  $Q$ , and infinite  $l$ 's,  $\beta_{l,Q(l)}$  lies on the outside, e.g., above of the interval  $\alpha_{Q(l)} \pm 1/l^t$ .

One now submits to  $T$ , a bit sequence consisting of 2 parts, with the lengths  $j + k = Q(l)$ , the left part is genuinely random and the right is generated by PRBG. One always compares two adjacent types of sequences.

genuinely random

$A_{j+1} = \{r_1 \dots r_j \ r_{j+1} \ b_1 \dots b_k\}$  results in test results closer to  $\alpha_{Q(l)}$

$A_j = \{r_1 \dots r_j \ b_0 \ b_1 \dots b_k\}$  results in test results that are further away, e.g., higher.

generated by PRBG

The things shown above must be at least  
be valid for one  $j$ , because  $A_0$   
are the PRBG sequences and  $A_{Q(l)}$   
are the genuine random-sequences.

Out of this, one constructs a predictor for the bit sequence  $b_1 \dots b_k$  as follows: One additionally chooses a genuine random sequence  $r_1 \dots r_j$ , and applies the test  $T$  twice, namely

$T$  to  $\{r_1 \dots r_j \ 0 \ b_1 \dots b_k\}$ : The result is called  $\alpha^0$

$T$  to  $\{r_1 \dots r_j \ 1 \ b_1 \dots b_k\}$ : The result is called  $\alpha^1$

Guess  $b_0 = 0$  with a probability of  $1/2 + 1/2(\alpha^0 - \alpha^1)$ .

(More detailed: [BIBS\_86, p.375f])

#### 3.4.3 The $s^2 - (\text{mod } n)$ -generator

The  $s^2 - (\text{mod } n)$ -generator is a certain PRBG, which works with squares (and the difficulty of the extracting roots and/or deciding whether something is a square). (In the original publication [BIBS\_86] it is called  $x^2 \pmod{N}$ -generator. I want to introduce this notation for consistency's sake, because the  $x$  is used as a variable designator for the plaintext here and the  $N$  is used in many other papers instead of the small  $n$ , where it is used as designator for the compounded Modulus consisting of two large prime numbers.)

**Definition of the  $s^2 - (\text{mod } n)$ -generator:**

### 3 Cryptography Basics

As the key,  $n = p * q$  is selected with  $|p| \approx |q|$ , and  $p, q$  prime numbers  $\equiv 3 \pmod{4}$ . (here the amount lines mean 'length' in bits.  $p$  and  $q$  are randomly and independently selected, whereby the conditions mentioned in §3.4.1 concerning  $p$  and  $q$  must be kept – otherwise  $n$  could be factorized.)

Select  $s$  randomly as the initial value  $\mathbb{Z}_n^* = \{a \in F_n^* \mid 0 < a < n, \gcd(a, n) = 1\}$ .

Generation of the sequence: Calculate gradually the internal states  $s_0 := s^2 - (\text{mod } n)$ ,  $s_{i+1} := s_i^2 \pmod{n}$ ,  
and form in each step  $b_i := s_i \pmod{2}$  (= last bit of  $s_i$ )

Output:  $b_0 b_1 b_2 \dots$

(as an observation assistance for the designator  $s_i$ : internal states = status = condition of the generator)

example:  $n = 3 * 11 = 33, s = 2$

$s_i:$	4	16	25	31	4	...
$b_i:$	0	0	1	1	0	...

#### security considerations

The security of the  $s^2 - (\text{mod } n)$ -generator can be proven using the factorizing assumption [ACGS\_84, ACGS\_88]. The slightly simpler proof using the CRA is shown here (cp. §3.4.1.9):

Assertion: Under QRA the  $s^2 - (\text{mod } n)$ -generator is an unpredictable (cryptographically strong) PRBG.

According to the paragraph above concerning demands on PRBG, it is sufficient to prove that there is no predictor which is able to forecast well, from the remaining bits, the left bit of the sequences which the  $s^2 - (\text{mod } n)$ -generator produces. More exactly:

- a) a predictor  $P[\bullet, \bullet]$  is a probabilistic, polynomial time-limited algorithm that puts 0 or 1 out when inserting  $n, b_1 \dots b_k (b_i \in \{0, 1\})$ .
- b) one says,  $P$  has an advantage for a certain  $n$  when predicting to the left of  $k$ -bit-sequences of the  $s^2 - (\text{mod } n)$ -generator if and only if:

$$\frac{\sum_{s \in \text{QR}_n} W(P[n, b_1(s) \dots b_k(s)] = b_0(s))}{\phi(n)/4} > \frac{1}{2} + \varepsilon$$

with  $b_i(s) = \text{last bit of } s^2 - (\text{mod } n)$ . (Be aware that  $\phi(n)/4$  is the sample size because  $s$  must be a quadratic residue at the forecasting to the left – otherwise one is not able to extract roots. In contrast to that,  $s$  is allowed to be arbitrary in  $\mathbb{Z}_n^*$  when the sequence is generated to the right.)

c) The  $s^2 - (\text{mod } n)$ -generator is secure if and only if it is valid for each predictor  $P$ , each constant  $0 < \delta < 1$ , each polynomial  $Q$ , and each natural numbers  $t$ : As long as  $l$  is sufficiently large:  $P$  has for all numbers (except of a  $\sigma$ -portion)  $n$  of the length  $l$  at the most, an advantage of  $1/l^t$  for  $n$  when forecasting  $Q(l)$ -bit-sequences to the left.

*Proof.* (sketch):

Assumption: There is a predictor  $P$  for the  $s^2 - (\text{mod } n)$ -generator with a  $\varepsilon$ -advantage

for numbers of the length  $l$ . (To complete the proof, we have to set all quantors over the  $\varepsilon$  as shown in c) ).

**First Step:**  $P$  can be efficiently and uniformly transformed into a procedure  $P^*$  that guesses with an  $\varepsilon$ -advantage the last bit of  $s_0$  to the given  $s_1$  from  $\text{QR}_n$ :

$s_1$  is given. Create  $b_1 b_2 b_3 \dots$  with the  $s^2 - (\text{mod } n)$ -generator. Now apply  $P$  to this sequence. Then  $P$  guesses the bit  $b_0$  with  $\varepsilon$ -advantage. This is exactly the appropriate result of  $P^*$ .

**Second Step:** Now the procedure  $P^*$  is given. From this a procedure  $R$  can be constructed efficiently and uniformly, which guesses with a  $\varepsilon$ -advantage whether a given  $s^*$  is a square with the Jacobi symbol +1:

$s^*$  is given. Set  $s_1 := s^{*2}$ . Apply  $P^*$  to  $s_1$ .  $P^*$  thus calculates the last bit of  $s_0$  with an  $\varepsilon$ -advantage. Thereby  $s_*$  and  $s_0$  are roots of  $s_1$ , and in fact  $s_0$  is itself exactly the one that is a square<sup>39</sup>. Then it is valid that:

$$s^* \in \text{QR}_n \Leftrightarrow s^* = s_0.$$

Based on the last bits, the known  $b^*$  of  $s^*$  and the guessed  $b_0$  of  $s_0$  can now determine if  $s^* = s_0$ . The following is obvious: If  $s^* = s_0$  then their last bits must be equal. It will be shown now that from  $s^* \neq s_0$  follows inversely that the last bits are different: If  $s^* \neq s_0$  then  $s^* \equiv -s_0 \pmod{n}$  (because of the equivalent Jacobi-symbols), thus  $s^* = n - s_0 \in \mathbb{Z}$ . But  $n$  is an odd number, so  $s^*$  and  $s_0$  have different bits.

Thus the procedure  $R$  ends that way: If, and only if,  $b^* = b_0$ , then it is guessed that  $s^*$  is a square, meaning that one guesses as well as  $P^*$ .

**Third Step:** The so constructed procedure  $R$  stands in contradiction to *QRA*.  $\square$

Note: One can show that it is possible to take more than one bit per squaring-step videlicet  $O(\log(l))$ .

### 3.4.4 $s^2 - (\text{mod } n)$ -generator used as an asymmetric encryption system

One can use the  $s^2 - (\text{mod } n)$ -generator as an asymmetric encryption system (cp. Figure 3.19):

- The secret key consists of the factors  $p$  and  $q$  of  $n$ .
- The public key is  $n$ .

---

<sup>39</sup>This was already shown in exercise 3-13k): That only one of the 4 can be a square itself, follows from  $p \equiv q \equiv 3 \pmod{4}$ . Then only one of the two roots  $\pm w_p$  of  $s_1 \pmod{p}$  is a square as shown in §3.4.1.6, and as well as for  $\pm w_q$ . According to §3.4.1.7, the 4 roots of  $s_1$  are combinations of  $\pm w_p$  and  $\pm w_q$ . Last of all, a combination is a square if, and only if, both parts are squares (follows from §3.4.1.7 too).

### 3 Cryptography Basics

- A sender and a recipient do not have a exchange a secret start value  $s$  anymore, but for each message a new random start value  $s$  is chosen by the sender. (In addition to that, the encryption becomes *indeterministic*, which means that the same clear texts are not necessarily encrypted to the same ciphertext, cp. §3.1.1 annotation 2.) So that the receiver is nevertheless able to decode, the help of  $p$  and  $q$  in extracting roots  $(\bmod n)$  should be utilized.
- So that the recipient also has something from which he can extract roots, the senders send him, in addition to the previous ciphertext,

$$x_0 \oplus b_0, x_1 \oplus b_1, \dots, x_k \oplus b_k$$

and also the value

$$s_{k+1}$$

just the value whose last bit would have gotten  $b_{k+1}$ , if one had still needed this.

- The receiver now decodes this way: He calculates backwards the sequence  $s_i$  from  $s_{k+1}$ ,

$$\begin{aligned} s_{i-1} &:= \text{the root from } s_i \text{ which is a square itself} \\ &= CRA(s_i^{(p+1)/4} \pmod p, s_i^{(q+1)/4} \pmod q). \text{<sup>40</sup>} \end{aligned}$$

Then the receiver can determine the last bits  $b_i$  of  $s_i$  and with that he receives the plaintext bits  $x_i$ .

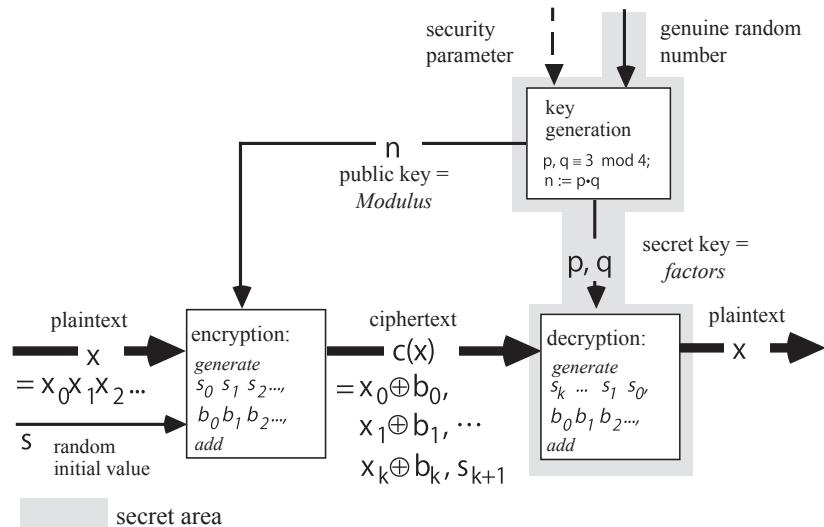


Figure 3.19:  $s^2 \pmod n$ -generator used as an asymmetric encryption system

### 3.4 The $s^2 - \text{mod} - n$ -pseudo-random-bitsequence-generator

**Secure against passive attacks:** Similar to the above, one can prove that the procedure is secure against attacks from passive attackers. (One only has to also consider that the attacker sees each  $s_{k+1}$ .)

**Unsecure against active attacks:** The aim of an active attack is to decode a ciphertext  $x_0 \oplus b_0, x_1 \oplus b_1, \dots, x_k \oplus b_k, s_{k+1}$  which has been observed by an attacker. Therefore the attacker gets arbitrary ciphertexts decoded, which are different from these.

A simple *chosen ciphertext-plaintext-attack* can happen this way (see Figure 3.20): The attacker squares  $s_{k+1}$  and receives  $s_{k+2}$ . He sends

$$x_0 \oplus b_0, x_1 \oplus b_1, \dots, x_k \oplus b_k, 1, s_{k+2}$$

to the victim, which is just the observed ciphertext, except for the inner state  $s_{k+1}$ , after this an inserted bit, here 1, and then the “next” inner state  $s_{k+2}$ . The first  $k + 1$  bits that are received by the attacker as an answer, are the plaintext that he is interested in.

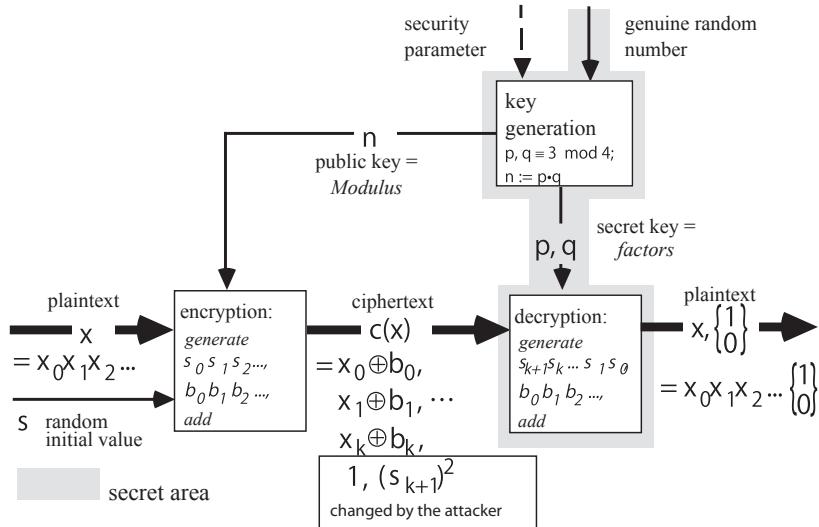


Figure 3.20: Chosen *Ciphertext-plaintext attack* on the  $s^2 \pmod{n}$ -generator used as an asymmetric encryption system

This simple attack could be detected by the practical application of a comparison, and then defeated by not answering the victim. But it can be varied in two different ways and is then much more difficult to recognize.

1. The attacker replaces  $x_0 \oplus b_0, x_1 \oplus b_1, \dots, x_k \oplus b_k$  with  $k + 1$  random bits. Through the answer of the victim, he is now able to calculate the pseudo-random-bit-sequence. With this he can decode the original ciphertext.
2. The attacker squares not just once, but  $j$  times, thus calculating  $s_{k+1+j}$ . Accordingly he has to insert  $j$  bits, whose value in the answer does not interest him. Even

when the answers only allow  $k + 1$ -bit-messages, the attacker has only to renounce the first  $1 + j$ -bit-messages.

Much more dangerous than these explained *message-orientated* attacks, is the fact that within an active attack, even a complete break is possible, i.e., the factorizing of the modulo. This cannot be seen here, but it can be read in [GoMT\_82, algorithm 5].

## 3.5 GMR: A strong cryptographic signature system

GMR, whose name is derived from the initials of its inventors Shafi Goldwasser, Sivio Micali, und Ronald L. Rivest, is historically seen as the first practical and cryptographically strong signature system. It can be shown that even with an adaptive active attack, it is impossible (if the factorization assumption applies) to fake even a single new signature (no matter under what senseless message). (cp. §3.1.3: It is secure against the weakest success c2 and on the strongest attack b. adaptive). It was published in [GoMR\_88].

The proof of security is relatively complex; that is why only the workings of the system and some essential parts of the proof are pictured <sup>41</sup>. You can also derive some properties of GMR from the security against adaptive active attacks; but here a simple bottom-up strategy is chosen:

1. permutation pairs are introduced where only the one who knows the cryptographic secret can find collisions. Such pairs are called *collision resistant* <sup>42</sup>.
2. with the help of a collision resistant permutation pair, it is shown how to sign a one bit message. Therefore, parallel to the permutation pair you need an authentic *reference* that is published as part of the public key.
3. constructing *many* collision resistant permutation pairs out of *one*.
4. it is explained how to dendriformly authenticate *many* references out of *one* so that many messages with many collision resistant permutations can be signed.
5. some efficiency improvements are shown.

### 3.5.1 Basic function: collision resistant permutation pairs

The basic function of GMR are the so called *claw-resistant permutation pairs with trap door* (collision resistant permutation pairs with secret) <sup>43</sup>.

A collision between two permutations  $f_0, f_1$  within the same domain definition is a pair  $(x, y)$  with  $f_0(x) = f_1(y)$  (fig. 3.21).

---

<sup>41</sup>By courtesy of [FoPf\_91]

<sup>42</sup>In the literature you can find an older, inappropriate term. Instead of collision resistant (which means it is really hard to find collisions) collision free was used - which of course cannot be applied literally

<sup>43</sup>In the original publication the term claw-“free” (collision-“free”) permutation pairs with trap door was used, which does not exist literally. That is why I use the term collision resistant.

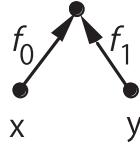


Figure 3.21: Collision of a permutation pair

“Collision resistant” means that under a cryptographic assumption it is impossible to find those collisions if you do not know the secret (even though they do exist: because they are in the same domain definition, for every  $z$  there is exactly one  $x$  with  $f_0(x) = z$  and exactly one  $y$  with  $f_1(y) = z$ .)

But the owner of the secret should be able to compute the inverse of the functions.

**Construction under the factorization assumption:** You can construct collision resistant permutation pairs under the factorization assumption with the usual secrets (cp. §3.4.1): The secrets are two randomly and stochastically, independently chosen big primes  $p$  and  $q$  with the additional condition:

$$p \equiv 3 \pmod{8}, q \equiv 7 \pmod{8}$$

As explained in §3.4.1 (“About prime number creation”) there are a sufficient amount of primes that meet these requirements.

In general, permutations are squaring functions of which we already know that the secret helps to invert them (therefore to extract the root).

But first of all, the squaring function is not a permutation, but a mapping from 4 roots onto the same square (cp. §3.4.17). That is why we have to limit the definition domain so that only one root is in it. Therefore:

$$D_n = \left\{ x \in \mathbb{Z}_n^* \mid \left(\frac{x}{n}\right) = 1, x < n/2 \right\}^{44}$$

The sign “ $<$ ” on numbers modulo  $n$  refers to the standard representation of  $0, \dots, n-1$ .

Secondly, we have to take care that the result is within the domain of definition. But that is easy (cp: §3.4.1.8): The result  $y$  at the squaring is a square; its Jacobi-symbol is

---

<sup>44</sup>That in fact exactly one of the 4 roots has this quality follows from  $p \equiv q \equiv 3 \pmod{8}$ :

1. §3.4.1.8 shows that the 4 roots are  $\text{CRA}(\pm w_p, \pm w_q)$  where  $\pm w_p, \pm w_q$  are the roots mod  $p$  and  $q$ .
2. Because of §3.4.1.6 holds: One of  $\pm w_p$  has the Jacobi symbol +1, the other -1. The same applies for  $\pm w_q$ . Let w.l.o.g.  $w_p, w_q$  be those with Jacobi-symbol +1.
3. If  $z = \text{CRA}(x, y)$ : According to the Jacobi-symbol’s definition  $\left(\frac{z}{n}\right) = \left(\frac{z}{p}\right) \cdot \left(\frac{z}{q}\right)$ . The Jacobi symbol of  $z$  modulo  $p$  only depends on the residue class mod  $p$ , therefore  $\left(\frac{z}{p}\right) = \left(\frac{x}{p}\right)$  and  $\left(\frac{z}{q}\right) = \left(\frac{y}{q}\right)$ . That is why  $\left(\frac{z}{n}\right) = \left(\frac{x}{p}\right) \cdot \left(\frac{y}{q}\right)$ .
4. Therefore the Jacobi-symbols of  $w := \text{CRA}(w_p, w_q)$  and  $-w := \text{CRA}(-w_p, -w_q)$  are both +1 and those of the other 2 roots -1. Of  $w$  and  $-w$ , one is exactly  $< n/2$ .

### 3 Cryptography Basics

+1. If  $y \geq n/2$  you will take  $-y$ . This is  $< n/2$  and its Jacobi-symbol is +1 too, because  $\frac{-1}{n} = +1$ .

Thirdly, we need a second permutation with similar properties like  $x^2$ . For this you will take  $4x^2 = (2x)^2$ . This choice is motivated by the following:

$$f_0(x) := \begin{cases} x^2 \bmod n, & \text{if } (x^2 \bmod n) < n/2 \\ -x^2 \bmod n, & \text{otherwise} \end{cases}$$

$$f_1(x) := \begin{cases} 4 \cdot x^2 \bmod n, & \text{if } (4 \cdot x^2 \bmod n) < n/2 \\ -4 \cdot x^2 \bmod n, & \text{otherwise} \end{cases}$$

**Proof.** **Outline explaining that those pairs of permutations are collision resistant:** That was already shown for  $f_0$  and similarly for  $f_1$ .

It will be shown for the collision resistance that you can factorize with every collision. For that you will need the unproven Lemma that for  $p \equiv 3 \pmod{8}$ ,  $q \equiv 7 \pmod{8}$ :

$$\left(\frac{2}{p}\right) = -1 \text{ and } \left(\frac{2}{q}\right) = +1$$

(Second complement theorem on the law of quadratic reciprocity [ScSc\_73, page 83])

From that follows:

$$\left(\frac{2}{n}\right) = -1.$$

Consider that somebody has found a collision  $x, y$  with  $f_0(x) = f_1(y)$ . Then

$$x^2 \equiv \pm(2y)^2 \bmod n$$

The sign “-” is not possible because -1 is not a square mod  $n$ . So  $x$  and  $2y$  are roots of the same number. In accordance to §3.4.1.9, we can definitively factorize, if they are substantially different:

$$x \neq \pm 2y$$

That follows from the fact that they have different Jacobi Symbols: Because  $x, y \in D_n$ ,  $x$  and  $y$  have the Jacobi Symbol +1 and because of §3.4.1.8:

$$\left(\frac{\pm 2y}{n}\right) = \left(\frac{\pm 1}{n}\right) \cdot \left(\frac{2}{n}\right) \cdot \left(\frac{y}{n}\right) = 1 \cdot (-1) \cdot 1 \neq 1 = \left(\frac{x}{n}\right)$$

The inversion through the secret works like this: If  $x = f_0^{-1}(y)$  is desired, you should first of all test if  $y$  is a square (cp. §3.4.1.7). Otherwise it is according to the definition  $f_0$ :  $y = -x^2$  and  $-y = x^2$ . Then you extract the root of  $y$  resp.  $-y$  with the method from §3.4.1.8. The resulting root  $w$  is a square itself (cp. §3.4.1.6), so it has the Jacobi Symbol +1. Depending on if it is  $<$  or  $\geq n/2$ , you let  $x = +w$  resp.  $-w$ .

For  $f_1^{-1}(y)$ , in addition to the root extraction mod  $n$ , you have to divide through 2. Because 2 has the Jacobi Symbol -1, but  $f_1^{-1}(y)$  is within the domain of definition of the permutations and therefore must have the Jacobi Symbol +1, and due to the multiplicativity of the Jacobi Symbol, the value from extracting the root must have the Jacobi Symbol -1. For this purpose, after the root extraction mod  $p$  and mod  $q$ , the results are put (with different signs) into the CRA.  $\square$

**In general:** If you define collision resistant pairs of permutations precisely, you will not only need a pair  $(f_0, f_1)$ , but a whole family of them, depending on the key, so that everybody who needs a pair can chose one with a secret only known to him. That requires a key generation algorithm  $gen$  for choosing a secret and a public key.

During the concrete construction this was automatically the case:  $gen$  is the algorithm for choosing  $p$  and  $q$  (which represent the secret) and calculating the public key  $n$ . For every  $n$  there is a different pair, so to be exact you would have to write  $f_0^{(n)}, f_1^{(n)}$ .

Collision resistance is defined by the nonexistence of a polynomial algorithm that can find collisions with a significant probability (analogue to the factorization assumption).

Furthermore, you will need a method for efficiently finding a random element from the domain of definition with only the public key:

**The choice of a random element from  $D_n$ :** Pick a  $z \in \mathbb{Z}_n^*$  randomly and square it; if  $z^2 < n/2$  you will pick  $x := z^2$  otherwise  $x := -z^2$ .

### 3.5.2 Mini-GMR for a message containing only one bit

If you want to sign a one-bit long message, you can simply chose a collision resistant pair of permutations: the secret signing key  $s$  is  $(p, q)$ ; the public verification key  $t$  consists of  $n$  and a reference  $R$ , which the signer randomly picks from  $D_n$ .

The signature beneath the message  $m=0$  is  $f_0^{-1}(R)$  and beneath  $m=1$  is  $f_1^{-1}(R)$  (fig. 3.22).

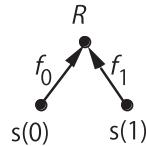


Figure 3.22: Mini-GMR for a one-bit-message.  $s(0)$  and  $s(1)$  are the possible signatures

*Proof. Security (outline):* A real active attack does not exists here because only one message is signed. But it is possible that the faker conceived the signature beneath  $m=0$  and wants to fake the one beneath  $m=1$  or vice versa. You could guess according to Figure 3.22 that this means the attacker finds a collision.

But it is not that easy: He got help for the one part of the collision from the signer, who knows the secret and can calculate collisions anyway. We must show that the faker can get *any* collision even without that help.

W.l.o.g. we only watch in the first case, where  $m=0$ . We assume that there is a good faking algorithm  $\mathcal{F}$  with the input  $(n, R, s(0))$ , and then we show that a collision finding algorithm  $\mathcal{K}$  must exist:

$\mathcal{K}$ : Input  $n$

Pick a “signature”  $S_0 \in D_n$  randomly (here it is required for this to be done with only the public key. Because  $S_0$  is not picked with the signing algorithm, it is not denoted with  $s(0)$ . The same goes for  $S_1$  and  $s(1)$ .)

Calculate  $R := f_0(S_0)$ .

Apply the input  $(n, R, S_0)$  to  $F$ . If  $F$  is successful, name the result  $S_1$ .

Output:  $(S_0, S_1)$ .

It is clear that this  $K$  always finds a collision if the call of  $F$  is successful. It only needs to be shown that  $F$  is equally often successful if it is called with a “real” reference and signature. Real references are random and  $R$  is random too (because  $S_0$  is random and  $f_0$  is a permutation, so every  $R$  occurs with exactly one  $S_0$ ). For this,  $R S_0$  is the (only) right signature.  $\square$

### 3.5.3 Basic signatures: big tuples of collision resistant permutations

For the second step, we will enhance GMR for signing messages (but still only one) with any length desired.

The idea is the same as above for one-bit: You would like to have a function  $f_m$  (like  $f_0$  and  $f_1$  for messages with  $m=0$  resp.  $=1$ ) for every one of these messages. As a public key you will again chose a reference  $R$ <sup>45</sup> in addition to  $n$ . The signature beneath that message shall be

$$s(m) := f_m^{-1}(R)$$

(Figure 3.23) whereas you can break  $f_m^{-1}$  only with the secret.

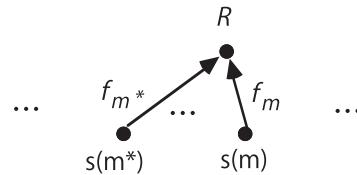


Figure 3.23: Principle of GMR’s basic signature

Of course you cannot specify all of the many functions  $f_m$  explicitly. Therefore they are constructed from one pair of permutations:

**Basic construction of the functions  $f_m$  out of a pair  $(f_0, f_1)$ :** If you have picked a collision resistant pair of permutations  $(f_0, f_1)$ , then you can construct a function  $f_m$  for the message  $m$  through a bit by bit composition of  $f_0$  and  $f_1$  in dependency from  $m$ :

<sup>45</sup>For the simplified case in which the reference is known to the attacker *before* his choice to sign a message is made known, the security of GMR is not proven. And we do not see how that could be achieved.

*example:*  $m = 100101, f_{100101}(x) := f_1(f_0(f_0(f_1(f_0(f_1(x))))))$

With the knowledge of the secret you may calculate  $f_0^{-1}, f_1^{-1}$  and so  $f_m^{-1}$ . The evaluation direction of  $m$  flips

*example:*  $m = 100101, f_{100101}^{-1}(x) := f_1^{-1}(f_0^{-1}(f_1^{-1}(f_0^{-1}(f_1^{-1}(f_0^{-1}(f_1^{-1}(x)))))))$

The advantage of this is that  $f_m$  exists for messages  $m$  with any length desired. There is no need to split  $m$  into blocks, nor hash functions to be used.

However the composition of  $f_0$  and  $f_1$  may contain hazards. The “back counting” receiver that tests the signature with  $f_m$ , obtains for every application of  $f_0$  and  $f_1$  a valid signature that is shortened by the last bit.

*example:*  $f_1(Sig_{100101}) = f_1(f_{100101}^{-1}(R)) = f_{100101}^{-1}(R) = Sig_{100101}$

That can be prevented by fixing the form of the “valid message” through a prefix free mapping  $pref(\cdot)$ : No prefix (i.e., beginning part) of a message mapped that way may be a prefix free mapping of another message at the same time. The original message obtains a “valid tag” that is voided if the last bit is lost. Now you sign  $pref(m)$  with  $f_{pref(m)}^{-1}(R)$ .

You can derive from the collision resistance of  $(f_0, f_1)$  that the family of  $f_{pref(m)}$  is collision resistant in the same manner. Therefore it is hard to find two messages  $m, m^*$  and “signatures”  $S, S^*$  with  $f_{pref(m)}(S) = f_{pref(m^*)}(S^*)$ .

**Prefix free mapping:** You can create an efficient prefix free mapping by adding a length predicate to the front of the message  $m$ . That is a constantly long field (like a computer word) that contains the length of the message. <sup>46</sup>

In the case that the length of the message is fixed, an additional prefix free mapping becomes unnecessary.

### 3.5.4 Complete GMR: authentication of many references

The simplest idea for signing many messages, would be to announce a certain number of references in the public key to “use up the” signature for a signature. This procedure would have the disadvantage that the public reference list would become very long. <sup>47</sup> What remains from this idea is that for any signature needed, another randomly chosen reference is used, like  $R_0, R_1, \dots$  for the 1-st, 2-nd, ... signature. (That makes it possible that an active attacker gets his signature with a different reference. This makes it more difficult for active attacks; one of the basic ideas of GMR).

<sup>46</sup>The advantage of GMR to sign messages with any length desired will not become lost if you e.g., enable the insertion of further length fields with an “extension bit”. To be specific, you can pick the sign bit of the length field.

<sup>47</sup>For this simplified case where the reference is known to the attacker *before* a message to sign is being picked, the security of GMR is not proven. And we do not see how that could be achieved.

### 3 Cryptography Basics

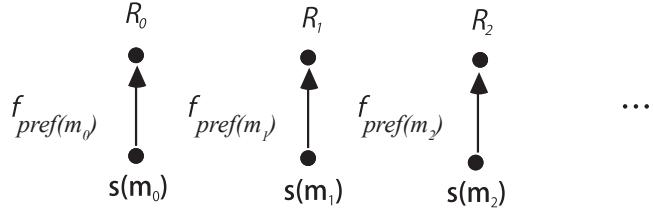


Figure 3.24: Basic idea, how to use GMR for signing multiple messages

But you cannot announce the specific  $R_i$  with the message - otherwise the attacker could chose a signature  $Sig$  randomly for his message of choice and calculate with  $f_0$  and  $f_1$  the appropriate reference  $R := f_{pref(m)}(Sig)$ .

So it would be an good idea to authenticate the references with signatures too, by including a single reference  $r_\varepsilon$  into the public key. Then you sign the list of references  $R_0, R_1, \dots$  in relation to  $r_\varepsilon$ . Now you can use the appropriate  $R_i$  for signing the actual message. The disadvantage is that every signature would have to come along with the list of all references.

Thats why you use a dendriform authentication of the references, where for every authenticated reference you sign two new ones. Before that you define the number  $2^b$  of references that you want to get that way.

Only one reference  $r_\varepsilon$  is published. The  $2^b$  references  $R_0, R_1, \dots$  provided for message signatures (now called N-Signatures) are now attached to the leafs of a binary tree with depth  $b$ ; every node of the tree again contains a randomly picked reference  $r_j$ . The tree is called references tree (fig. 3.25).

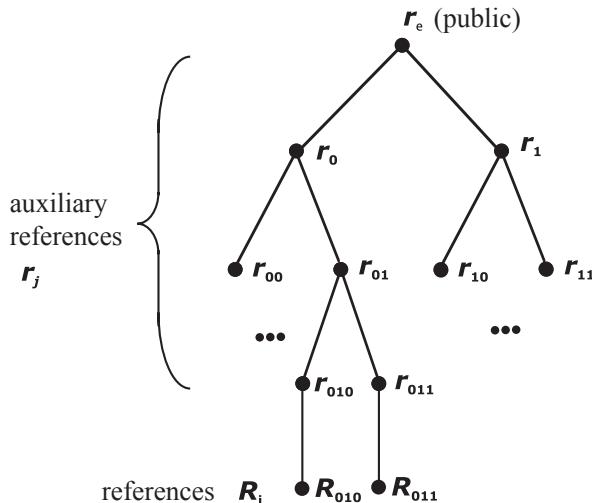


Figure 3.25: Tree of references

### 3.5 GMR: A strong cryptographic signature system

For every signature a signature head is calculated that authenticates the current reference  $R_i$  in relation to the public  $r_\varepsilon$ :

Starting with public reference  $r_\varepsilon$  (the root of the reference tree), every auxiliary reference  $r_j$  on the path from  $r_\varepsilon$  to  $R_i$  is used to sign both of their successor nodes  $r_{j0}$  and  $r_{j1}$  (example in fig. 3.26): for this  $r_{j0}$  and  $r_{j1}$  are concatenated (with separators), interpreted as messages, and mapped prefix free. Then the signature  $KSIG := f_{pref(r_{j0}|r_{j1})}^{-1}(r_j)$  is calculated. These node signatures are called K-Signatures. Finally the attached reference  $R_i$  is authenticated with  $RSIG := f_{pref(R_i)}^{-1}(r_j)$ ; this reference signature is called R-Signature.<sup>48</sup>

Because the references  $r_j$  at the nodes of the reference tree that are authenticated in the signature head almost all have the same length, you can assume a fixed length if you enlarge the shorter references by adding zeros in front. The security parameter  $l$  that defines the length of the modulus  $n$  (in bit) should obviously be chosen. Therefore an additional prefix free mapping becomes unnecessary.

During testing, not only the N-Signature need to be verified, but also the  $b$  K-Signatures to all nodes  $r_j$  of the path through the reference tree and the R-Signature of the reference  $R_i$ . As complexity estimations have shown, the additional effort becomes minimal with longer messages.

**Efficiency improvements:** First of all, you have to notice that the signer does not have to generate the whole reference tree beforehand, but only the reference  $R_i$  that is actually needed, the upper auxiliary references and their brothers and sisters (because they are authenticated together). What needs to be saved is shown in example  $R_2$  (fig. 3.26).

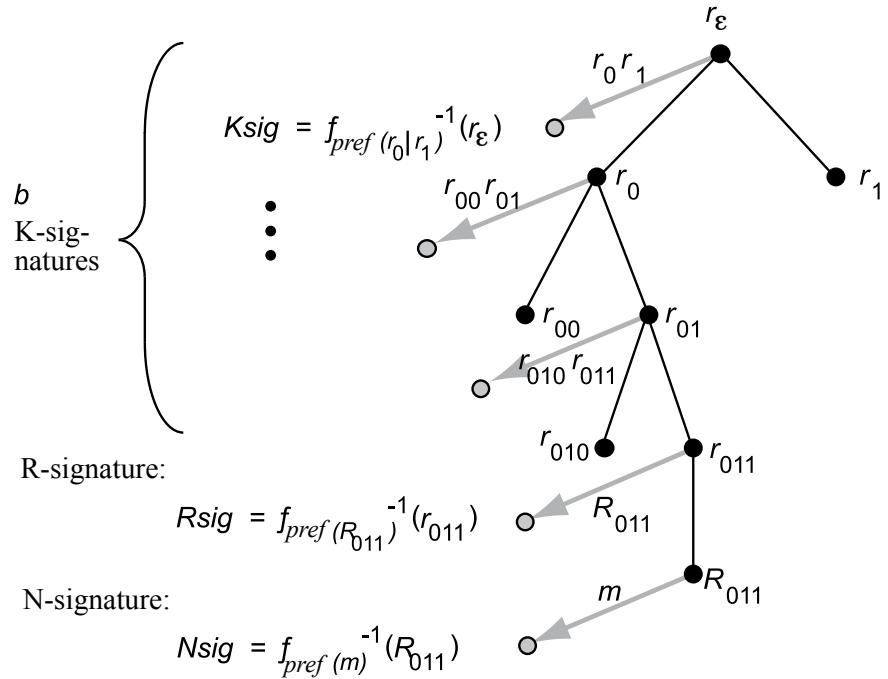
For every signature, the signer can reuse (adding to its own signature head) the previous beginning part of the path in the reference tree with all already calculated K-Signatures from the former signature. Therefore it is recommended not only to keep the necessary auxiliary references “memorized”, but also the whole signature head of the former signature.

Because calculation of the inverse function  $(f_0^{-1}, f_1^{-1})$  is more expensive than calculating with  $f_0$  and  $f_1$ , you should avoid using  $f_0^{-1}$  and  $f_1^{-1}$  wherever possible. It becomes clear that indeed for every signature, only one authentication with  $f^{-1}$  ( $f_0^{-1}$  and  $f_1^{-1}$ ) is necessary for each one:

For every new signature, the signature head can be calculated “bottom-up” with  $f_0$  and  $f_1$ : The signature  $NSIG$  for message  $m$  is picked randomly and the reference is calculated as  $R_i := f_{pref(m)}(NSIG)$ . (Because  $f_0$  and  $f_1$  are permutations, the randomness of  $NSIG$  guarantees the randomness of  $R_i$ .) Subsequently the R-Signature  $RSIG$  is picked randomly,  $r_i$  is calculated etc. until a “joint node” is reached where the path taken over

---

<sup>48</sup>For K- and R-Signatures you must use different secrets as for N-Signatures; this is needed in the security proof and is shown in Figure 3.28. For the sake of readability, this discrimination is not mentioned. For example, we simply write  $p$  and  $q$ , even though for N-Signatures they are different from those for K- and R-Signatures. Which one is meant is clear enough through the context. Also, the necessity of R-Signature only becomes clear in the proof.


 Figure 3.26: Signature for  $m$  relative to reference  $R_2$ 

is joint with the currently newly calculated one. Here you need  $f_0^{-1}$  and  $f_1^{-1}$  once (fig. 3.27).

For the sake of an overview into GMR, at the end of this introduction it is shown in Figure 3.28 in the usual representation of cryptographic systems of §3.1.1. Depending on the implementation, it may be useful to make the tree depth  $b$  a part of the public key.

#### 3.5.4.1 Further efficiency improvements

Especially the expensive calculations can essentially be sped up by using the trick of Goldreich, which permits, instead of a bit by bit application of  $f_0^{-1}$  and  $f_1^{-1}$ , an efficient calculation of  $f_w^{-1}(R)$  for any bit sequences  $w$  within one step. Furthermore, there are some more places where the signer can make use of the secret  $(p, q)$ . For details see [FoPf.91].

For an indication of the achievable efficiency with this method: In a software implementation

- on a MAC IIfx with the MC68030 CPU and 40 Mhz
- for a message of 8192 bit
- for a tree depth of 10 and modulus length 512

### 3.6 RSA: The most known system for asymmetric encryption and digital signatures

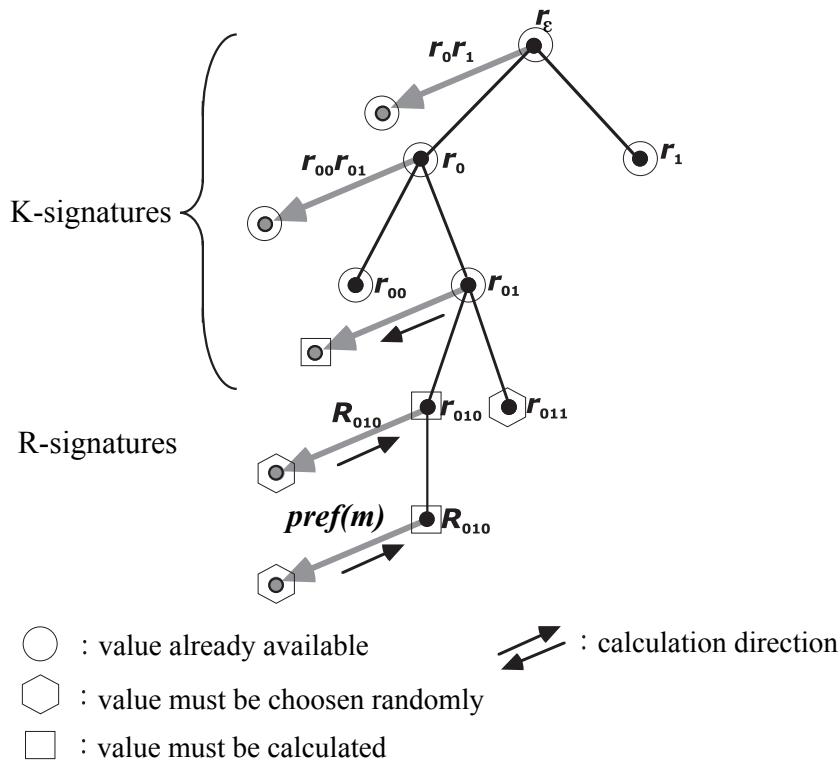


Figure 3.27: “Top-down” and “bottom-up” calculations

signing takes about 3,2 sec. and testing 16,6 sec. [FoPf\_91]. Testing takes significantly longer because you cannot calculate mod of the primes but only mod of the products (cp. §3.6.5.2) and you need to test the whole tree. At signing, you need to calculate the N-Signature and R-Signature, but on average only one K-Signature.

Of course hardware implementations would definitively be faster.

## 3.6 RSA: The most known system for asymmetric encryption and digital signatures

RSA, named after the initial letters of its inventors Ronald L. Rivest, Adi Shamir and Leonard M. Adleman, is the most known asymmetric cryptographic system. It was published in 1978 [RSA\_78] and can be used either as an asymmetric encryption system or digital signature system.

As with the  $s^2\text{-mod-}n$  generators (§3.4) and the collision resistant permutations and as described in GMR, the security of RSA is based on the factorization assumption. But unlike these systems, there is no proof that breaking RSA would consequently mean an algorithm for factorization. That is why RSA can only be put into the class of “well

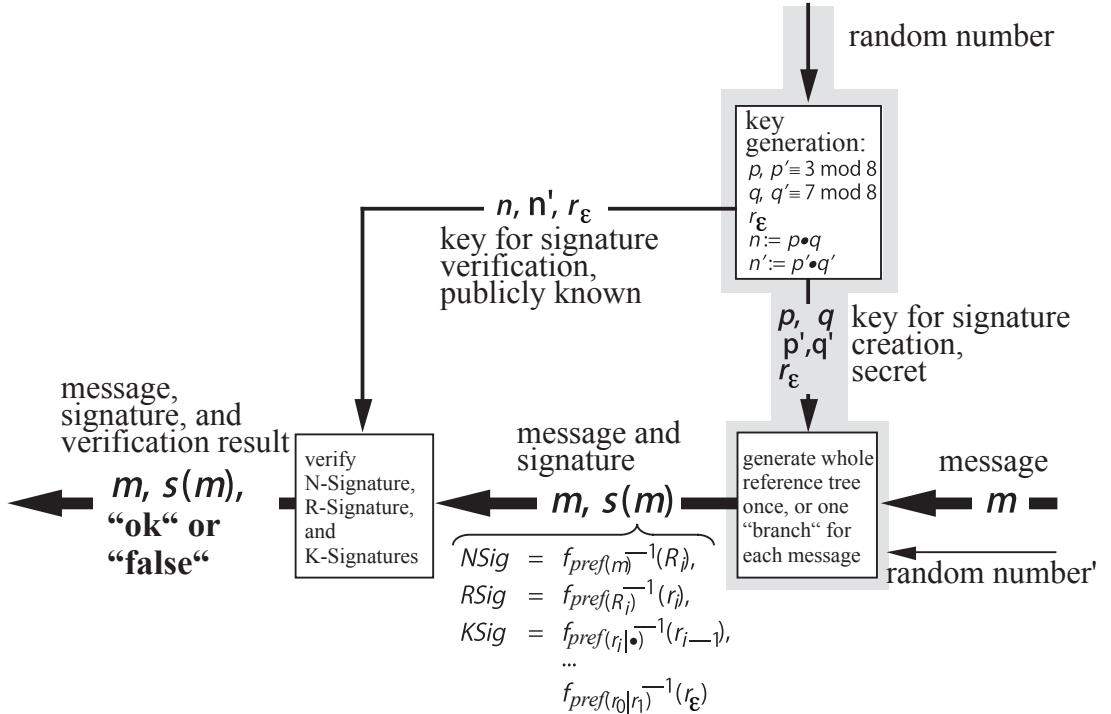


Figure 3.28: The digital signature system GMR

examined” in Figure 3.12.

### 3.6.1 The asymmetric cryptographic system RSA

The core idea of RSA consists of exponentiation of messages (in a residue class ring), calculating exponents inversive to each other, and distributing the knowledge about the exponents properly.

**Key generation** with security parameter  $l$ :

1. Pick two primes  $p$  and  $q$  randomly and stochastical independent so that  $|p| \approx |q| = l$  and  $p \neq q$ . <sup>49</sup>

<sup>49</sup>  $|p|$  means the length of  $p$  in a bit.

The condition could be left out because it would only harm an exponentially small fraction of all cases. Because for  $p \neq q$   $\phi(n)$  can be represented quite easily and the decryption not almost always but always works  $p \neq q$  is demanded here. (Who does not believe that the decryption does not always work should calculate the following example:  $p=q=5$ ;  $\phi(p^2)=p(p-1)$ , so  $\phi(25) = 20$ .  $c=7$  results in  $d=3$  and  $d \cdot c = 21 \equiv 1 \pmod{20}$ .  $(15^3)^7$  is  $\equiv 0 \pmod{25}$  instead of  $\equiv 15$ ). While  $p$  and  $q$  should be picked as almost equally long (even the same length would be no problem),  $p$  and  $q$  - as mentioned - should not be equal. They even should not be almost equal because then  $n$  would be easy to factorize. The factorization would be done as such: Extract the root of  $n$  in  $\mathbb{R}$  and look below it for a factor

### 3.6 RSA: The most known system for asymmetric encryption and digital signatures

2. Calculate  $n := p \cdot q$

3. Pick  $c$  with  $3 \leq c < \phi(n)$  and  $\text{bcd}(c, \phi(n))=1$ . <sup>50</sup>

4. Calculate  $d$  with  $p, q, c$  as multiplicative inverses of  $c$  modulo  $\phi(n)$ . <sup>51</sup>. Thus holds:

$$c \cdot d \equiv 1 \pmod{\phi(n)}$$

Publish  $c$  and  $n$  as public keys and keep  $d, p, q$  as private key.

**Encryption resp. decryption** is made by exponentiation with  $c$  and  $d$  in  $\mathbb{Z}_n$ .

What needs to be proven is that encryption and decryption are inversive for all messages  $m$ :

---

of  $n$ . If  $p$  and  $q$  are approximately equal this can be done efficiently. Even if  $p$  and  $q$  are exactly equally long (but their choice was stochastically independent) only an exponentially small part would be “approximately equal”, which means it does not need to be proofed. To make the breaking of RSA more difficult, the primes  $p$  and  $q$  should not only have the properties of §3.4.1 which make the factorization more difficult. Additionally for at least one of the big prime factors of  $p-1$  and  $q-1$  the following should hold: Reduced by one it has a big prime factor on its own. That parries a special attack on RSA [DaPr\_89, page 232]. Even with these additional properties there are enough primes  $p$  and  $q$  left and they can be found efficiently [BrDL\_91].

<sup>50</sup>Often  $c$  is picked *randomly* off the interval from the ones that are relatively prime to  $\phi(n)$ . In other words: pick  $c$  from  $\mathbb{Z}_{\phi(n)}^*$

<sup>51</sup>The is done with the extended algorithm of euklid described in §3.4.1.1.

### 3 Cryptography Basics

claim For all  $m \in \mathbb{Z}_n$  holds:  $(m^c)^d \equiv m^{c \cdot d} \equiv (m^d)^c \equiv m \pmod{n}$

proof The first two congruence signs are valid within the calculation of congruences. The third's validity is to be shown.

$$\begin{aligned} c \cdot d &\equiv 1 \pmod{\phi(n)} \Rightarrow \\ c \cdot d &\equiv 1 \pmod{p-1} \Leftrightarrow \\ \exists k \in \mathbb{Z} : c \cdot d &= k \cdot (p-1) + 1 \end{aligned}$$

For this  $k$  and all  $m \in \mathbb{Z}_n$  holds:  $m^{c \cdot d} \equiv m^{k \cdot (p-1)+1} \equiv m \cdot (m^{p-1})^k \pmod{p}$ . With Fermat's little theorem (§3.4.1.2) for all  $m$  relatively prime to  $p$  holds:

$$m^{p-1} \equiv 1 \pmod{p}$$

Consequently, for all  $m$  relatively prime to  $p$ :  $m \cdot (m^{p-1})^k \equiv m \cdot 1^k \equiv m \pmod{p}$ . This also holds for the trivial case  $m \equiv 0 \pmod{p}$ :

$$\forall m \in \mathbb{Z}_n : m^{c \cdot d} \equiv m \pmod{p}$$

The corresponding argument for  $q$  produces:

$$m^{c \cdot d} \equiv m \pmod{q}$$

Because congruence holds true for  $p$  and  $q$ , it also holds (due to §3.4.1.3) for  $p \cdot q = n$ . Finally:

For all  $m \in \mathbb{Z}_n$  holds:  $m^{c \cdot d} \equiv m \pmod{n}$ .  $\square$

Every element of  $\mathbb{Z}_n$  can be a message which can be decrypted unambiguously after encryption through modular exponentiation. This makes the modular exponentiation with  $c$  *injective*. Because all ciphertexts are in  $\mathbb{Z}_n$  so image and inverse image are the same and therefore equally powerful. So modular exponentiation with  $c$  is also *surjective* which makes it a *permutation*. Then you can extract the  $c$ -th root out of *every* element of  $\mathbb{Z}_n$ .

Unfortunately there is no proof of security of RSA known (yet?). It should show that: If RSA is easy to break, then it is easy to factorize. For the strongest form of breaking (the weakest form of security proofs): If someone can extract the  $c$ -th root modulo  $n$ , then he can factorize  $n$ . Yet nobody could do so.

#### 3.6.2 Naive and insecure usage of RSA (of a person or action) showing a lack of experience, wisdom, or judgement.

Here the insecure and naive (we will see this in §3.6.3) usage of RSA as an asymmetric encryption system and digital signature system is shown. In both cases, plaintext is represented by numbers  $m$  with  $0 \leq m \leq n$ . For this purpose, plaintexts have to be divided into several blocks (each with maximal length of  $|n| - 1$  bit; **blockage**).

**3.6 RSA: The most known system for asymmetric encryption and digital signatures**

### 3.6.2.1 RSA as asymmetric encryption system

The identifiers from §3.6.1 are already suitably chosen::

**Encryption** is done by modular exponentiation with  $c$

For the plaintext block  $m$ :  $m^c \bmod n$

**Decryption** is done by modular exponentiation with  $d$

For the ciphertext block  $m^c$ :  $(m^c)^d \bmod n = m$

Figure 3.29 shows the complete system.

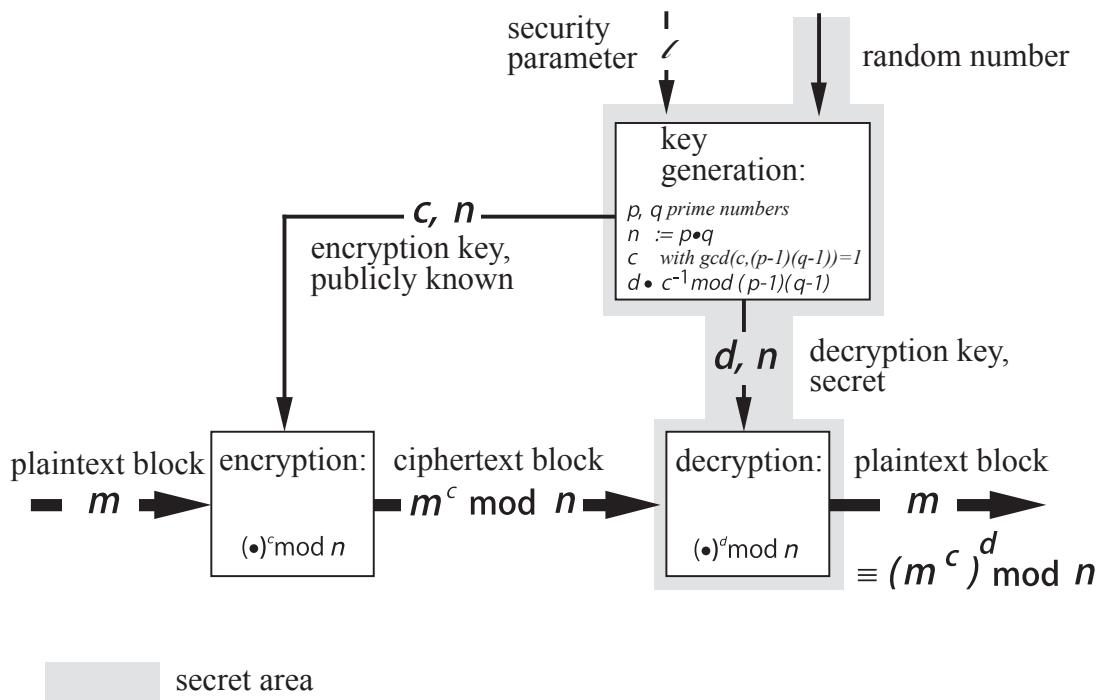


Figure 3.29: Naive and insecure usage of RSA as an asymmetric encryption system

*example:* Key generation: Let  $p = 3, q = 17$  so  $n = 51$

$$\text{So } \phi(n) = (p-1) \cdot (q-1) = 32$$

Let  $c = 5$ ; verify if  $\text{bcd}(5,32)=1$ . (Euklids algorithm). This is true.

Desired  $d = c^{-1} \bmod \phi(n)$  so  $5^{-1} \bmod 32$ . With Euklids extended algorithm  $d = 13$ .

*Encryption:* Let the plaintext  $m = 19$ . The according cipher text  $S = 19^5 \equiv 19^2 \cdot 19^2 \cdot 19 \equiv 361 \cdot 361 \cdot 19 \equiv 4 \cdot 4 \cdot 19 \equiv 4 \cdot 76 \equiv 4 \cdot 25 \equiv 49 \pmod{51}$ .

*Decryption:*  $S^d = 49^{13} \equiv (-2)^{13} \equiv 1024 \cdot (-8) \equiv 4 \cdot (-8) \equiv -32 \equiv 19 \pmod{51}$  what is indeed the plaintext.

### 3.6.2.2 RSA as digital signature system

The identifiers from §3.6.1 are renamed as follows:

$$c \rightarrow t, \quad d \rightarrow s.$$

Thus this holds:

**Signing** is done by modular exponentiation with  $s$ .

For the plaintext block  $m$ :  $m^s \bmod n$ .

**Testing** is done by modular exponentiation of the signature with  $t$  and an appended comparison of the result with the corresponding text block.

For the plaintext block  $m$  with the signature  $m^S$ :  $(m^S)^t \bmod n = m?$

Figure 3.30 shows the complete system.

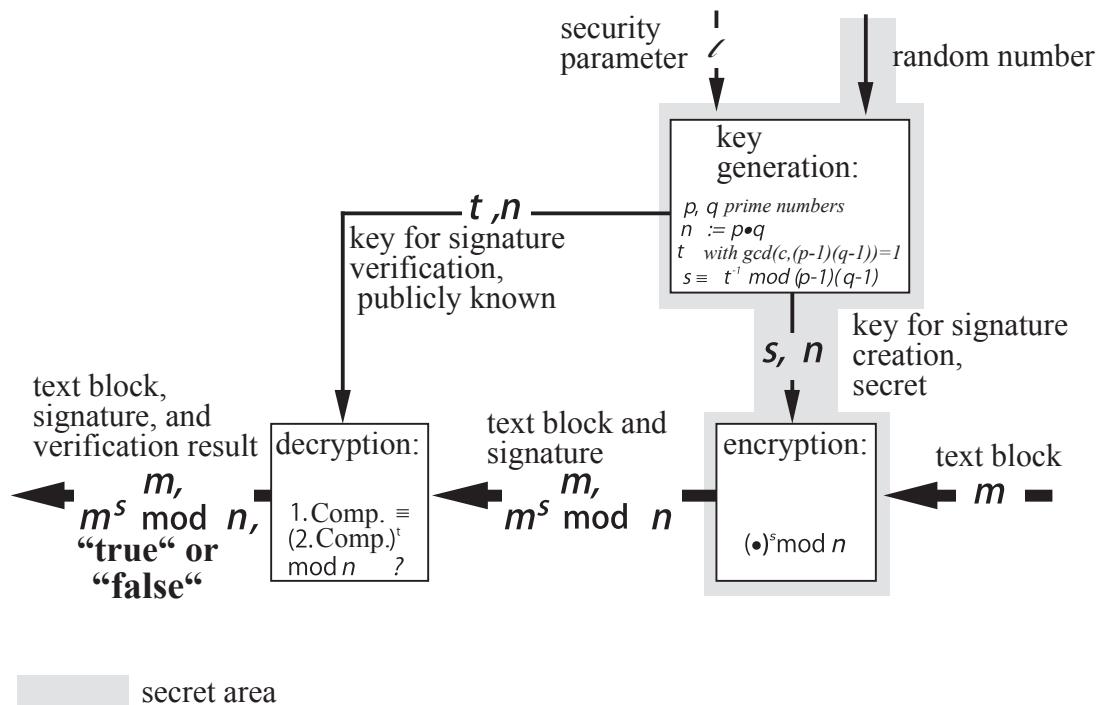


Figure 3.30: Naive and insecure usage of RSA as a digital signature system

### 3.6.3 Attacks, especially multiplicative attacks from Davida and Moore

Unfortunately, while the usage of RSA shown in §3.6.2 is easy to understand, it is mostly insecure. Because RSA has a multiplicative structure (this will be explained soon) the

### 3.6 RSA: The most known system for asymmetric encryption and digital signatures

encryption system as well as the digital signature system can be broken without the attacker knowing anything about the secret key.

Since the multiplicative attack for RSA as a signature can be more systematically explained, we will begin with it here.

#### 3.6.3.1 RSA as a digital signature system

Anybody could fake the digital signatures described in §3.6.2.2 by calculating backwards: He picks a signature, exponentiates it modularly with  $t$  and receives the corresponding text block. This **passive attack** breaks RSA **existentially** but not selectively, because the attacker cannot specify the message he would like to have a signature for (cp. §3.1.3.1 and §3.1.3.2). The minimum that must be demanded, in addition to §3.6.2.2, is that “useful” text blocks occur only occasionally with this attack.

In order to systematically explain active attacks which allow a *selective* break of the signature system of §3.6.2.2, a simple passive attack will first be described. It does not fake signature better than the one currently shown, but explains the multiplicative structure of RSA. This is required for the following active attacks.

If an attacker knows two signatures  $m_1^S$  and  $m_2^S$  for text blocks  $m_1$  and  $m_2$ , he can gain a third signature  $m_3^S$  and the corresponding textblock  $m_3$  through modular multiplication:

$$\begin{aligned} m_3^S &:= m_1^S \cdot m_2^S \bmod n, \\ m_3 &:= m_1 \cdot m_2 \bmod n \end{aligned}$$

$m_3^S$  is indeed the signature for  $m_3$  because:  $m_1^S \cdot m_2^S = (m_1 \cdot m_2)^S$ . In other words: RSA has a **multiplicative structure** or more specifically: RSA is a **homomorphism** on multiplication. As mentioned above, you can only do an **existential** break with this passive attack.

If an active attack is possible, a selective break with help from the multiplicative structure can be performed (attack from Davida [Merr\_83][Denn\_84][Bras\_88]):

The attacker picks a text block  $m_3$  of his own choice.

He picks a  $m_1 \bmod n$  for which a mutliplicative inversive  $m_1^{-1}$  exists.

He calculates  $m_2 := m_3 \cdot m_1^{-1} \bmod n$ .

He has  $m_1$  and  $m_2$  signed.

He receives  $m_1^S$  and  $m_2^S$ .

He calculates  $m_3^S := m_1^S \cdot m_2^S \bmod n$ .

This (non-adaptive) active attack requires *two* signatures from the victim to fake one selectively. The following attack, improved by Judy Moore [Denn\_84], only requires *one* signature to fake one selectively.

The attacker picks a text block  $m_3$  of his own choice.

He picks a random number  $r$  from  $\mathbb{Z}_n$  ( $1 \leq r < n$  and  $r$  has for mod  $n$  a multiplicative inversive  $r^{-1}$ ).

### 3 Cryptography Basics

He calculates  $m_2 := m_3 \cdot r^{-1} \pmod{n}$ .

He has  $m_2$  signed. <sup>52</sup>

He receives  $m_2^S$ .

He knows:  $m_2^S \equiv (m_3 \cdot r^t)^S \equiv m_3^S \cdot r^{t \cdot S} \equiv m_3^S \cdot r \pmod{n}$ .

He calculates  $m_3^S := m_2^S \cdot r^{-1} \pmod{n}$

The idea behind this (non-adaptive) active attack is to multiply the message  $m_3$  which he would like to have signed with a “processed” random number  $r^t$ . The one attacked signs it blindly (i.e., without knowledge of what is behind it) and the attacker just has to divide it through  $r$ . The application of this attack from Judy Moore was published in 1985 by David Chaum [Cha8\_85]. In §3.9.5 and §6.4.1 it is explained thoroughly.

#### 3.6.3.2 RSA as asymmetric encryption system

If RSA is applied as in §3.6.2.1, only a *deterministic* asymmetric encryption system is executed. An attacker can guess plaintext blocks that are likely to occur and encrypt them with the public key deterministically. Then he compares the ciphertext sent to the key owner with the self generated ones. If they are equal he can “decrypt” them (cp. annotation 2 in Figure 3.4).

Because the multiplicative attacks described in §3.6.3.1 only use properties of RSA which, in the naive usage, exist as a signature system as well as a encryption system, they can be transferred for use as encryption system. This is only shown for the improved attack from Judy Moore.

*Non-adaptive active* attack for a *selective* break of RSA naively used as an asymmetric encryption system:

The attacker picks a ciphertext block  $S_3$  of his own choice (like the one he observed).

He picks a random number  $r$  from  $\mathbb{Z}_n$  ( $1 \leq r < n$  and  $r$  has for mod  $n$  a multiplicative inversive  $r^{-1}$ ).

He calculates  $S_2 := S_3 \cdot r^c \pmod{n}$ .

He decrypts  $S_2$ . <sup>53</sup>

He receives  $S_2^d$ .

He knows:  $S_2^d \equiv (S_3 \cdot r^c)^d \equiv S_3^d \cdot r^{c \cdot d} \equiv S_3^d \cdot r \pmod{n}$ .

He calculates  $S_3^d := S_2^d \cdot r^{-1} \pmod{n}$

The idea behind this (non-adaptive) active attack is the multiplication of the ciphertext  $S_3$ , which he wants to have decrypted with a “processed” random number  $r^c$ . The one

<sup>52</sup>Is  $m_3 \in \mathbb{Z}_n^*$  the attacked one does not gain any knowledge about  $m_3$  because  $r^t$  is distributed uniformly in  $\mathbb{Z}_n$ . However, the attacker does not need the help from the one attacked anyway: For  $m_3 = 0$  for all  $n$  is  $m_3^S = 0$ . In all other cases the  $bcd(m_3, n)$  gives a non-trivial factor of  $n$  (concretely  $p$  or  $q$ ). That would mean the attacker would have broken RSA completely (cp. §3.1.3.1)

<sup>53</sup>Here too the attacked one does not gain any knowledge about  $S_3$  because  $r^c$  is distributed uniformly in  $\mathbb{Z}_n$ . However the attacker does not need the help from the attacked one anyway: For  $S_3 = 0$  for all  $n$  is  $S_3^d = 0$ . In all other cases the  $bcd(S_3, n)$  gives a non trivial factor of  $n$  concretely  $p$  or  $q$ . That would mean the attacker would have broken RSA completely (cp. §3.1.3.1)

### 3.6 RSA: The most known system for asymmetric encryption and digital signatures

attacked decrypts it **blindly** (i.e., without knowledge of what is behind it) and the attacker just has to divide the plaintext received through  $r$ .

#### 3.6.4 Thwarting the attack (prevent (someone) from accomplishing something.)

In this section it is explained how all known attacks on RSA as an encryption system as well as a digital signature system can be thwarted. However “known attacks” are clearly pointed out because even we do not have any proofs for them. Finally, not even the most skillful usage of RSA can be better than RSA on its own (cp. annotation about security of RSA at the end of §3.6.1).

The usage of RSA for encryption as well as for authentication makes the application of a **collision resistant hash function** helpful. We already learned something about that in §3.5.3. But there we considered them as a **permutation family**. As a reminder: Bit sequences  $m$  with any length desired were assigned to collision resistant permutations  $f_m$ . If you consider these sequences as input of a function, which applies to permutation onto a fixed value  $x$  and the permuted value  $y$  as output, then you have constructed a **collision resistant hash function**. To be more precise:

$$f_m(x) = y$$

is a **permutation** for  $m$  fixed: Input  $x$ , Output  $y$   
and a **hash function** for  $x$  fixed: Input  $m$ , Output  $y$

A collision of the hash function

$$f_m(x) = f_{m^*}(x) \text{ with } m \neq m^*$$

is also a collision of the permutation family

$$f_m(x) = f_{m^*}(z) \text{ with } m \neq m^*$$

because  $x = z$  is allowed. The other way around does not hold of course because  $x = z$  is not true in general. Therefore the constructed hash function is at least as collision resistant as the underlying permutation family. For  $f_m(x)$  with  $x$  fixed and input  $m$  we will write shortly  $h(m)$  and name  $h$  a collision resistant hash function (constructed from a collision resistant permutation family).

More efficient, but not provably more secure collision resistant hash functions can be constructed out of block ciphers (cp. §3.8.3).

##### 3.6.4.1 RSA as asymmetric encryption system

RSA becomes an **indeterministic** encryption system where a random number  $z$  is assigned (it has nothing to do with the random number for the key generation; therefore it is marked as random number' in Figure 3.31) to every plaintext block  $m$ .

Active attacks are prevented by adding **redundancy** to every plaintext block through encryption, which is checked by the decrypter. This addition of redundancy should

guarantee that the modular multiplication of two plaintext blocks with redundancy will not result in a third plaintext block with appropriate redundancy.

The redundancy can be created by applying the plaintext block on a hash function whose output is appended on the text block. Of course, the hash function should have (if it does at all) another multiplicative structure than RSA for neutralizing RSA's one. This can be done by choosing a modulus for the permutations in §3.5.3 that is independent from the RSA modulus.

Figure 3.31 shows the complete system for combining indeterministic encryption and redundancy with collision resistant hash functions.

**Encryption** of a plaintext block is done by prepending a random number, applying the collision resistant hash function  $h$  on the random number and the plaintext block, appending the result, and together with the modular exponentiation with  $c$  of all three components.

For the plaintext block  $m$  and the random number  $z$ :  $(z, m, h(z.m))^c \bmod n$

**Decryption** is done by modular exponentiation with  $d$ . This results in three components. It is checked by applying the hash function  $h$  on the first two elements if it results in the third one. Only then is the second component issued.

For the ciphertext block  $(z, m, y)^c \bmod n$ :

$((z, m, y)^c)^d \bmod n =: z, m, y. h(z.m) = y?$ . Output if  $h(z.m) = y$  is  $m$ .

Complicated and therefore in a certain sense, provable secure constructions for the combination of redundancy and indeterministic encryption are described in [BeRo.95]. These constructions do not only apply to RSA, but also every asymmetric encryption system which is a permutation (aka length fixed block ciphers; cp. §3.8.2 and §3.8.2.7).

### 3.6.4.2 RSA as digital signature system

To prevent backward calculating and for neutralizing the multiplicative structure of RSA, a collision resistant hash function is applied to the text block before the modular exponentiation. If the hash function accepts arguments of any length desired, it has the additional advantage of signing texts with any length desired in one piece. The blockage problem for digital signature systems is therefore solved.

If the hash function can be computed faster than RSA its usage with longer messages gives a speed advance to the complete system compared to the application of RSA as described in §3.6.2.2.

**Signing** is done by applying the hash function and modular exponentiation of hash result with  $s$ .

For the text block  $m$ :  $(h(m))^s \bmod n$ .

**Testing** is done by modular exponentiation of the signature with  $t$  and a subsequent comparison of the result with the hash value of the corresponding text block.

### 3.6 RSA: The most known system for asymmetric encryption and digital signatures

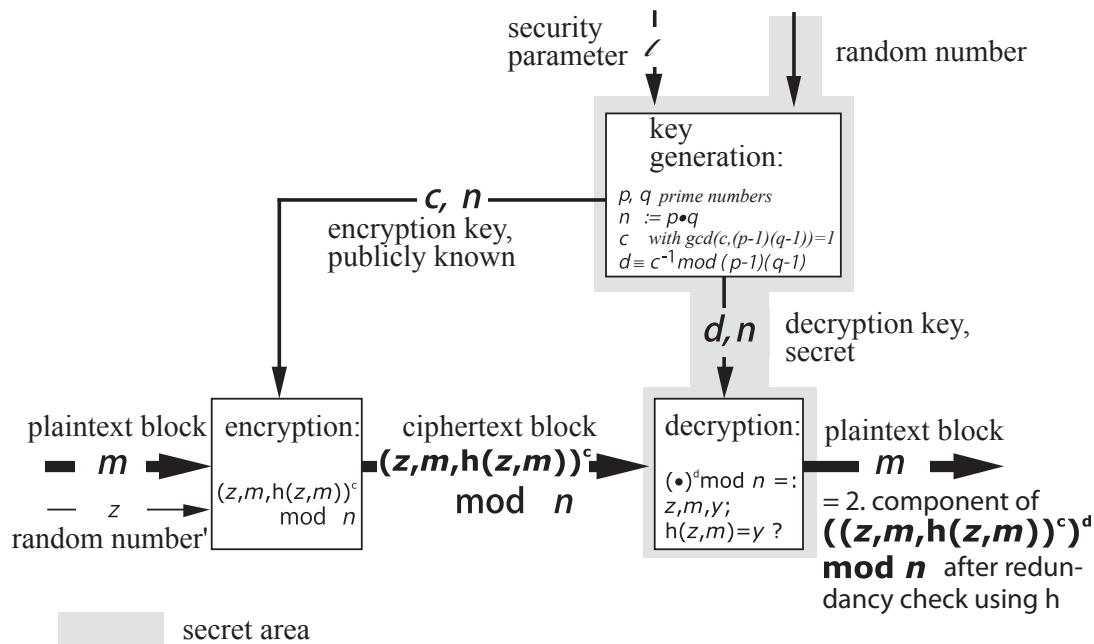


Figure 3.31: Secure usage of RSA as an asymmetric encryption system: Redundancy is checked by a collision resistant hash function  $h$  and indeterministic encryption

For a text block  $m$  and a signature  $(h(m))^s$ :  $((h(m))^s)^t \pmod{n} =: y$ ,  $h(m) = y$ ?

Figure 3.32 shows the complete system.

## 3.6.5 Efficient implementations of RSA

Through skillful implementation the complexity of RSA can be reduced significantly. First of all, this is shown for the user of the public key and then for the owner of the private key.

### 3.6.5.1 Public exponent with almost only zeros

Except for the condition that the public exponent must be relatively prime to  $(p-1)(q-1)$ , it can be chosen freely (cp. §3.6.1). It is obvious to choose one as short as possible (except for leading zeros) to reduce the necessary multiplication and squaring operation. If it contains some zeros after the leading one, you save one modular multiplication per digit, cp. “square and multiply” in §3.4.1.1.  $p$  and  $q$  have to be picked appropriately so that  $(p-1)$  and  $(q-1)$  are relatively prime to the public exponent.

Because  $(p-1)(q-1)$  is even, the smallest possible choice of the public exponent is 3.

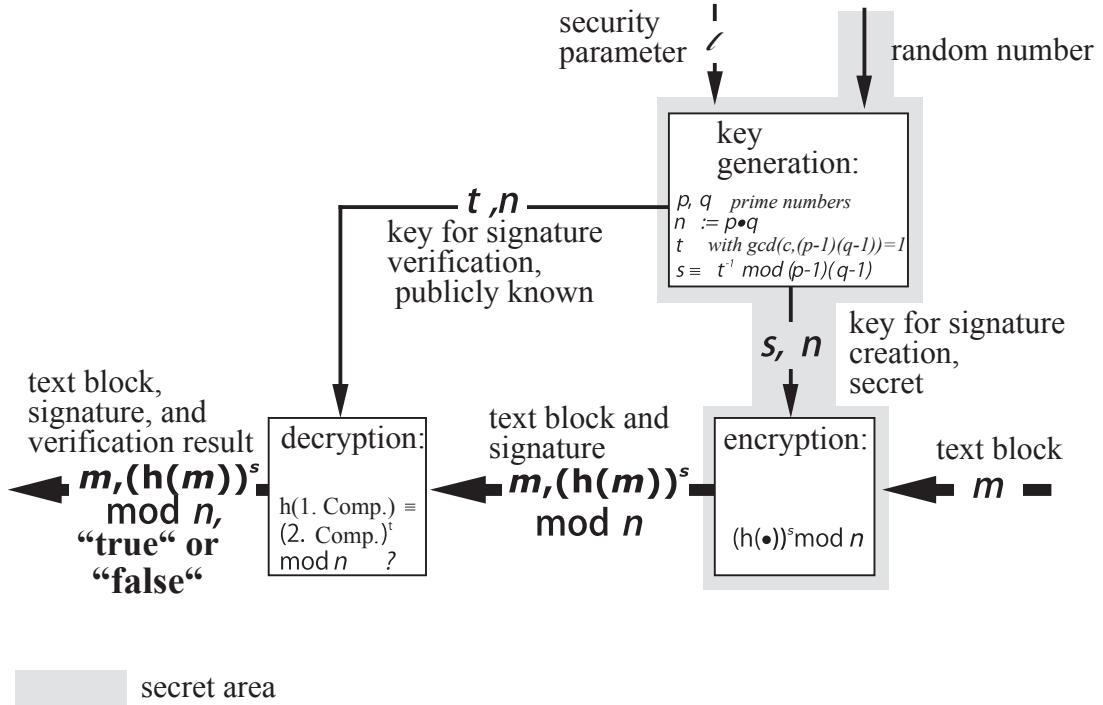


Figure 3.32: Secure usage of RSA as a digital signature system with a collision resistant hash function  $h$

For RSA as a digital signature system there is no known opposing argument. For RSA as an encryption system there is a *passive attack* from Johan Håstad that breaks RSA *message oriented* [Hast\_86][Hast\_88]:

A participant sends the same message  $m$  to three others, whose public exponents are 3 and their moduli  $n_1, n_2$ , and  $n_3$ . The moduli are pairwise relatively prime, otherwise the attacker could factorize at least one (so he would get  $m$ ).

The attacker observes:  $m^3 \bmod n_1$ ,  $m^3 \bmod n_2$ ,  $m^3 \bmod n_3$ .

With CRA he can determine  $m^3 \bmod n_1 \cdot n_2 \cdot n_3$ .

Because  $m^3 < n_1 \cdot n_2 \cdot n_3$ ,  $m^3 \bmod n_1 \cdot n_2 \cdot n_3 = m^3$ . The attacker can extract the root from  $m^3 \bmod n_1 \cdot n_2 \cdot n_3 = m^3$  in  $\mathbb{Z}$  and receives  $m$ .

The attack can be generalized in two ways:

Because there is no search through the plaintext space and it is a *passive attack*, the application of RSA (indeterministic encryption as well as redundancy creation and verification) described in §3.6.4.1 does not thwart the attack if the same random number was used for all three messages.

If the public exponent is  $c$ , then  $c$  encryptions of the same message are sufficient for the attack. Though in this case the attacker also has to calculate with numbers  $c$ -times long.

### 3.6 RSA: The most known system for asymmetric encryption and digital signatures

As a good compromise between thwarting the attack and the required computing power, the value of  $2^{16} + 1$  is proposed as a public exponent in many places. [DaPr\_89, page 235].

There is no argument against all participants picking this as their public exponent. The public exponents would not be need to be published then, and a public RSA key would consist of only the modulus.

#### 3.6.5.2 The owner of the secret key calculates the factors modulo

In situations where the efficiency for the owner of the private key is emphasized, you are tempted to pick a shorter *private key* instead of a “shorter” *public key* (cp. §3.6.5.1). This is possible because the public and the private exponent behave symmetrically due to the formulas from §3.6.1. Even if you do not make the clumsy mistake of choosing a secret exponent like 3,  $2^{16} + 1$  or any other small number that can be found through trial and error (simple search), the system may be insecure if the private exponent is smaller than a fourth of the modulus length [Wien\_90]. So be careful with this kind of efficiency improvements!

Without changing the choosing of the key in the slightest - therefore without any change to the security - the owner of the secret key can save up to three quarters of his calculations: Instead of modulus  $n$ , he calculates modulo with both the factors  $p$  and  $q$ . Afterwards he calculates the value desired with modulo  $n$  from both of the results with CRA:

To calculate the  $c$ -th root of the ciphertext blocks  $S$  efficiently - that is  $S^d \equiv w \pmod{n}$  - the owner of the secret key determines once and for all:

$$\begin{aligned} d_p &:= c^{-1} \pmod{p-1} \text{ so that } (S^{d_p})^c \equiv S \pmod{p} \text{ and} \\ d_q &:= c^{-1} \pmod{q-1} \text{ so that } (S^{d_q})^c \equiv S \pmod{q}. \end{aligned}$$

To extract the  $c$ -th root from a concrete  $S$ :

$$\begin{aligned} (S^{d_p})^c \pmod{p}, (S^{d_q})^c \pmod{q} \text{ and then} \\ w := CRA((S^{d_p})^c \pmod{p}, (S^{d_q})^c \pmod{q}) \end{aligned}$$

Then  $w^c \equiv (S^{d_p})^c \equiv S \pmod{p}$  holds, as well as  $w^c \equiv (S^{d_q})^c \equiv S \pmod{q}$ .

Because  $p$  and  $q$  are relative prime follows:  $w^c \equiv S \pmod{n}$ .

How much faster is this actually? In the domain of the security parameter (the length of  $p$  and  $q$ ) we are interested in holds:

The complexity of the modular exponentiation grows cubical, the modular multiplication quadratic, and the modular addition linear with the length of the modulus. Because  $|n| = 2 \cdot l$ , the complexity of an exponentiation modulo  $n$  is proportional to  $(2 \cdot l)^3 = 8 \cdot l^3$ . The complexity of two exponentiations of half the length (of  $p$  and  $q$ ) is proportional to  $2 \cdot l^3$ . The complexity of CRA is based on two multiplications and one addition modulo  $n$  and is therefore proportional to  $2 \cdot (2 \cdot l)^2 + 2 \cdot l = 8 \cdot l^2 + 2 \cdot l$ .

### 3 Cryptography Basics

Because the proportional constants <sup>54</sup> are approximately equal, the complexity of CRA for even the smallest relevant value of  $l=300$  is notably smaller than 2% of the complexity of 2 exponentiations of half length. So calculating with modulo  $p$  and  $q$  reduces the complexity by the factor

$$\frac{8 \cdot l^3}{2 \cdot l^3} = 4$$

#### 3.6.5.3 Encryption performance

In [Sedl\_88], a hardware implementation of RSA according to §3.6.2 is specified with a speed of 200 kbit/s for the decryption with modulus length of 660 bit. This seems to be sufficient for todays security of RSA. The speed is achieved through the usage of the method shown in §3.6.5.2. If the encryption is done with the exponent  $2^{16} + 1$ , then the speed goes up to 3 Mbit/s.

A corresponding software implementation on an Apple Macintosh IIfx (MC68030, 40Mhz, 32 Kbyte onboard cache, 80 ns RAM) achieved a decryption speed of 2.5 kbit/s with a modules length of 512 bit.

#### 3.6.6 Security and usage of RSA

No matter how you make use of RSA as an asymmetric cryptographic system, it cannot be more secure than the factorizing the modulus is hard to do. And there exists no proof yet, that breaking RSA is as hard as factorizing (cp. §3.6.1).

So what are the useful fields of operation for RSA?

It is helpful for the discussion to go back to Figure 3.12 in §3.1.3.5. There we can see that the important field 3 is filled with only one system which is not equivalent to a pure standard assumption. Therefore efficient asymmetric **encryption systems** that are cryptographically strong relative to a pure standard assumption, are still not known.

RSA may not be proven to be cryptographically strong (as secure as factorizing is difficult) but for its use as an indeterministic asymmetric encryption system as in §3.6.4.1, no successful attacks are known - and this for years. If asymmetric concealation is required and if you cannot rule out active attacks, you should use RSA (or CS). If you can rule out active attacks then the  $s^2 \text{ mod } n$  generator is preferable: On the one hand it is cryptographically strong (proven), so if you can break it you can factorize and therefore break RSA too. On the other hand, it is proven for the  $s^2 \text{ mod } n$  generator that a passive attacker does not gain any information about the plaintext. Such a proof does not exist for RSA either.

In regards to the computational complexity  $s^2 \text{ mod } n$  generator and RSA are comparable. That does not offer a good distinguishing criteria.

---

<sup>54</sup>for the cubical complexity of the exponentiation, the quadratic of the multiplication and the linear of the addition

### 3.7 DES: The most known system for symmetric concealation and authentication

There are no reasonable security arguments against the employment of RSA instead of GMR as a **digital signature system**. However, if you can break GMR, then you can factorize and can therefore break RSA. So you should prefer GMR over RSA where ever possible.

In regards to the computational complexity, it is unfortunately reversed: If a hash function that is just as efficient with GMR is used for RSA (cp. §3.6.4.2)(a more complex one would not result in better security), then both will have about the same complexity. RSA is only slightly better because it will not have to create and verify a reference tree. But it is possible to use a much more efficient hash function for RSA because it *never* needs to be inverted. In GMR this is not necessarily needed for N-Signatures but inevitable for R- and K-Signatures (cp. fig. 3.27). The complexity for creation and verification of the R- and K-Signatures in GMR cannot be reduced in the same manner.

The application of a hash function that cannot be inverted by anybody can boost the efficiency, but requires further cryptographic assumptions, namely, that the hash function must be collision resistant. The more unproven assumptions are needed, the bigger the probability that at least one is not fulfilled. That would make security a mere illusion. In regards to digital signature systems, you have to decide between security (GMR) and efficiency (RSA with a fast hash function).

RSA was only patented for the USA and nowhere else. The patent ran out September 20th, 2000 [[Schn\\_96](#), page 474].

## 3.7 DES: The most known system for symmetric concealation and authentication

DES is the first standardized (hence the name: Data Encryption Standard) and (therefore) the most used cryptographic system. DES is a symmetrical cryptographic system that only has a 56 bit long key and encodes blocks of 64 bit length.

DES was published first in [[DES\\_77](#)]. It is printed in [[MeMa\\_82](#)].

### 3.7.1 Overview of DES

The DES algorithm is composed of a - cryptographically insignificant - **initial permutation IP** which separates the input block into the two 32 bit blocks  $L_0$  and  $R_0$ , **16 iteration rounds** which perform the actual encryption, and a **inversive permutation  $IP^{-1}$**  which is the inverse to the initial permutation. Before the inversive permutation is applied,  $L_{16}$  and  $R_{16}$ , the result of iteration round 16, are swapped.

The key is used to create 16 **partial keys**  $K_1$  to  $K_{16}$  - one for every iteration round.

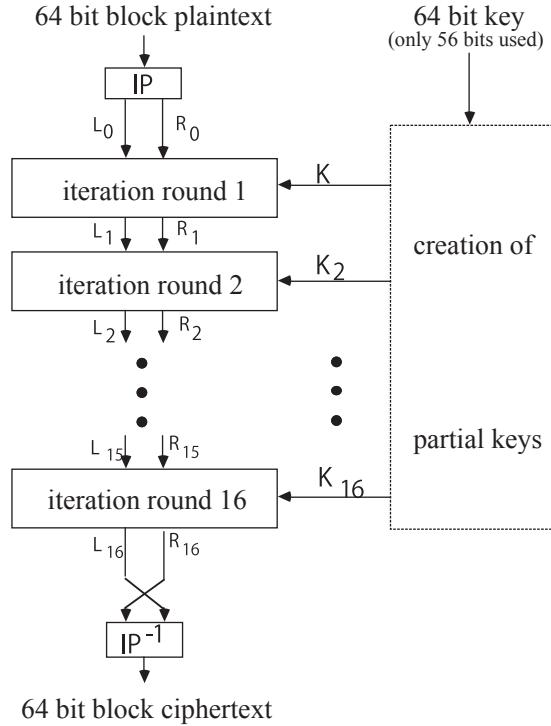


Figure 3.33: Sequence of steps in DES

### 3.7.2 An iteration round

Every iteration round  $i$  (cp. fig. 3.34) obtains the blocks  $L_{i-1}$  and  $R_{i-1}$  as input and provides the blocks  $L_i$  and  $R_i$  as output. Thereby  $L_i$  is directly copied from  $R_{i-1}$  and  $R_i$  is created by the addition (bit by bit modulo 2) of  $L_{i-1}$  and  $f(R_{i-1}, K_i)$ :

$$L_i := R_{i-1} \quad (1)$$

$$R_i := L_{i-1} \oplus f(R_{i-1}, K_i) \quad (2)$$

$f$  marks the **encryption function**.

### 3.7.3 The decryption principle of DES

Figure 3.34 shows how an iteration round of DES works. It is vital for the decryption in DES that this process be invertible. As a result of the iteration round, you will get:

$$L_i := R_{i-1} \quad (1)$$

$$R_i := L_{i-1} \oplus f(R_{i-1}, K_i) \quad (2)$$

Now you swap both of blocks  $L_i$  and  $R_i$ . This is done (fig. 3.35) by placing the identifiers of the values of the left side in the corresponding places on the right side, although according to the rule L=left and R=right, all identifiers on the right side L and R

### 3.7 DES: The most known system for symmetric concealation and authentication

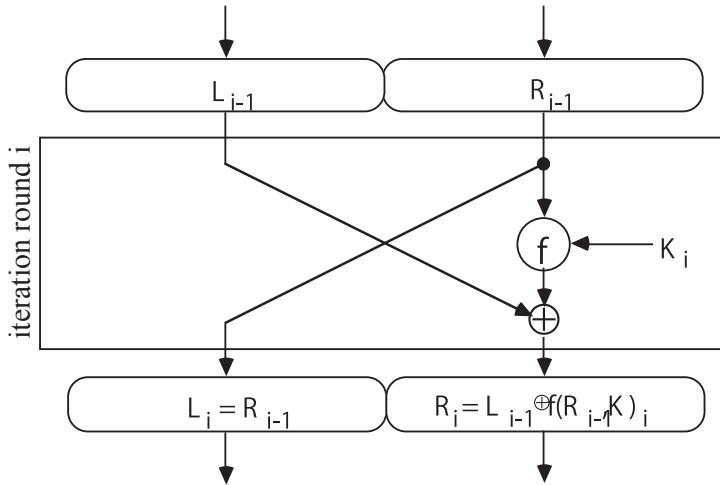


Figure 3.34: One iteration round of DES

should have been swapped. Besides, the index on the right side is counted backwards instead of forward. If you perform the same operation again:

$$L_i := R_{i-1} \quad (1)$$

and

$$L_{i-1} \oplus f(R_{i-1}, K_i) \oplus f(L_j, K_j) = L_{i-1} \oplus f(L_i, K_i) \oplus f(L_i, K_i) =: L_{i-1} \quad (2)$$

The necessary swapping of the blocks is done after the 16th iteration round. So the swapping is done before the beginning of the decryption. It is kept throughout every encryption round and is reversed with a further swapping after the last encryption round. Thus you can en- and decrypt data with a single algorithm. You just have to invert the order of the iteration rounds which can be achieved by inverting the order of the partial keys.

As you can see, the decryption principle is independent from the length of the blocks and the choice of the initial permutation IP (so is  $IP^{-1}$ ). Also, the encryption function  $f$  can be chosen without restriction.

55

Even though the choice of the encryption function  $f$  is irrelevant in regards to the invertability (with knowledge about the key). But essential for the security (i.e., uninvertability) for anybody else who does not know the key.

---

<sup>55</sup>This means, with the identifiers being introduced in the next section, that for the expansion permutation  $E$ , the substitution boxes  $S_i$ , ( $i = 1, \dots, 8$ ), and the permutation  $P$  you may use self-defined tables (by sticking to a basic structure). The creation of the partial key and the type of linking of the partial keys with the block  $R_{i-1}$  can also be very variable. Additionally it should be mentioned that for decryption,  $f$  will not need to be inverted.

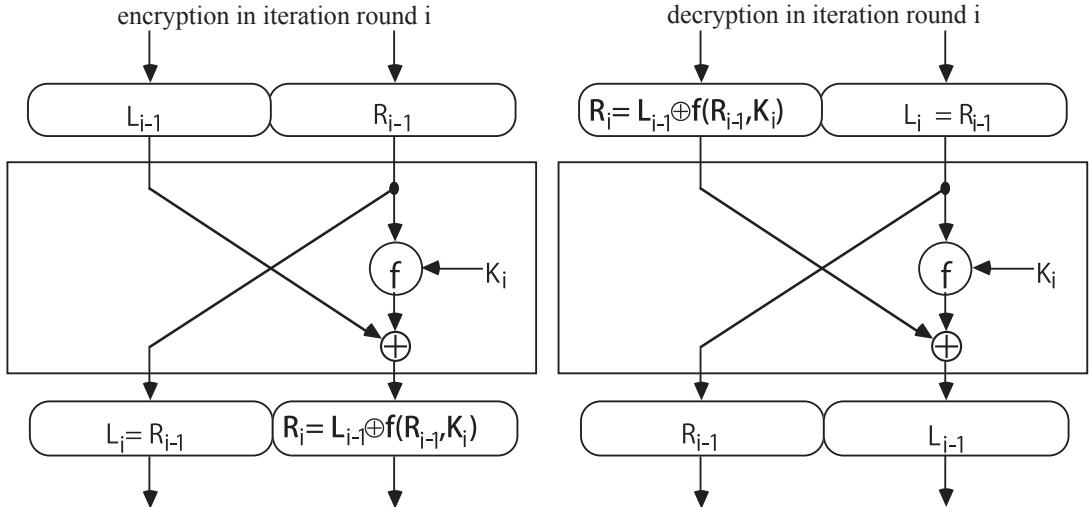


Figure 3.35: Decryption principle of DES

### 3.7.4 The encryption function

The first parameter of the encryption function shown in Figure 3.36, a 32-bit block, is expanded to 48 bit with the **expansion permutation**  $E$  (a permutation swaps single bits without changing their value or their number; in contrast, with the expansion permutation some input bits appear more than once in the output).

Next the second parameter, a 48 bit block (one of the partial keys), is **added** (bit by bit modulo 2).

The sum is split into eight 6 bit blocks with each selecting a 4 bit value from one of the **substitution boxes**  $S_j$ , ( $j = 1,..8$ ). (The substitution boxes do not work bit by bit but they assign values of bit blocks to values of bit blocks. This creates non-linearity, while permutations and adding modulo 2 cause an inversive output bit if the input bit was inverted. If DES were to be linear, a plaintext ciphertext attack could determine the value of the key bit for bit (cp. exercise 3-21f)).

The eight 4 bit values are combined into a 32 bit block. The **permutation**  $P$  is applied on this, through which one gets the function result.  $P$  scrambles the output of the substitution boxes so that in the next iteration round the output of every substitution box becomes a partial input into many other substitution boxes. This is how 8 little substitution boxes each with  $2^6$  entries and 4 bits each (i.e.,  $8 \cdot 2^6 \cdot 4$  bit =  $2^{16}$  byte memory consumption) become as big as a single substitution box with  $2^{48}$  entries each with 32 bit. This would consume  $2^{48} \cdot 32$  bit =  $2^{50}$  byte.

### 3.7 DES: The most known system for symmetric concealation and authentication

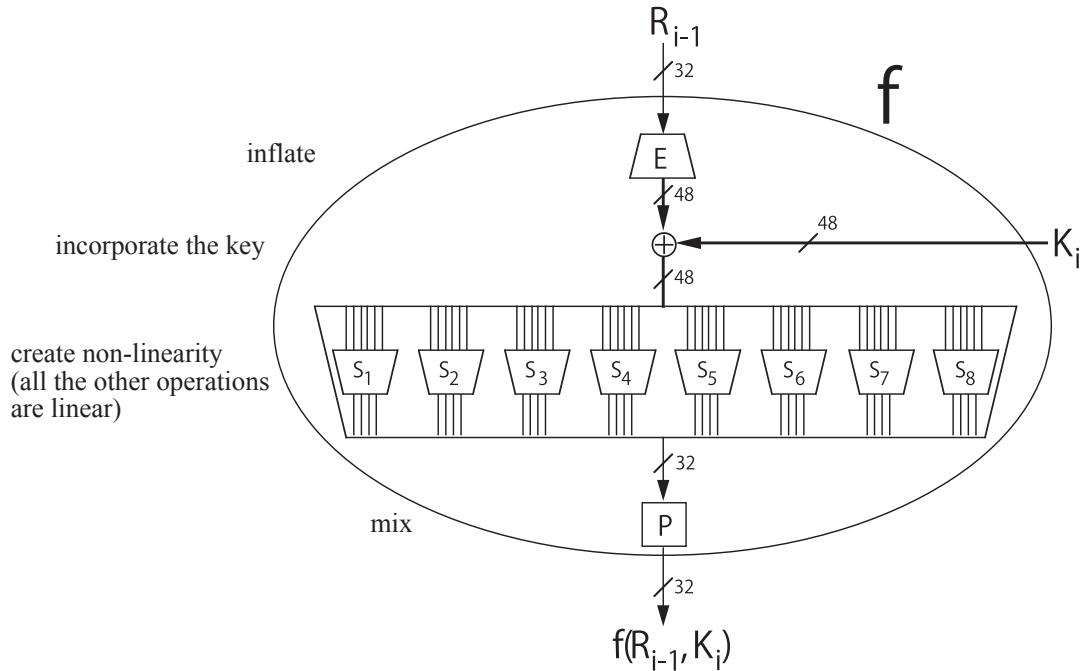


Figure 3.36: Encryption function  $f$  of DES

#### 3.7.5 Partial key generation

As seen in Figure 3.37, 56 bits are picked from the 64 bit key through the permuted choice function PC-1 and split into two 28 bit blocks  $C_0$  and  $D_0$ . You get  $C_i$  and  $D_i$ , ( $i = 1, \dots, 16$ ) by rotating the bits leftward (cyclic left shift  $LS_i$ ) for 1 or 2 bits (depending on  $i$ ). The partial keys  $K_i$  are generated by concatenating  $C_i$  and  $D_i$  and picking 48 bits with PC-2.

With DES the encryption rotates 2 bits left in the 12th and 16th iteration round and 1 bit in all the other four iteration rounds. In total, a 28 bit rotation makes the final state equal to the beginning state ( $C_0 = C_{16}$  and  $D_0 = D_{16}$ ). Consequently you have the opportunity to generate the partial keys for the decryption through the corresponding (i.e., the flipped order) right-rotation of the amount of bits.

#### 3.7.6 The complementary property of DES

Inside the encryption function and the partial key generation, DES has a single and simple regularity: The so-called **complementary property**:

$$\text{DES}(\bar{k}, \bar{x}) = \overline{\text{DES}(k, x)}$$

If plaintext block and key are complemented bit by bit (every 0 is replaced by 1 and the other way round) then the complementarity cipher text arises because

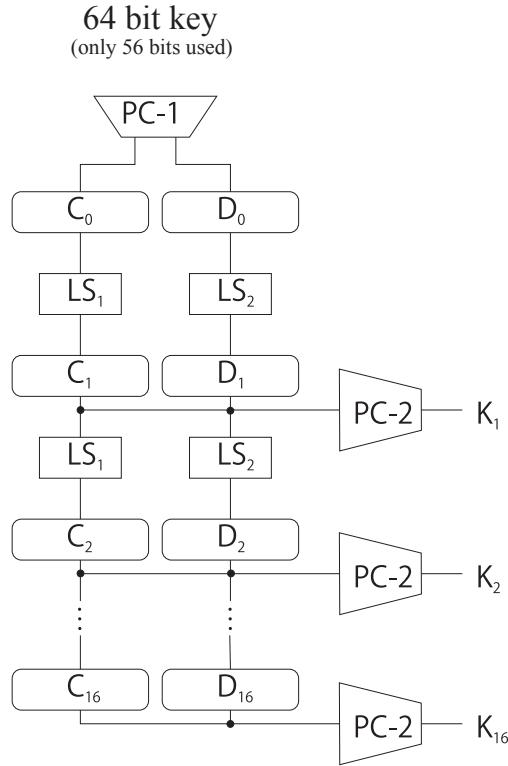


Figure 3.37: Generation of partial keys with DES

- permutations, especially PC-1, PC-2, IP and  $IP^{-1}$ , receive the complementarity
- rotations receive the complementarity so that at complementarity keys the partial key are also complementary
- every iteration round receives complementarity because after the expansion permutations  $E$  the complementarity partial key is added (modulo 2) to the complementary value. This voids the complementarity before the substitution (because  $0 \oplus 0 = 1 \oplus 1$  and  $1 \oplus 0 = 0 \oplus 1$ ). But it is recovered after the substitution by the addition modulo 2 of  $L_{i-1}$  and because the copying of  $R_{i-1}$  to  $L_i$  contains the complementarity anyway,
- swapping of the left and right side contains the complementarity.

Figures 3.38 and 3.39 illustrate the gaining of the complementarity in every iteration round.

This regularity can be used to nearly halve the complexity of the search through the key room: If you have the corresponding cipher text blocks for two complementary plaintext blocks (e.g., as a result of a chosen plaintext ciphertext attack) then one of the plaintext blocks is encrypted with other keys for as long as it takes to find the

### 3.7 DES: The most known system for symmetric concealation and authentication

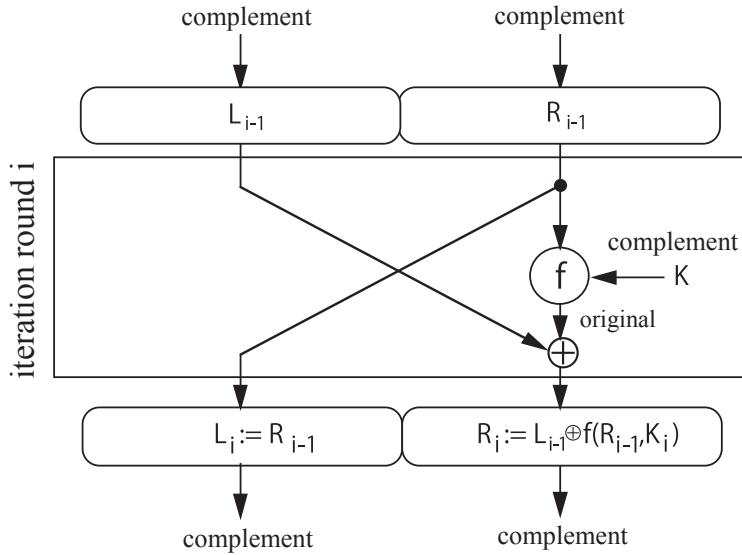


Figure 3.38: Complementarity in every iteration round of DES

corresponding ciphertext block or the complement of the other key. In the first case the key was found. In the second case the complement of the key was found because the complement of the plain text block encrypted with the complement results in the complement of the ciphertext.

In both cases the situation may occur (with a very small probability) that for one plaintext ciphertext pair there would be more than one key. So the correctness must be checked for both cases on further plaintext ciphertext pairs. This holds - independent from the complementarity - for DES in general.

#### 3.7.7 Generalization of DES: G-DES

The security of DES has long been criticized for two reasons:

- The key with its 56 bit is too short for applications that require security against massive attacks (i.e., intelligence services):

DES was completely broken with a plaintext ciphertext attack in January 1998. During their idle time, 22000 computers, coordinated over the Internet, searched the whole key room. Within 39 days, 85% of all possible keys were verified until the right one was found [CZ\_98]. As an alternative to those software based attack, which are almost for free, the development of special machines is evaluated [Schn\_96, page 152-157]: With the technology available in 1995, the product of costs and time needed (in seconds) to find a DES key would be  $10^{10}$ . For  $10^5$  US\$ you could build a machine that would find the key within 35 hours, with  $10^6$

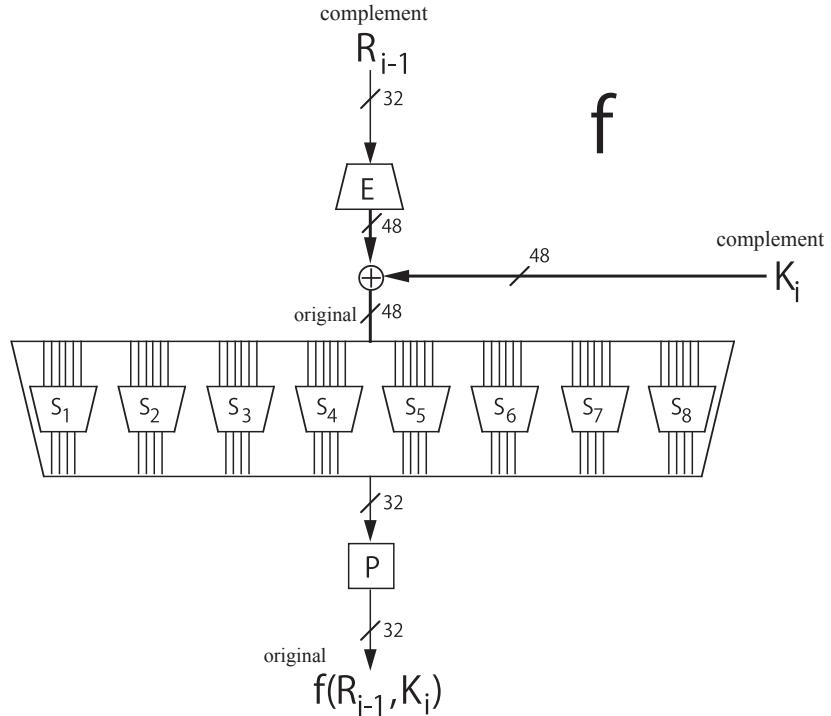


Figure 3.39: Complementarity in the encryption function  $f$  of DES

US\$ in 3.5 hours. In the past, computer power doubled every 18 month. So this product could be halved every 18 months and in year 2000 the factor would be  $10^9$ . In July 1998 the Electronic Frontier Foundation published that you could built a special machine for breaking DES in 3 days for less than 250,000 US\$ [EFF\_98]. By utilizing these upper bounds, the real product of  $250,000 \cdot 3 \cdot 24 \cdot 60 \cdot 60 = 6.48 \cdot 10^{10}$  is slightly above the value that was ascertained at the desk for the technology available in 1995.

- The design criteria for the specific values of the substitutions and the permutations (not mentioned here, but you may take a look into the standard) are still kept under wraps. But [Copp\_92][Copp2\_94] reveals the secret almost completely.

Under the cryptographic and implementations point of view, the following generalizations takes care of those criticisms:

1. The 48 key bits of each of the 16 partial keys is entered directly instead of being created by the 56 bit. The key then is  $16 \cdot 48 = 768$  bit long instead of 56 bit.
2. The content of the substitution boxes  $S_j (j = 1, \dots, 8)$  becomes variable. This enlarges the key by maximal  $8 \cdot 2^6 \cdot 4 = 2048$  bits.

3. The permutations P and IP become variable. This increases the key length by maximal  $32 \cdot [ld32] + 64 \cdot [ld64] = 160 + 384 = 544$  bits.
4. The expansion permutation becomes variable. This increases the key length by maximal  $48 \cdot [ld32] = 240$  bits.

This makes a total of 3600 bit maximal. In regards to the complexity and memory consumption, this is less today than the 56 bits in 1977. While for DES a random choice for the 1. generalization is recommended, this does not necessarily hold for the 2nd, 3rd, and 4th generalizations [BiSh\_93].

To prevent the emergence of false expectations about the security: While the “normal” DES can be broken with a chosen plaintext ciphertext attack (called differential cryptoanalysis) with  $2^{47}$  plaintext ciphertext blocks, the 1. generalization needs  $2^{61}$  [BiSh3\_91][BiSh4\_91][BiSh\_93]. Although far away from the complexity naively hoped for of  $2^{786} - 1$ , it is very close to the optimal complexity of  $2^{64} - 1$  bit: *Every* permutating block cipher with block length  $b$  can be broken with a complexity of  $2^b - 1$  - the attacker simply lets every block that he is not interested in being encrypted resp. decrypted.

### 3.7.8 Decryption performance

With a software implementation, an Apple Macintosh Classic (MC68000, 8 Mhz), with the application of the 1st, 2nd, and 3rd generalization of DES, can encrypt resp. decrypt at over 100 kbit/s and on a PowerBook 180 (MC 68030, 33 Mhz) at over 900 kbit/s [PfA81\_90][PfA8\_93].

With a hardware implementation (standard cell design, 7155 gates,  $2\mu\text{m}$  technology) on a DES, using the 1st generalization, you can encrypt resp. decrypt with 40 Mbit/s. Because the I/O is asycron (2 way handshake) and only works byte for byte, only 10 Mbit/s are created. [KFBG\_90].

## 3.8 Cryptographic systems in operation

To operate cryptographic systems it is useful to know about terms such as:

- block cipher vs. stream cipher, synchronous vs. self synchronizing
- operation modes of block ciphers, i.e., the important construction of self-synchronous and synchronous stream ciphers from block ciphers, and
- at least one efficient way to construct a collision-resistant hash function from block ciphers.

All these are covered in the following subsections.

### 3.8.1 Block Ciphers, Stream Ciphers

Given a finite alphabet

- a **block cipher** en-/decrypts only strings of *fixed* length, while
- a **stream cipher** can work with strings of *variable* length.

If we take an alphabet  $\{0, 1\}$  as a basis; the Vernam-cipher (one-time pad), the  $s^2$ -mod- $n$ -generator, and GMR are stream ciphers, while RSA and DES would be block ciphers. But if we use an alphabet  $\{0, 1, 2, 3, \dots, 2^{64}-1\}$  and assume that several symbols become encrypted independently from each other, then DES would be a stream cipher according to the above definition. The given distinction between stream and block ciphers depends substantially on the used alphabet and is therefore relative in itself. For many applications the alphabet is specified (or at least canonical) and leads to a precise distinction.

*Annotation* Some authors define a more restrictive classification of stream ciphers: For them a stream cipher has to have a given (one-time pad) or generated ( $s^2$ -mod- $n$ -generator) key stream which becomes added (or subtracted) to the plaintext or the ciphertext. In our case, this restrictive definition seems unnecessary to me.

For stream ciphers [Hors\_85, HeKW\_85], messages are encoded as a sequence of *symbols* to encrypt single symbols of the alphabet. These symbols do not necessarily become encrypted independently, but their encryption can depend:

1. on their position inside the message or more generally from all previous plain- and/or ciphertext symbols or
2. only from a certain number of ciphertext symbols directly in front of the current symbol

Ciphers matching the first case are called ***synchronous stream ciphers***, because en- and decryption has to be performed strictly synchronously. If a symbol gets lost or becomes added to the stream (loss of synchronization), the decryption cannot take place without further manipulation. The communication partners have to synchronize afresh. As long as the de- and encryption not only depends on the position within the message, but also on all previous plain and ciphertext symbols, the encrypting and decrypting party have to synchronize afresh even for simple cipher symbols changes.

In the second case we talk about ***self synchronizing stream ciphers***, because no matter how many symbols become interfered with, at the latest after a limited number of symbols, the decoder has re-synchronized itself with the encoder.

*Annotation* According to the above definition each stream cipher is either synchronous or self synchronizing.<sup>56</sup> Some authors give a more restrictive definition of synchronous, e. g. [Denn\_82, page 136]: The key stream (cf. definition above) is generated independently from the plaintext stream. Now there exist stream ciphers which are neither synchronous nor self synchronizing, such as *output cipher feedback*, see picture 3.50. In my point of view this fine distinction into 3 categories is not necessary for what follows.

The following applies for each symmetric or asymmetric deterministic stream cipher and arbitrary nonempty strings  $x_1, x_2$  and pairs of keys  $(c, d)$  respectively key  $k$ :

If two strings become encrypted separately, but directly after each other, the overall result will be the same as if the two strings had first been concatenated and then encrypted. Using the introduced notation and given a collateral (i. e. parallel, each other not influencing) evaluation on both sides from the left, we obtain:

$$k(x_1), k(x_2) = k(x_1, x_2) \text{ or } c(x_1), c(x_2) = c(x_1, x_2)$$

If other strings become encrypted in between the two separately encrypted strings or the concatenated string becomes encrypted as third in a row after the two separate others (so not collateral), it is most likely that the given equations will not be valid.

### 3.8.2 Operation modes of block ciphers

To point out the connections between stream and block ciphers, and due to their importance for practical usage<sup>57</sup>, we will specify in the following, several partly standardized and important modes of block ciphers (namely: the construction of self synchronizing or synchronous stream ciphers from block ciphers). Even if we believe today that the usage of a secure block cipher for the construction of stream ciphers usually implies their security, in my opinion two restrictive annotations have to be made.

1. Even for strong definitions of what is secure about a block cipher, I do not know any (reduction) proofs for the constructed stream ciphers security nor do I know any qualifications for what “usually” means.
2. For at least some of these constructions, pathological examples to prove the opposite have been found. I will describe these in the following. It is up to the reader to assess these systems’ “security”.

Next we will describe the standardized operation modes ECB, CBC, CFB, and OFB first, and then the two generalizations PCBC and OFCB.

---

<sup>56</sup>To allow character-oriented encryption in the second case we restrict the dependency on predecessor symbols to 0. Doing this also covers the example where DES is used as stream cipher on the alphabet  $\{0, 1, 2, 3, \dots, 2^{64} - 1\}$ .

<sup>57</sup>Up to this point we did not discuss if DES can be used to encrypt messages which are not of the exact length of 64 Bit. Likewise it is unclear if DES can be used for authentication.

### 3 Cryptography Basics

From each symmetric or asymmetric deterministic block cipher, self synchronizing stream ciphers can be designed with the following constructions:

- *electronic code book*, ECB [DaPr\_89]
- *cipher block chaining*, CBC [DaPr\_89]
- *cipher feedback*, CFB [DaPr\_89]

At this, the alphabet underlying the terms block and stream cipher for the modes ECB and CBC becomes changed – for CFB it can be changed.

From each symmetric or asymmetric deterministic block cipher, synchronous stream ciphers can be designed through the following constructions:

- *output feedback*, short OFB [DaPr\_89]
- *plain cipher block chaining*, short PCBC, in literature also referred to as plaintext-ciphertext feedback or block chaining using plaintext and ciphertext feedback [EMMT\_78, page 110, 111] and [MeMa\_82, page 69]
- *output cipher feedback*, short OCFB

With this, the alphabet underlying the terms block and stream cipher for the mode PCBC becomes changed – for OFB and OCFB it can be changed.

Each of the 6 constructions is discussed in a separate sub-chapter.

These are followed by sub-chapters with annotations to expanding block ciphers (in the previous sub-chapters only static length block ciphers were discussed), memory initialization, and an overview of the essential properties of all described constructions.

#### 3.8.2.1 Electronic code book (ECB)

The simplest operation mode of block ciphers is to split long messages into blocks, so that each block can be en- and decrypted *independently* from all others, see picture 3.40. This mode is called **electronic code book** (ECB for short).

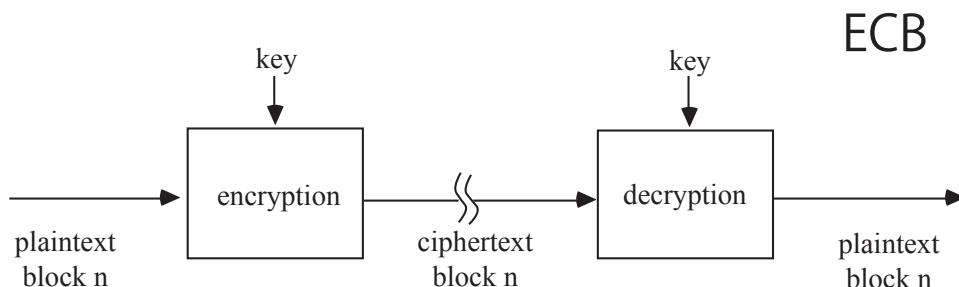
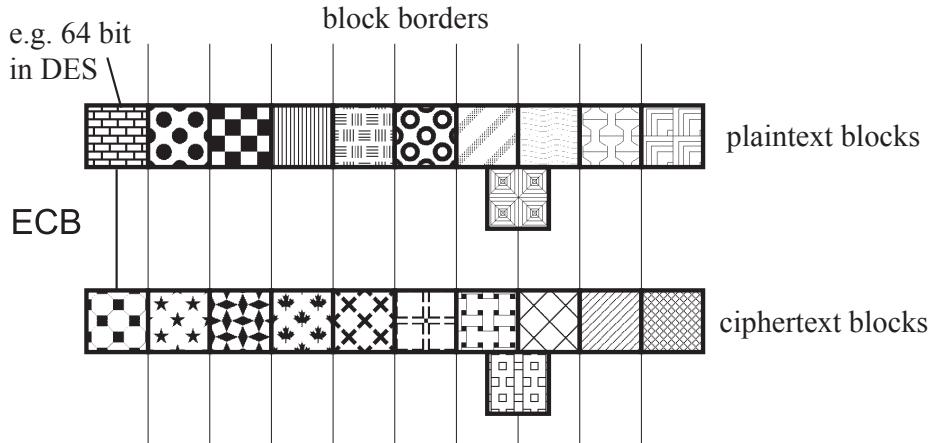


Figure 3.40: Operation mode “electronic code book”

Their properties are canonical and therefore only tabulated briefly in picture 3.51. A disadvantage of deterministic block cipher is the direct mapping of plaintext patterns onto the ciphertext for patterns filling a whole block<sup>58</sup>, see picture 3.41. Even for indeterministic block ciphers, patterns in the ciphertext become mapped to adequate patterns in the plaintext. In reality this will almost never happen, whereas regular patterns in uncompressed plaintext will appear frequently and have to be considered.<sup>59</sup>



With ECB identical plaintext blocks are mapped to identical ciphertext blocks. Patterns which overlap block borders in general map to different ciphertext patterns. This example shows that for the blocks below the block sequences.

Figure 3.41: Block patterns with ECB

<sup>58</sup>For non context depending encryption (steps are not depending on previous ones) of semantic units, there exist (in a certain sense) the possibility of breaking the code through learning: The attacker observes a message (ciphertext) and the happenings in the real world. If the same message is observed again, we could assume that the same will happen as it did last time. Here a short example: An admiral of a fleet gives commands to all ships such as: “turn to starboard”, “full speed”, “stop engine”, etc. If all these commands become encrypted deterministically and mutually independent without any connection with the previous events, i. e. without serial numbers, time stamps, etc., then each command corresponds to exactly one ciphertext. After a little while, the attacker has learned the meaning of the ciphertexts regardless of how secure the block cipher might have been or if we work with a symmetric or asymmetric system. To impede such non-cryptographic breaking through learning, semantic units have to become encrypted in a context sensitive, or even better, indeterministic way.

<sup>59</sup>Let us assume a telefax becomes encrypted using a 64-Bit block cipher in ECB mode. We assume that the most common color of pixels will be white. Now we assign to the most frequent cipher-text-block 64 white pixels and to all the others 64 black pixels. Let us also assume that our fax has a resolution of 300 dpi, circa 12 pixel per mm. The encryption using a 64-Bit-ECB decreases the horizontal resolution from 0.08 to 5.3 mm – at least big fonts will be readable without issues. If the pixels are not encoded horizontal, but in squares of 8x8 pixels instead, the resolution changes only from 0.08 to 0.67 mm. Now even small fonts should be legible.

### 3.8.2.2 Cipher block chaining (CBC)

Cipher block chaining is illustrated in picture 3.42: Before the encryption of a block takes place (except the first one), the ciphertext of the previous block becomes modularly added to the plaintext and adequately subtracted before decryption.

the number of characters carried on all lines correspond to the block length

$\oplus$  addition, e.g. bit by bit modulo 2

$\ominus$  subtraction, e.g. bit by bit modulo 2

CBC

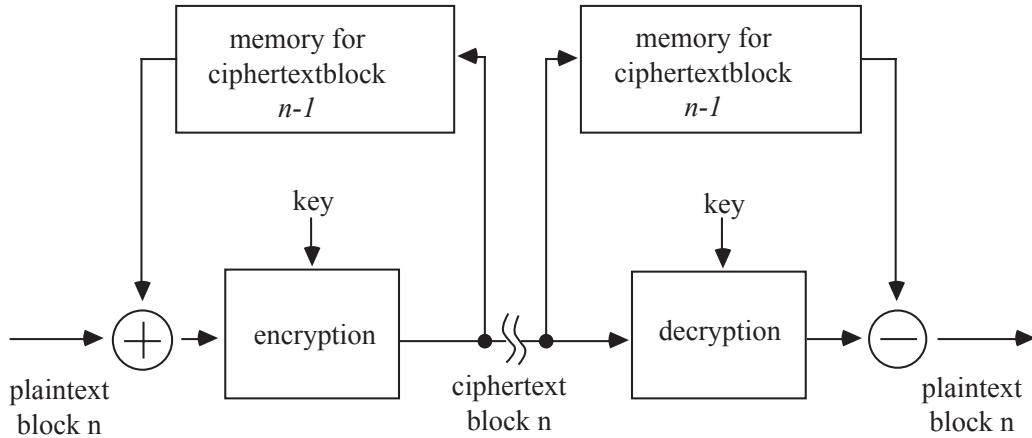


Figure 3.42: Construction of symmetrical resp. asymmetrical self-synchronizing stream cipher from symmetrical resp. asymmetrical block cipher: block cipher with block chaining

This design can be used for encryption as well as for authentication. The following advantages and disadvantages, or ambivalent properties, come along with encryption:

- + The usage of an *indeterministic block cipher* is possible. A suitable selection has to be made for summation and subtraction, because for indeterministic block ciphers the block length of the ciphertext is longer than that of the plaintext.
- + If you use an *asymmetric block cipher*, the resulting stream cipher is *asymmetric*
- The length of the encodable units is limited by the *block length* of the used block cipher. These units cannot become mapped onto the storing and transmission units. Special block boundaries have to be defined to make self-synchronization possible.
- \* Even for the slightest corruption in a unit of the adequate cipher text block, (or for transient errors in en- or decoder) all the following plaintext characters of this unit will, with a probability of

$$\frac{|\text{alphabet}| - 1}{|\text{alphabet}|}$$

be interfered with. Otherwise the block cipher is useless. Additionally, the corresponding place in the following plaintext block is interfered with. This is because the interference becomes stored in the ciphertext and is used in the next round for the second time (but in a different way). Later on plaintext blocks are not influenced and therefore decrypted correctly. Due to its limited memory depth, the decoder synchronizes itself automatically with the encoder.

This is also the case if it is not a transmission error outside the encryption process but a transient error during encryption. This is remarkable, because when starting at the location of the error a completely different cipher text is generated. The error is passed from one cipher block to the next one and becomes added to the next plaintext block which results in an interfered input to the block cipher and a completely different ciphertext block. Nevertheless, the decoder obtains the correct plaintext already one block after the end of the transient error. The same applies to transient errors in the decoder. Both for transient errors in the encoder and in the decoder, we assumed in the above examples that the keys of the block cipher are not changed by the transient error. In this case the plaintext obtained by the decoder from this point would be pseudo-random and therefore useless.

Because it is irrelevant for the result of the addition in picture 3.42 if the plaintext or the ciphertext becomes changed, the same (as described before) applies for the slightest changes in the plaintext. This gives the motivation to use the left part of the **construction for authentication**(picture 3.42):

- + The one who wants to authenticate encrypts the plaintext and sends the last block obtained together with the message to the other one who can now test if the resulting block he got after encryption equals the one sent with the message. If they equal each other, the message is authentic. If you want to have shorter authenticators, a part of the last ciphertext block can be sent.
- \* The used block cipher has to be *deterministic*.
- The length of the units to authenticate is limited by the *block length* of the used block cipher.

This mode for authentication is standardized in [ISO8731-1\_87]. Here it is specified that only the leftmost 32 bit should be used as authenticator and that incomplete plaintext blocks should be filled with binary zeros.

As described so far, the attacker is able to detect if two plaintexts start with the same blocks: the cipher blocks then also start the same.

This can be avoided by concatenating the real plaintext onto a randomly chosen one<sup>60</sup>: We just saw when doing this, that all following ciphertext blocks came out differently each time. To make it more efficient, it is possible to choose the ciphertext instead

---

<sup>60</sup>This makes the encryption indeterministic. We used a similar approach for asymmetric encryption in §3.1.1 annotation 2 and §3.4.4

### CBC for authentication

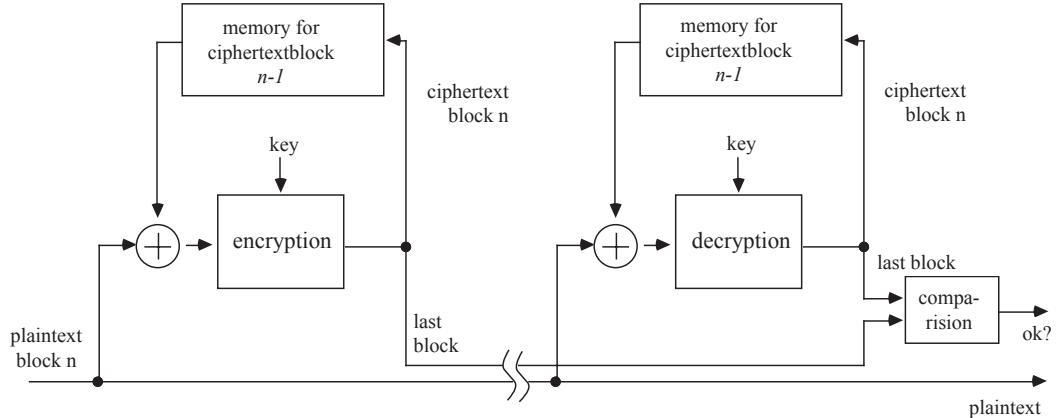


Figure 3.43: Block cipher with block chaining used for authentication: Constructed using a deterministic block cipher

of encrypting a randomly chosen plaintext. This ciphertext block must certainly made known to the decoder, i. e. as the first ciphertext block where the adequate plaintext block is a random number too and is therefore thrown away by the decoder.

**A pathological example to prove the opposite** for encryption using *cipher block chaining*: addition and subtraction are performed modulo 2. The block cipher encrypts even blocks (last bit 0) securely to uneven ones (last bit 1) and uneven blocks insecurely to even ones, see picture 3.44. For restricting the plaintext domain to even blocks, i. e. view our block cipher as a block cipher expanded by one bit (see picture 3.45), we are dealing with a secure block cipher.

Nevertheless, the resulting stream cipher is insecure even for the restricted domain: For the last bit of the block  $0 \oplus 1 = 1$  applies, encryption results in 0,  $0 \oplus 0 = 0$ , encryption results in 1, etc., every second input block for the block cipher is uneven. The attacker is now able to calculate these uneven input blocks from the even ciphertext blocks and the adequate plaintext blocks by subtraction of the previously observed cipher text blocks. The stream cipher is insecure in relation to every second plaintext block and therefore altogether insecure.

#### 3.8.2.3 Cipher Feedback (CFB)

Cipher feedback is shown in picture 3.46: Not the plaintext, but the content of a shift register becomes encrypted by the block cipher and a part of the result becomes added modular to the plaintext (which gives the ciphertext) and accordingly it becomes subtracted from the ciphertext by the decoder to obtain the plaintext. The ciphertext becomes shifted into the shift register (feedback), where the name *cipher feedback* comes from. If the shift registers become initialized right away and the ciphertext is transmit-

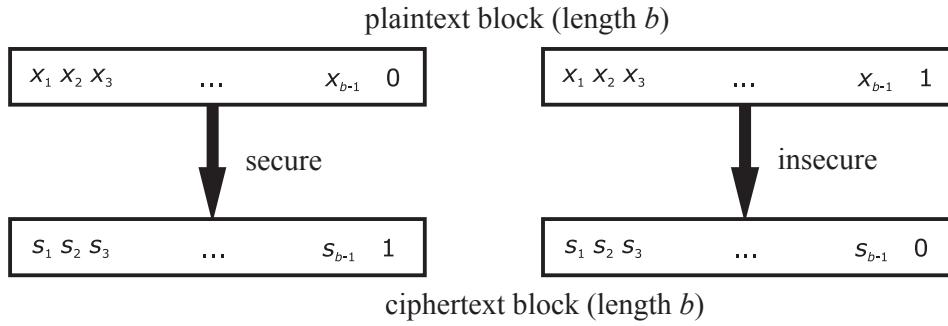


Figure 3.44: Pathological block cipher (only secure with plaintext blocks, which contain a 0 on the right-hand side)

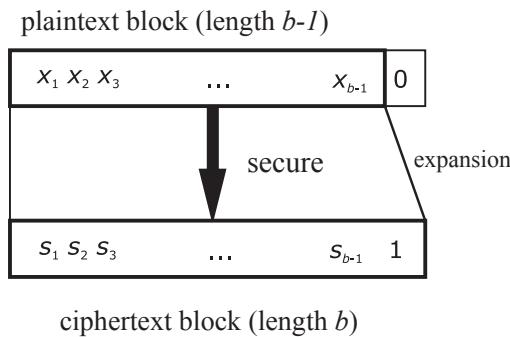


Figure 3.45: Block cipher expanding by one bit

ted accurately, then the shift register will get the same values from the en- and decoder. In this case (and only in this case), the decoder returns the original plaintext.

In picture 3.46 the most common case is illustrated: The ciphertext becomes not directly shifted into the register, rather a selection is made or even fixed values are added. Though the latter one is specified in [DINISO8372.87], I feel it is not useful in view of the cryptographic security, because the number of possible values in the shift register becomes limited. Due to the fact that the function “select or add” has to be specified publicly, a possible attacker would be able to detect equal values in the register. Through calculating the difference between the two ciphertexts he would obtain the difference between two plaintexts. The selection is especially problematic for authentication<sup>61</sup>, because single characters of the ciphertext stream (and the plaintext stream) can be altered without influencing the register content and the further encryption process. With this, changes to the plain or ciphertext do not have an influence on the authenticator and do not become detected.

---

<sup>61</sup>therefore the function “select or add” was left out in picture 3.47 – because a reduction of the number of possible values in the shift register is not useful

### 3 Cryptography Basics

Cipher feedback has the following advantages and disadvantages, or ambivalent properties, for cipher block chaining for encryption:

- + *Smaller units* than the one defined through the block length can be en- and decrypted. If the elemental unit of the storing or transmission system is used as encryption unit, self-synchronization is given without a special labeling of the “block boundaries”.
- The used block cipher has to be *deterministic*.
- Regardless of whether or not a symmetric or an asymmetric block cipher is used in this mode, the obtained stream cipher is always *symmetric*, because the different decryption function of the asymmetric stream cipher is not used in this construction.
- \* Even for the slightest failures in a unit of the ciphertext stream (or even transient errors in the decoder), all characters of the plaintext are influenced in the current and the following

$$\lceil \frac{\text{blocklength} - \delta}{\text{length of the feedback unit}} \rceil$$

feedback units ( $\delta$  denotes the distance of the error from the right border of the feedback unit;  $\lceil x \rceil$  denotes the smallest integer  $z$  with  $z \geq x$ ). The first faulty unit is at the point where the error occurred. For latter ones, all characters are interfered with a probability of

$$\frac{|\text{alphabet}| - 1}{|\text{alphabet}|}$$

According to the reasons given for CBC, the left part of CFB can be used for **authentication**, see picture [3.47](#):

- + The authenticating party encrypts the plaintext with CFB. Additionally the content of the shift register becomes encrypted using a block cipher<sup>62</sup> and sent together with the plaintext to the verifying party. Here it becomes processed in the same way and the results are compared. If they match, the plaintext is authentic. Assuming a shorter authenticator would be sufficient, it is possible to use only a part of the output for it.
- \* The used block cipher has to be *deterministic*.
- Units smaller than the block length of the block cipher can become authenticated.

---

<sup>62</sup>If the content of the shift register is not encrypted separately with the block cipher (is directly used as an authenticator), the attacker could change the output unit undetected by addition of the difference between falsified and original unit to the authenticator.

- b block length
- a length of the output unit,  $a \leq b$
- r length of the feedback unit,  $r \leq b$
- $\oplus$  addition relative to suitably chosen module
- $\ominus$  subtraction relative to suitably chosen module

CFB

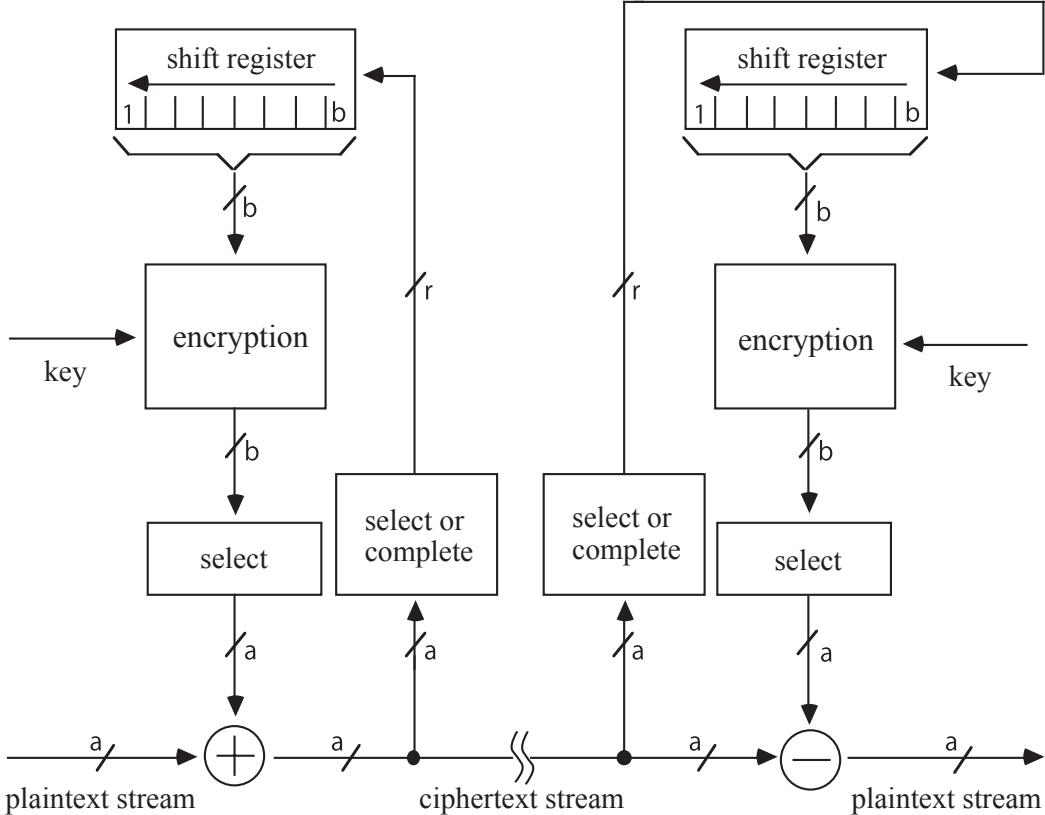


Figure 3.46: Construction of a symmetrical self-synchronizing stream cipher from a deterministic block cipher: ciphertext feedback

### 3.8.2.4 Output feedback (OFB)

Output feedback is shown in picture 3.48. In contrast to cipher feedback, not the ciphertext, but the output of the block cipher becomes passed back into the shift register. Accordingly, this is called *output feedback* (short OFB).<sup>63</sup>

As already shown in picture 3.46, the general case is specified in picture 3.48. The

<sup>63</sup>Described in a different way in the upper part of the construction, a *pseudo-random generator* using the block cipher generates a pseudo-random number. This number is added to the plaintext stream or subtracted from the ciphertext stream in the lower part of the construction. In §3.4.2 we called this a one-time pad

### 3 Cryptography Basics

- b block length
- a length of the output unit,  $a \leq b$
- $\oplus$  addition relative to suitably choosen module

### CFB for authentication

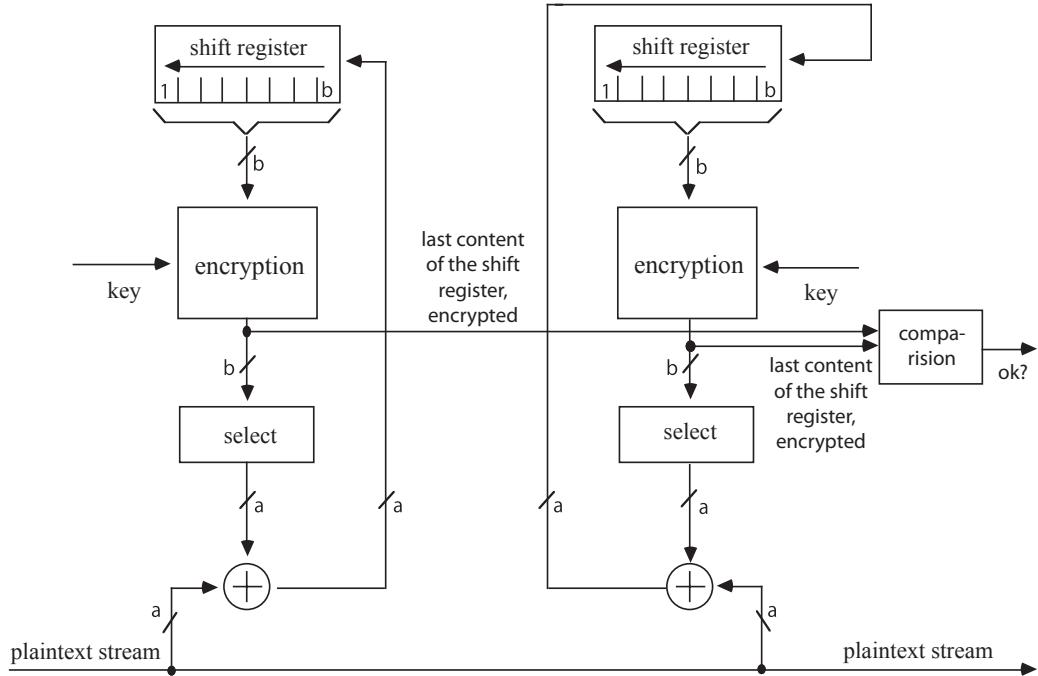


Figure 3.47: Cipher feedback for authentication: Construction using a deterministic block cipher

block encryption is subject to selection or completion by defined values. Completion is possible, but seems not to be useful from a cryptographic point of view. The number of possible values in the shift register becomes reduced, which means the number of elements the random number generator can generate before producing repetitions, decreases. If the function “select or complete” is the identity as described in [DaPa.83] and intended in [DINISO8372\_87], a simple memory can be used instead of an shift register.

Output feedback has the following advantages and disadvantages, or ambivalent, properties:

- + The length of the en- and decryptable units is not restricted by the block length of the used block cipher. This simplifies the coordination with regards to the units of the transmission and storing system.
- The used block cipher has to be *deterministic*.
- Independent from the usage of a symmetric or asymmetric block cipher a *symmetric* stream cipher is obtained. The decryption function, which is different to from encryption in asymmetric cryptography, is not used for the construction.

- \* Failures in a single character of the ciphertext cause single false plaintext character. No error extension occurs [DaPr\_89]. Depending on the application this can be advantageous (e. g. for encryption) or disadvantageous (e. g. for integrity). To prevent the reader from getting a wrong impression, a general property of synchronous stream ciphers (therefore for CFB) must be mentioned here: Only when alterations are made to ciphertext characters do failure extension not occur. Lost or added cipher stream characters cause false plaintext characters with a probability of

$$\frac{|\text{alphabet}| - 1}{|\text{alphabet}|}$$

till the system re-synchronizes. The same applies to transient failures in the feed-back of the en- or decoder.

**Pathological example to prove the opposite** for output feedback: No selection or addition takes place. The block cipher (encryption in picture 3.48) maps *plaintext blocks* onto *ciphertext blocks* of the same length  $b$  and is defined as follows: Choose a plaintext block  $K$ , which has no ciphertext block assigned to itself and assign one free ciphertext block  $S$  to it randomly. Now assign to the plaintext block  $S$  the ciphertext block  $K$  as its encryption. This construction produces a secure block cipher, because the assignment of ciphertext blocks is “random”.

Note that the above mentioned plaintext blocks in picture 3.48 are the ones from the shift register and do not have anything to do with the message to be secured, which is referred to as a plaintext stream in picture 3.48.

Despite of the secure block cipher, the stream cipher resulting from output feedback is insecure. It was constructed in a way that the same blocks are always encrypted in alternating order and accordingly the same values are added to the plaintext stream with the same alternation.

### 3.8.2.5 Plain cipher block chaining (PCBC)

**Plain cipher block chaining (PCBC)**(cf. picture 3.49) can be used for applications requiring failure extension even for single changes in the cipher text stream with the above mentioned probabilities. For PCBC the blocks are not only dependent on the ciphertext of the previous block, but also the plain text of the previous block is taken into account. It is remarkable that for decryption, no inverse function to  $h$  is required to combine the previous plain and ciphertext block. To construct secure symmetric and asymmetric *synchronous* stream ciphers from secure symmetric and asymmetric block ciphers, an appropriate function  $h$  has to be chosen according to the used modulus for the operations addition and subtraction, cf. [MeMa\_82, page 69-71]. In particular, it is important not to choose for  $h$  simply the used operations addition and subtraction<sup>64</sup>. [MeMa\_82] recommends the selection used in the example in picture 3.49.

---

<sup>64</sup>The attack refers to authentication in PCBC mode as described below.

$h$  is defined as  $\oplus$ . The initial value  $IP$  for memory of the plaintext block  $n - 1$  as well as the expected value  $RP$  of the redundancy block at the end of the plaintext are not defined. There is just an agreement that they have to be the same. Now the attacker can make the receiver of

- b block length
  - a length of the output unit,  $a \leq b$
  - r length of the feedback unit,  $r \leq b$
  - $\oplus$  addition relative to suitably chosen module
  - $\ominus$  subtraction relative to suitably chosen module
- OFB

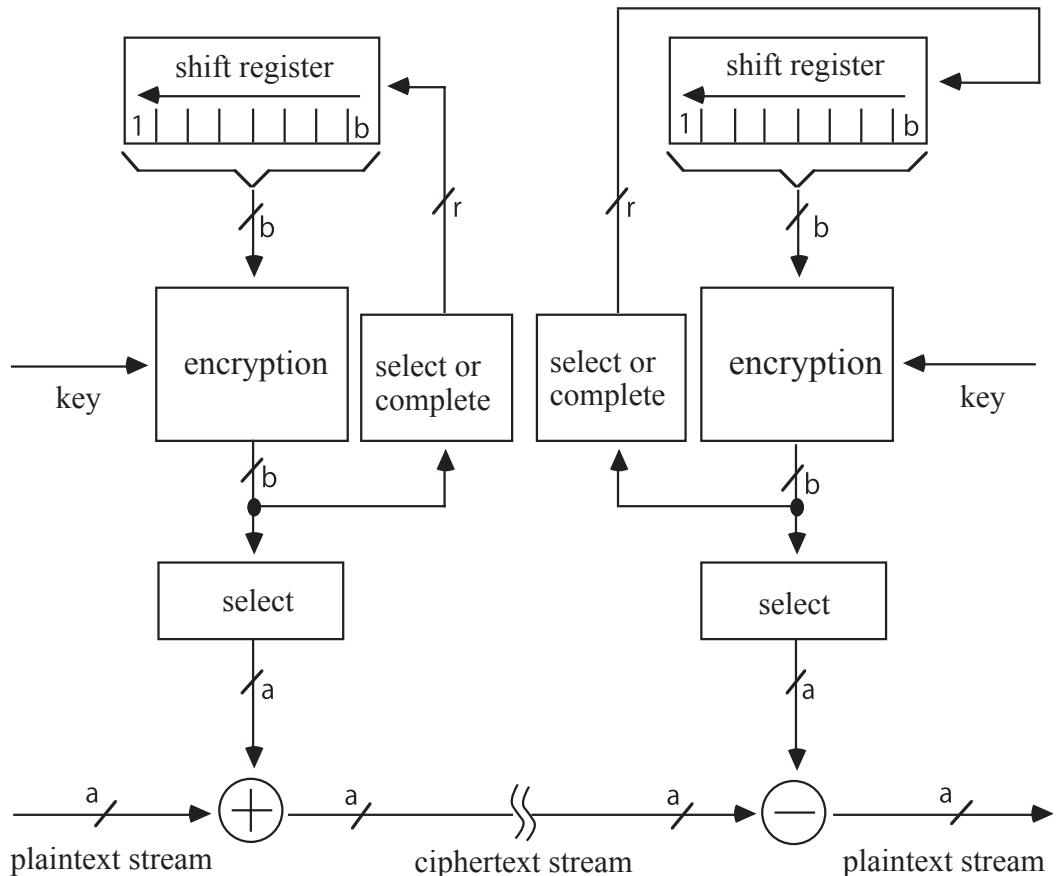


Figure 3.48: Construction of a symmetrical synchronous stream cipher from a deterministic block cipher: output feedback

This construction has the following advantages and disadvantages, or ambivalent, properties:

---

a message  $M$  accept a different message as authentic:  $M$  consist of blocks  $M_1, M_2, M_3, etc.$ . The attacker wants to change all plaintext blocks by the value  $V$ , which he chooses. The attacker catches  $IP$  and the ciphertext, alters  $IP$  to  $IP \oplus V$  and sends both to the receiver, who now gets the blocks  $N_1 \oplus V, N_2 \oplus V, N_3 \oplus V, etc.$ . For an even number of blocks (incl.  $RP$ ),  $RP$  has the “correct” value  $IP \oplus V$ . If we operate bit-by-bit modulo 2 it even works for an uneven number of blocks. (slightly generalized attack from [MeMa\_82, page 416])

- + An *indeterministic block cipher* can be used. For an indeterministic block cipher, the ciphertext blocks are longer than the plain text blocks. Therefore a suitable selection has to be made.
- + Using an *asymmetric* block cipher for the construction leads to an *asymmetric* stream cipher.
- The length of the encodable units is defined by the *block length* of the used block cipher. This impedes an easy adaption to the units of the transmission and storing system.
- \* A suitable selection of the function  $h$ , e. g.  $\text{mod } 2^{\text{blocklength}}$ , ensures that for even the slightest change in a block, all corresponding ciphertext blocks, including all symbols, are disturbed with a probability of

$$\frac{|\text{alphabet}| - 1}{|\text{alphabet}|}$$

Failure extension is therefore *unlimited*.

Unlimited failure extension can be used to achieve secrecy and authentication in one encryption step by concatenating at the end of the message a block of fixed values, e. g. all bits 0. Decryption will show if exactly this block is generated.

- + Generally the computationally cost that applies to the block cipher is higher than that of addition and subtraction. This saves half of the costs in comparison to encryption and authentication (e. g. CBC), because each block has to be transformed by the block cipher into two separate runs.

When comparing “cipher block chaining” with “plain cipher block chaining”, two points have to be mentioned:

- The constructions are very similar – both use en- *and* decryption of the block cipher and operate invertible on the plaintext. In particular, the latter construction contains the first one (choose a function  $h$  in such a way that it returns the chosen ciphertext block and ignores the plaintext).
- From this similarity the same advantages and disadvantages are derived. The ambivalent property “failure extension” can be divided in *limited* (self synchronizing block cipher) and *potential unlimited* (synchronous block cipher). The failure extension can be potentially unlimited, because the function  $h$  can use the previous plaintext block, which can be dependent on all previous ciphertext blocks on the decoder’s side.

Picture 3.49 makes a comparison between PCBC and CBC easily possible. For an implementation you would not save the plaintext and ciphertext block  $n - 1$ , but the output of function  $h$ . This saves memory on the one hand and decreases the costs of transmission during initialization of the memory on the other.

the number of characters carried on all lines corresponds to the block length

$\oplus$  addition, e.g. bit by bit modulo 2

$\ominus$  subtraction, e.g. bit by bit modulo 2

$\nabla^h$  arbitrary function, e.g. addition mod  $2^{\text{block\_length}}$

PCBC

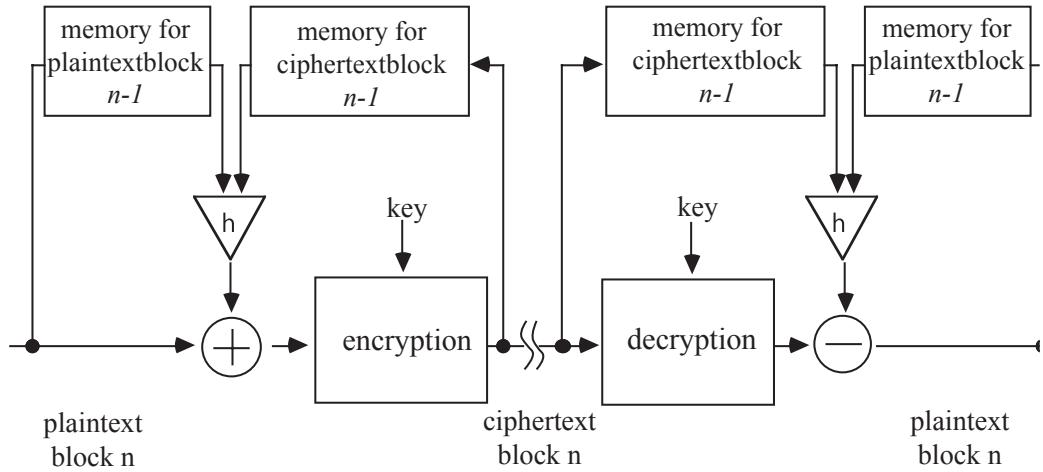


Figure 3.49: Construction of a symmetrical resp. asymmetrical synchronous stream cipher from a symmetrical resp. asymmetrical block cipher: block cipher with ciphertext and plaintext block chaining

### 3.8.2.6 Output cipher feedback (OCFB)

Accordingly, the following applies to "cipher block chaining" and "plain cipher block chaining" as well as "cipher feedback" and "output feedback"

- The constructions are very similar. In both, only the encryption of the block cipher is used to generate a pseudo-random character stream to achieve encryption using modular addition. For decryption, modular subtraction is used. Picture 3.50 shows a general construction of **output cipher feedback** (OCFB) which includes cipher feedback and output feedback given an appropriate function  $h$ . As seen in picture 3.49, it is not necessary to invert  $h$  for decryption.
- Derived from their similarity, the same advantages and disadvantages can be found. For the ambivalent property "failure extension" we distinguish between *limited* (self synchronizing block ciphers) and *potentially unlimited* (synchronous block ciphers). The failure extension can be potentially unlimited, because the function  $h$  can use the result of the block encryption which in turn can depend on the full previous ciphertext stream.

- b block length
- a length of the output unit,  $a \leq b$
- r length of the feedback unit,  $r \leq b$
- $\oplus$  addition relative to suitably chosen module
- $\ominus$  subtraction relative to suitably chosen module
- $\nabla^h$  arbitrary function

OCFB

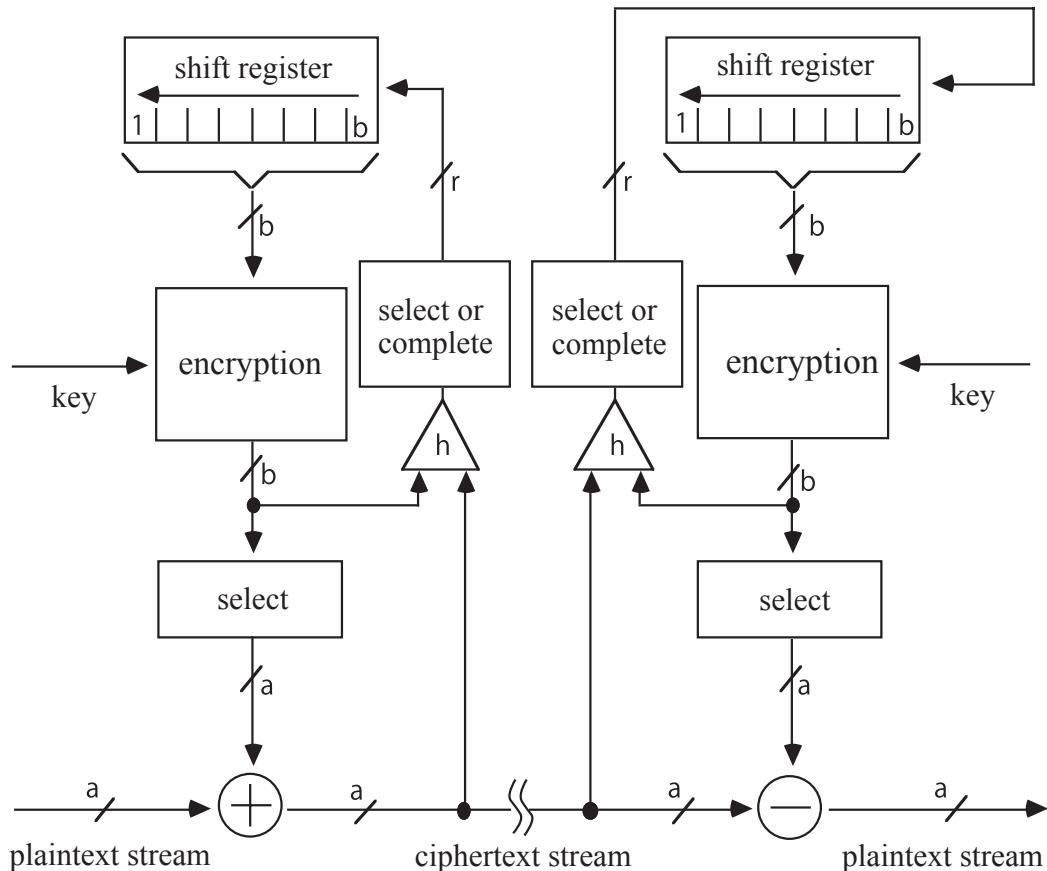


Figure 3.50: Construction of a synchronous stream cipher from a deterministic block cipher: Cipher and output feedback<sup>65</sup>

### 3.8.2.7 Annotations to expanding block ciphers and memory initialization

In pictures 3.42, 3.46, 3.48, 3.49, and 3.50 as well as in the associated operation descriptions, only the usual case is described: The block cipher maps plaintext blocks onto ciphertext blocks of the same length. For longer *ciphertext blocks* we have to select a number of characters. This selection should be made previous to the "memory for ciphertext block  $n - 1$ ". The same applies to picture 3.49. In the pictures 3.46, 3.48, and

3.50 the existing "selection function" has to be modified.

Another topic that can not be covered here is the necessity of the same initialization of the memory. The memory for the ciphertext block  $n - 1$  in picture 3.42 *should* be initialized with suitable values. The memory for ciphertext block  $n - 1$  and plaintext block  $n - 1$  in picture 3.49 *have to* be initialized with suitable values. The shift registers in picture 3.46 *should* and the ones in picture 3.48 and 3.50 *have to* be initialized with suitable values [MeMa\_82, DaPr\_89]. These statements are made in the case that the systems is used to achieve secrecy. For authentication, when generating and testing the authenticator the same initialization has to be used, but which can be the same for all partners and messages. A more detailed view is given in question 3-18 d).

#### 3.8.2.8 Summary on the properties of operation modes

Picture 3.51 gives a compact summary of the properties of block ciphers described before. While the text starts describing the simple (standardized, as ECB, CBC, CFB, and OFB) block ciphers first, before giving descriptions for more complex solutions or, in other words, begins with special followed by general constructions, picture 3.51 distinguishes between two groups according to similar properties.

The left group contains the ciphers where both functions (en- and decryption) of the block cipher are used and the block cipher is embedded into the direct transformation path from the plaintext over the cipher back to the plaintext.

The right group distinguish itself through the generation of a pseudo random number using the block ciphers encryption function. The decryption of the block cypher is not used. The transformation path only contains addition and subtraction operations as we already know from the Vernam cipher.

These group specific properties are derived from the first three properties. The mode listed on the right side contains both of those on the left. Therefore the error extension of the rightmost mode is only *potentially* unlimited and it is  for authentication in the same round. An appropriate function  $h$  has to be applied at the same time. This also applies to the question of PCBC or OCFB being synchronous stream ciphers. For an appropriate choice of  $h$  they are synchronous stream ciphers.

In addition, the following 3 properties are of interest for operation modes in practical usage by means of utilization and implementation:

**Elective access:** Is it possible to access parts of the ciphertext and/or plaintext electively? Meaning, is it possible to achieve the cipher text from an elective chosen plaintext by calculation and/or vice versa? Or is it necessary to calculate the ciphertext or plaintext starting from the beginning of the message?

**Pipelining:** Is it possible to run encryption and/or decryption in parallel? Or do they have to be strictly sequential?

**Pre-calculation of the block cipher:** Is it possible to calculate the block cipher in advance or do we have to wait until the plaintext or cipher text is available to process this, by far computationally most expensive, part?

	ECB	CBC	PCBC	CFB	OFB	OCFB
usage of indeterministic block ciphers	+ possible			– impossible		
asymmetrical block cipher produces	+ asymmetrical stream cipher			– symmetrical stream cipher		
length of encryptable units	– specified by block length of block cipher			+ arbitrary		
error expansion	inside one block only	2 blocks	potentially unlimited	1 + $[b/r]$ blocks, if error on the left hand side, otherwise possibly one block less	none with falsification	potentially unlimited
also usable for authentication?	with redundancy in every block: yes	with deterministic block cipher: yes	yes, even concealation in the same process	yes	with appropriate redundancy: yes	yes, even concealation in the same process
random access possible?	yes	only with decryption	no	only with decryption	no	no
parallelizing possible?	yes	only with decryption	no	only with decryption	no	no
Calculation of the block cipher in advance possible?	no			for one block at a time	yes	for one block at a time

Figure 3.51: Properties of the operation modes

### 3 Cryptography Basics

The following connections exist between the properties of operation modes and these 3 additional properties:

*Elective access for decryption* follows from *self synchronous*, because at least the part following after the part of the cipher text, being necessary for synchronization, can be decrypted separately.

*Pipelining* follows from *selective access*, because independence of parts from each other makes parallelism possible.

For block ciphers directly located on the transformation path from the plain text over the ciphertext towards the plaintext, it is not possible to pre-calculate the block cipher. Otherwise, at least one block can be calculated in advance.

Collectively this knowledge results in the 3 lines at the end of picture 3.51.

### 3.8.3 Construction of a collision-resistant hash function from block ciphers

In this section it will be described, how to construct a collision resistant hash function from a deterministic block cipher, with its key not much longer than its block length.

As already described for collision resistant permutation pairs in §3.5.1, finding collisions could only prove to be complexity theoretical difficult – while informational theoretical difficulty is easy to prove, because the attacker could always check all possibilities. The same applies for finding collision resistant hash functions: Because they map arbitrary long arguments onto values of fixed length, an arbitrary number of collisions must exist (that means different arguments get mapped onto the same value). To find a collision, the attacker only has to (be able to) search persistently.

The best to happen would be an *efficient* construction which creates collision resistant hash functions from *any* block cipher. These *have to prove to be* as secure as the block cipher which was used to create them. Unfortunately, none such construction is known yet. There are not even constructions fulfilling two of the three emphasized requirements. As there were already secure collision resistant hash functions given in §3.6.4 (under the assumption on factorization), we are now setting a great store by efficiency. Therefore we show in picture 3.12 “well explored” very efficient collision resistant hash functions, which can be constructed from deterministic block ciphers using keys which are not allowed to be longer than their block length due to security reasons [LuRa\_89].

In [DaPr\_84] and [DaPr\_89, page 265] the following model of a collision resistant hash function constructed from a deterministic block cipher (e. g. DES) is proposed:

The plaintext to be hashed becomes split into bits of the length equal to one of the block ciphers keys. In each round one of these plaintext blocks is used as a key for the block cipher.

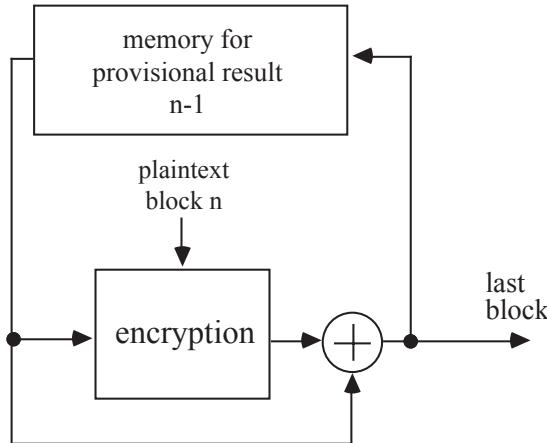
Input for the first round is an initial value, while for following rounds the result of the previous round is used as input. (In picture 3.52 the initial value is set as the result of round 0)

In each round not only encryption takes place, but also the block ciphers input becomes added modulo 2 to its output. This stops the attacker from calculating backwards by

choosing a key (means: plaintext block) followed by inverting according to the chosen key (means: decrypt) to get the input of this round. By doing this it would be possible for him (assuming the initial value can be chosen without restrictions) to calculate collisions just by doing the described with 2 different keys.

The result of the last round is the output of the hash function. (To make it possible for others to test this hash value, the initial value has to be known. For initial values not globally defined, it has to be provided.)

the number of characters carried on all lines corresponds to the block length  
 length of plaintext blocks is equal to key length of the block cipher  
 ⊕ addition mod 2



For security reasons the following should apply:  
 Key length of block cipher is not substantially longer than its block length.

Figure 3.52: Construction of a collision resistant hash function from a deterministic block cipher

In Picture 3.52 the arrangement of the components is chosen as in picture 3.42, which makes it easy to see that in the process the plaintext is used in an unusual way – i. e. as key.

It is important – regardless of how the hash function is constructed – that the initial value is **fixed** or the message is encoded **postfix free**<sup>66</sup>. Otherwise the following trivial collisions will occur: Suitable initial values for the not yet processed part of the message are derived as a result of the previous round.

This weakness can be overcome by storing the **length of the message in Bit** in the last plaintext block and it can be determined how the number of Bits added [LaMa\_92, page 57].

---

<sup>66</sup>No postfix – final part of a postfix free mapped message can be postfix free mapping of another message.

It is also possible to calculate collisions with a probability of  $2^{b/2}$  for block ciphers with an output of length  $b$  using a brute force search (e. g. this applies to the just given construction using a block cipher of length  $b$ ). It derives from the so called **birthday paradox**: If two people who have their birthday on the same day, then  $365/2$  people are not required, but only  $\sqrt{365}$  [DaPr\_89, page 279-281 ]. Therefore DES will be insufficient for many applications using the specified construction, because the amount of  $2^{32}$  attempts by many possible attackers cannot be ruled out. Even a block length of 128 bit will not be sufficient in the near future, because soon  $2^{64}$  attempts will not be an obstacle for clever attackers. Accordingly, collision resistant hash functions should provide results with length of at least 160 bit.

A precise classification of collision resistant hash function based on which type of attack (analogous to §3.1.3.1 and §3.1.3.2) they resist and a few sentences about the dependence of the hash functions collision resistance from the collision resistance of a single iteration, can be found in [LaMa\_92].

## 3.9 Outline of further systems

### 3.9.1 Diffie-Hellman key exchange

So far only asymmetric cryptographic systems whose security relies on the factorization assumption have been discussed, cf. §3.4.1. The following procedure for the calculation of shared secret keys based on public information only (Diffie-Hellman-Key-Exchange [DiHe\_76]) relies on the discrete logarithm assumption, but has become of both theoretical and practical importance in recent years:

From version 5, Pretty Good Privacy - PGP uses a variation of the Diffie-Hellman-Key-Exchange (**ElGamal encryption system** with individual parameter selection, cf. question 3-23c) in a hybrid cryptographic system, cf. §3.1.4, instead of RSA, because – in contrast to RSA – the corresponding patent has expired.

The Diffie-Hellman-Key-Exchange allows Steganography with public keys, cf. §4.1.2.

First a few fundamental principles, in particular, what generators of  $\mathbb{Z}_p^*$  are and how we can obtain them:

An element  $g$  of a multiplicative group  $G$  is called **generator** of  $G$ , if the powers of  $g$  create the group  $G$ . This means that for each element  $h$  of  $G$  there is an integer  $i$  with  $0 \leq i < |G|$  so that  $g^i = h$ . A group with at least one generator is called *cyclic*.

For each prime number  $p$ ,  $\mathbb{Z}_p^*$  is a cyclic group. It is easy to chose  $\mathbb{Z}_p^*$  and a generator  $g$  randomly [MeOV\_97, page 163]:

1. Chose a prime number  $p$  randomly (cf. §3.4.1).
2. Factorize  $|\mathbb{Z}_p^*| = p-1$ . Be  $p-1 = p_1^{e_1} \cdot p_2^{e_2} \cdot \dots \cdot p_k^{e_k}$ , where  $p_1 \dots p_k$  are pairwise distinct prime numbers. If the factorization is not successful within an acceptable length of time, go to 1.

3. Choose an element  $g$  in  $\mathbb{Z}_p^*$  randomly.

4. For  $i$  from 1 to  $k$

Calculate  $b := g^{(p-1)/p_i}$

If  $b = 1$ , go to 3.

Steps 1. and 2. chose a group  $\mathbb{Z}_p^*$ , of which the prime factorization of the number of elements is known. In order to increase the efficiency of the factorization in step 2. and to ensure that  $p - 1$  has at least one big prime factor – which is desirable for the security from the calculation of the discrete logarithm – , an alternative to steps 1. and 2. is described below:

Generate a prime number  $p_1$  with length  $l = \text{const}$ , where  $l$  is the security parameter

Factorize  $f$ . (This is easy because  $f$  is not too big.) Let  $f = p_2^{c_2} \bullet \dots \bullet p_k^{c_k}$ .

Search for prime numbers among the numbers  $p := f \bullet p_1 + 1$  with  $|f| = \text{const}$ .

Factorize  $f$ . (This is easy because  $f$  is not too big.) Be  $f = p_2^{c_2} \bullet \dots \bullet p_k^{c_k}$ .

This procedure changes the distribution of the choice of the prime number  $p$ . But as far as we know, this does not matter.

Steps 3. and 4. chose a generator within the group  $\mathbb{Z}_p^*$ , of which the prime factorization of the number of elements is known:

Let  $h$  be an arbitrary element of the multiplicative group  $G$ .

The powers of  $h$  form is a subgroup of  $G$ , as  $h^x \bullet h^y = h^{x+y}$  and  $(h^x)^{-1} = h^{-x}$ .

The Lagrange proposition establishes: If  $H$  is subgroup of  $G$ , then  $|H|$  is factor of  $|G|$ .<sup>67</sup>

If  $H$  is a proper subgroup of  $G$ , i. e.  $|H| < |G|$ , and  $|G| = p_1^{e_1} \bullet p_2^{e_2} \bullet \dots \bullet p_k^{e_k}$ , then  $\exists i, 1 \leq i \leq k : |H|$  is factor of  $|G|/p_i$ .

For this  $i$  is true:  $h^{|H|} = 1$ , and especially  $h^{|G|/p_i} = 1$ .

If  $h$  is a generator of  $G$ , then  $h^{|G|/p_i}$  must always have a value  $\neq 1$ , otherwise  $h$  would not create a group with  $|G|$  elements, but a group with  $h^{|G|/p_i}$  elements at the most. The reason is, that from value 1, only repetitions and no new elements will be created when the power is increased.

Therefore the above algorithm recognizes in step 4 whether  $g$  is a generator of  $\mathbb{Z}_p^*$  or not.

---

<sup>67</sup>This was already used in §3.4.1.2 to prove Fermat's small proposition.

### 3 Cryptography Basics

The Diffie-Hellman key exchange relies on the difficulty of calculating **discrete logarithms**, i. e. logarithms modulus a prime number. Let  $p$  be a prime number,  $g$  a generator of  $\mathbb{Z}_p^*$ , the multiplicative group modulus  $p$ . Let

$$g^x = h \pmod{p}$$

Then  $x$  is called discrete logarithm of  $h$  to the base  $g$  modulus  $p$ :

$$x = \log_g(h) \pmod{p}$$

**Assumption on the Discrete Logarithm:** According to §3.1.3.4 it is not provable at the moment, that it is hard to calculate discrete logarithms. An assumption has to be made which says that for each fast (discrete-logarithm-)algorithm  $\mathcal{L}$ , the probability that  $\mathcal{L}$  is capable of calculating the discrete logarithm, quickly decreases with the length of the modulus.

*Assumption:* For each (probabilistic) polynomial algorithm  $\mathcal{L}$  and each polynome  $\mathcal{Q}$  it is true that: There exists an  $L$  with  $l \geq L$  and it is: If  $p$  is a randomly chosen prime number of length  $l$  and  $g$  is randomly chosen from the set of generators of  $\mathbb{Z}_p^*$  and then  $x$  is randomly chosen from  $\mathbb{Z}_p^*$  and  $g^x = h \pmod{p}$

$$W(\mathcal{L}(p, g, h) = x \leq \frac{1}{\mathcal{Q}(l)},$$

**Diffie-Hellman key exchange:**  $p$  and  $g$  are public and fixed. Participants choose their private keys from elements of  $\mathbb{Z}_p^*$  randomly, calculate  $g$  to the power of this element mod  $p$  and publish the result<sup>68</sup>. Each communication partner can "exchange" his own public key by simply raising the others public key to the power of his own private key. Due to the commutativity of the exponential function and therewith the modular exponential function, both get the same secret key, cf. picture 3.53.

Now we hope that nobody other than these two is capable of calculating the secret key  $k = g^{xy} \pmod{p}$ . This is the **Diffie-Hellman-Assumption**. It is a stronger one than the assumption for the discrete logarithm, because if someone is able to calculate the discrete logarithm, then one will get  $x$  from  $g^x$  and  $y$  from  $g^y$ . With this one can easily get  $g^{xy} \pmod{p}$  from  $g^x \pmod{p}$  and  $g^y \pmod{p}$ . When put backwards it could not be shown that  $x$  and  $y$  can be derived from  $g^x \pmod{p}$ ,  $g^y \pmod{p}$  and  $g^{xy} \pmod{p}$ .

It is worth remarking that the key does not have to be transmitted through the network, but can be calculated locally by the communication partners. This is important for applications using Steganography and leads to Public Key Steganography, cf. §4.1.2.

For Diffie-Hellman key exchange, instead of the public encryption keys, the public keys of the participants are stored in public key registers. Apart from that, everything works as shown in picture 3.5 and question 3-4.

---

<sup>68</sup>The publishing must be authentic, see question 3-4. This can be ensured by multiple certification of the public key, cf. §3.1.1.2

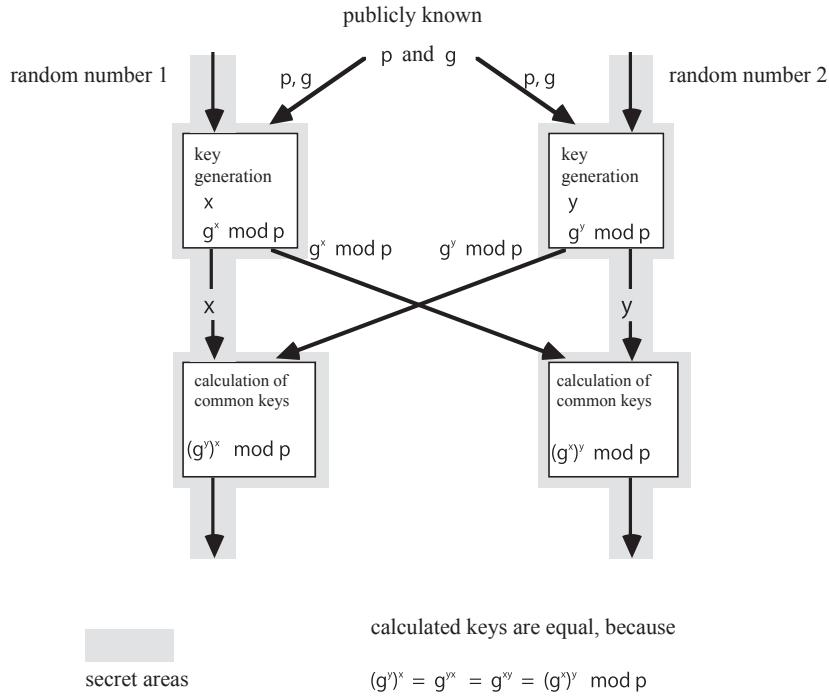


Figure 3.53: Diffie–Hellman key exchange

### 3.9.2 For the signee unconditional secure signatures

The security of digital signature systems has to be seen as "asymmetric":

- unconditionally secure for the recipient of the signature – given the recipient possesses a pair (text, signature), which can be tested positive, he is able to convince everybody that this text was signed by the owner of the pair of keys  $(s, t)$
- only cryptographically secure for the signee – because it is possible for an attacker to calculate the unique signature  $s(x)$  for a text  $x$  and a public key  $t$ , cf. picture 3.54. How should our poor signee prove that he did not calculate  $s(x)$ , even though he is capable of this.

The security can also be divided "asymmetric" the other way around. For a signee using an unconditionally secure authentication system (picture 3.55 and 3.56) it is:

- unconditionally secure for the signee – for each public testing key there exist many secret signing keys, which on the other hand are loosely distributed in the space of signing keys. A signee, restricted by means of complexity theory, only knows one of all these keys, namely  $s$  in picture 3.55. Using  $s$  he can produce exactly one signature  $s(x)$ . A signature generated by a complexity theoretical non-restricted

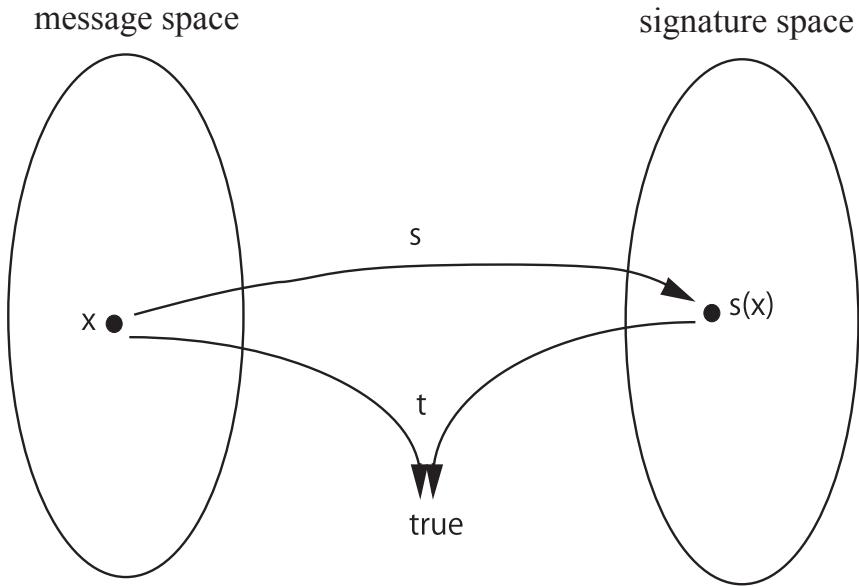


Figure 3.54: Usual digital signature system: There is one and only one signature  $s(x)$  for one message  $x$  and one validation key  $t$  (and possibly one signature environment, see GMR).

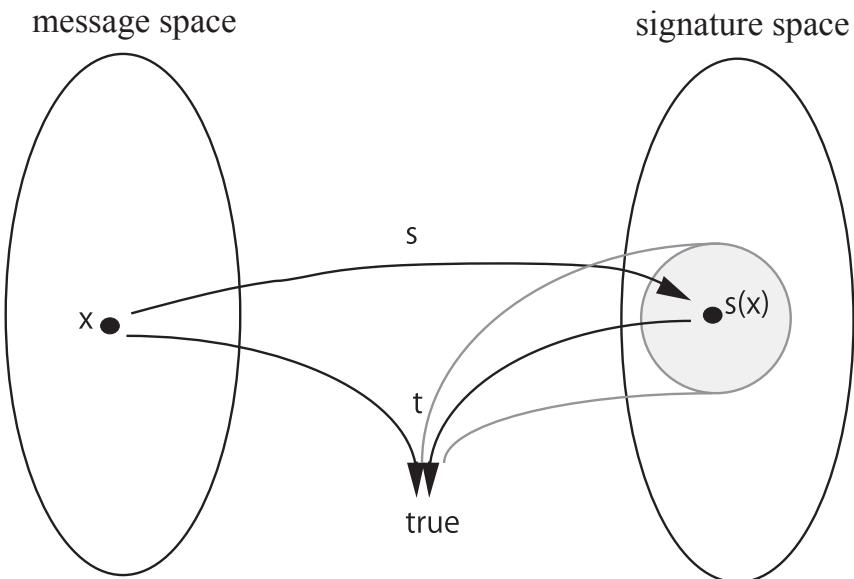


Figure 3.55: For the signee unconditionally secure signature system: For one message  $x$  and one validation key  $t$  there are multiple signatures (gray area), which are rather sparse within the signature space

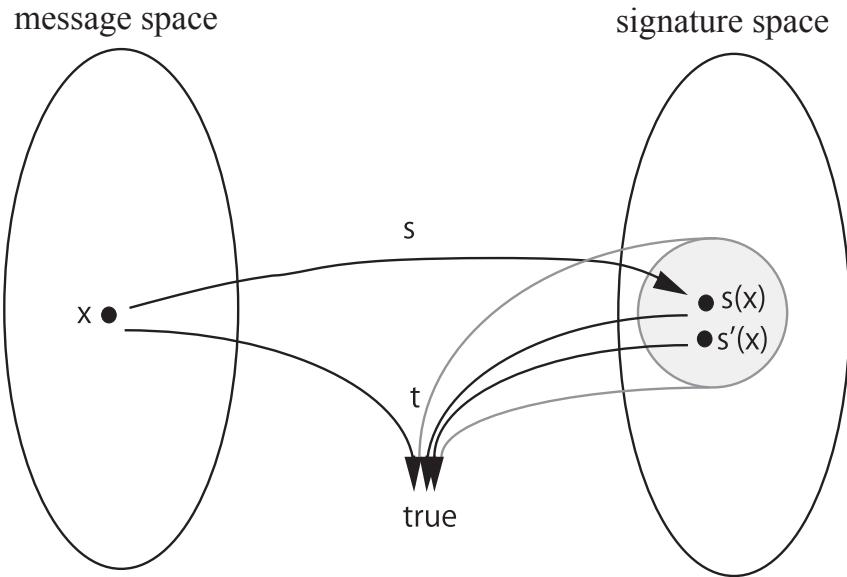


Figure 3.56: Proof of falsification in the form of a collision in a signature system unconditional secure for the signee

attacker is different from  $s(x)$  with a probability of

$$\frac{1}{\text{number of all possible signatures}}$$

The signee has the possibility of checking with his own secret key  $s$ , if  $s'(x) = s(x)$ , cf. picture 3.57. If not, i. e.  $s \neq s'$ , the signee has the proof that the signature was falsified and the assumption on complexity theory violated. One should no longer trust the assumption, e. g. use longer prime numbers. Moreover, the responsibility can be defined in such a way that the risk is on the recipient's side.

- only cryptographically secure for the recipient – because is it possible that an attacker or an actual signee finds a proof of falsification.

The main advantage of a signature system like the one described, is that one gets a proof if the system is broken (Fail-Stop-Signature-System). Therefore the responsibility and damage can be arbitrarily shared: Each complexity theoretical restricted participant (including the legitim signee, the generator of the key pair  $(t, s)$ ) is only able to give a proof on falsification with a negligible probability.

More about this kind of authentication system can be read in [PfWa\_91, Pfit8\_96] as well as in [WaPf\_89, WaPf1\_89, Pfit\_89, Bleu\_90, BIPW\_91, HePe\_93]

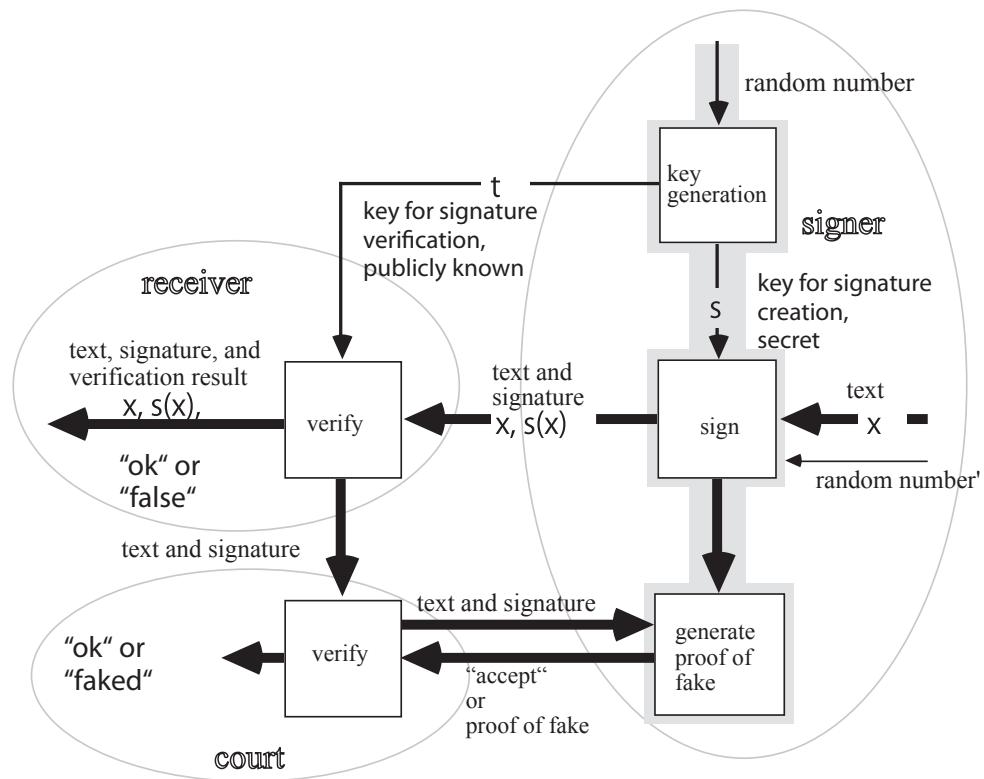


Figure 3.57: Fail-stop signature system (Glass box with a lock; there is only one key for putting things in. The glass can be broken, but it can not be replaced unnoticed.)

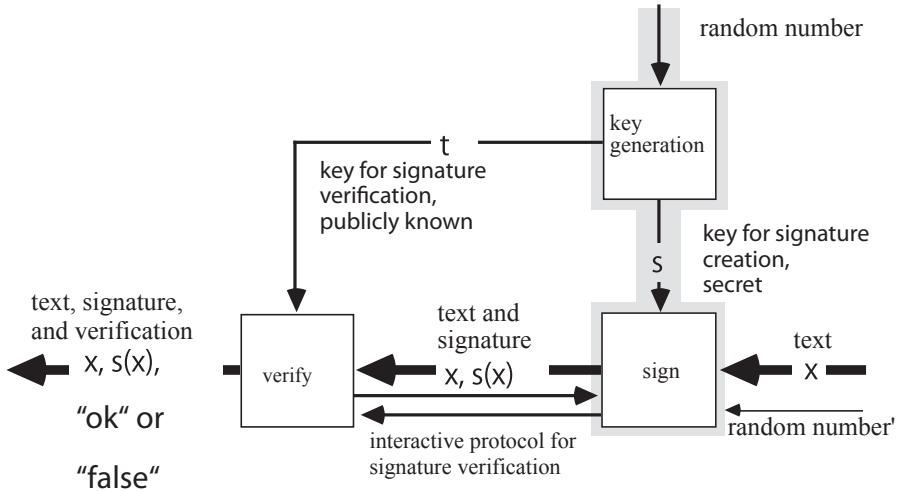


Figure 3.58: Signature system with undeniable signatures (black box with a spyhole; there is only one key for putting things in. The key owner controls access to the spyhole.)

### 3.9.3 Unconditional secure pseudo-signatures

There now exists a systems which has almost the same properties as digital signatures and is informational theoretical secure, i. e. signatures cannot be falsified (except of the exponential small probability that the attacker guesses right, which all authentication systems have in common). Unfortunately, so far it is quite impracticable. More about this kind of systems can be found in [ChRo\_91].

### 3.9.4 Undeniable signatures

Usual digital signatures can be copied perfectly and therefore shown around without restrictions. The idea is to evolve the supposed signee in the testing procedure using an interactive protocol instead of testing signatures using an autonomous deterministic algorithm, cf. picture 3.58. This way the signee finds out if his signature ought to be shared with a third party ("share" means not only to hand over the knowledge about the bit chain, but also to verify that it is a valid signature.). The important thing about the protocol is that it is impossible for the signee to deny a valid signature (therefore the name by David Chaum: undeniable): If he cheats he will get caught with a probability close to 1. If he refuses to participate in the protocol, even though he is obliged to, it gives a proof for the authenticity of the signature.

### 3.9.5 Blind signatures

The aim of this type of signature systems is for the signees to be able to sign without any knowledge about *what* they sign. At first glance this might sound completely pointless, because the signee should be interested in what he signs. For some applications the opposite applies: Here it is important knowledge to the signee that he signs. He is not interested in what he signs – he even must not be interested in it.

An example could be a digital payment system, where the signee creates notes of certain values by signing them with the appropriate key (a random number of suitable length and redundancy). He just has to realize and to be sure that he only creates one note of this value. Now he can demand the appropriate amount of money for it. The one who buys the bank notes, by means of getting a blind signature for them, does not want information about him to be transferred. Then the issuer cannot analyze payments in digital payment systems. More about this application can be found in §6.4.

Picture 3.59 shows the procedure of blind signatures:

As usual, the signee creates a pair of keys  $(t, s)$  and  $t$  is made public.

In contrast to previous procedures, the text  $x$  to be signed, is generated by the recipient of the signature (not by the signee) – if necessary, redundancy is added (cannot be seen in picture 3.59) and it becomes masked by a random number  $z'$ , in a certain sense it is encrypted.

The masked text  $z'(x)$  becomes transmitted to the signee for signing.

The signee signs  $z'(x)$  with  $s$  and returns  $z'(x), s(z'(x))$  to the recipient of the signature.

Only the recipient knows the random number, which inhibits overhearing parties on the transmission path and even the signee itself from determining the utility of the blind given signature. The recipient unmasks  $z'(x), s(z'(x))$  with  $z'$  and gets the signature  $s(x)$

Now one can test with  $t$  if the signature  $s(x)$  fits to  $x$ .

#### Blind signatures with RSA:

The attack by David in Judy Moore's variation, cf. §3.6.3.1, this time used for good:

Key generation, adding redundancy to texts (has to be performed before masking), signing and testing follows exactly the description in §3.6.4.2

Choose the random number  $z'$  normal distributed and randomly out of  $\mathbb{Z}_n^*$

Masking is performed by modular multiplication of the text including the added redundancy with  $z'^t$ . The result is the masked text  $x \bullet z'^t \pmod{n}$ .

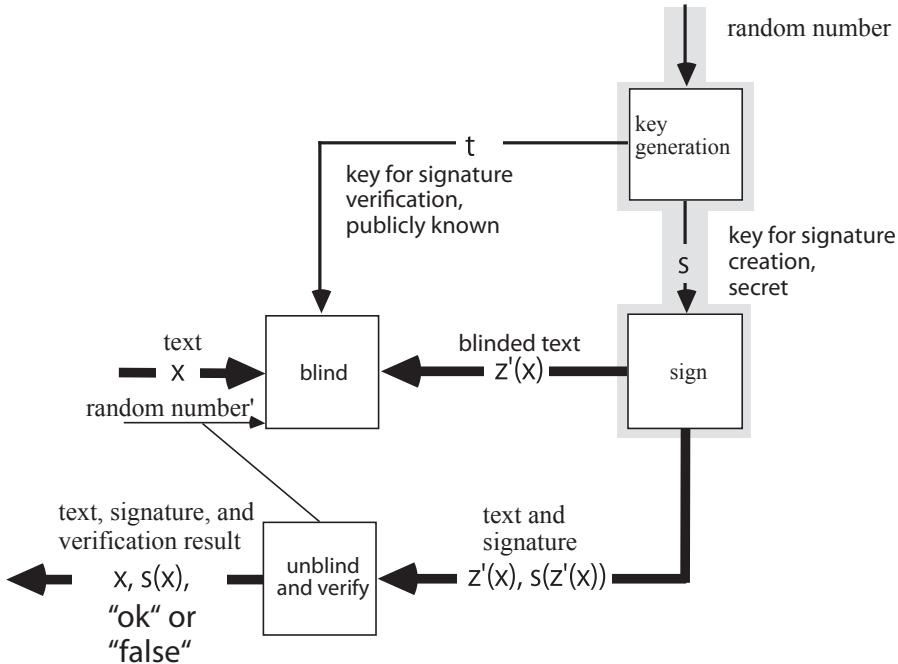


Figure 3.59: Signature system for blind signatures

(Annotation: For  $x \in \mathbb{Z}_n^*$  the signee does not learn anything about  $x$  by means of information theory, because  $z'^t$  is normal distributed in  $\mathbb{Z}_n^*$ . Otherwise the generator of  $x$  is not dependent on the signee:  $ggt(x, n)$  gives a non trivial factor of  $n$ . With this the generator of  $x$  breaks RSA completely, cf. §3.1.3.1 and §3.6.1, in such a way that he is now able to sign on his own.)

The masked text with signature  $(x \bullet z'^t)^s \pmod{n}$  derives by modular exponentiation of  $x \bullet z'^t$  with  $s$ .

Unmasking is achieved by division by  $z'$ :  $(x \bullet z'^t)^s \bullet z'^{-1} \pmod{n} = x^s \bullet (z'^t)^s \bullet z'^{-1} \pmod{n} = x^s \bullet z' \bullet z'^{-1} \pmod{n} = x^s$

The blindly signed text with signature is:  $x, x^s$

More about this type of signature system in [[Chau\\_83](#), [Cha1\\_83](#), [Cha1\\_88](#), [Cha8\\_85](#), [Chau\\_89](#), [Schn\\_96](#)]

### 3.9.6 Threshold scheme

Using *threshold schemes* (or *secret sharing schemes*), the holder of a secret  $G$  can split it into  $n$  parts, so that at least  $k$  parts are necessary to construct the secret efficiently and  $k - 1$  parts do not provide any information about the secret.

### Adi Shamirs polynome interpolation

Let the secret  $S$  be an element of  $\mathbb{Z}_p$ , and  $p$  be prime<sup>69</sup>

The holder chooses a polynome  $q(x)$  of order  $k - 1$

He chooses the coefficients  $a_1, a_2, \dots, a_{k-1}$  randomly, uniformly, and independently in  $\mathbb{Z}_p$ .

He defines  $q(x) := S + a_1x + a_2x^2 + \dots + a_{k-1}x^{k-1}$ , where all operation take place in  $\mathbb{Z}_p$ .

He calculates all  $n$  parts  $(i, q(i))$  with  $1 \leq i \leq n$  for  $n < p$  in  $\mathbb{Z}_p$ .

From  $k$  parts  $(x_j, q(x_j)) (j = 1, \dots, k)$  the polynome  $q(x)$  can be calculated as follows (Lagrange Polynome Interpolation [Denn\_82, p.180]):

$$q(x) = \sum_{j=1}^k q(x_j) \prod_{m=1, m \neq j}^k \frac{(x - x_m)}{(x_j - x_m)} \mod p$$

The secret  $S$  is produced as  $q(0)$ .

Proof outline:

1. With the knowledge about only  $k - 1$  parts  $(x_j, q(x_j))$  the threshold scheme gives no information about  $S$ , because for *each* value of  $S$  there always exists exactly *one* polynome of order  $k - 1$ , which goes through the  $k$  points  $(j, q(j))$  and  $(0, S)$
2. The formula above calculates the correct polynome, it has the correct order (each factor is of order  $k - 1$  and summation does not change this) and returns the correct value  $q(x_j)$  for each  $x_j$ : Substitution of  $x$  by  $x_j$  in  $\prod$ , the product will be 1 for index  $j$  in  $\sum$  (because each single quotient would evaluate to 1 and their product would be 1) and otherwise 0 (because one factor would be 0). Multiplication of this 1 and  $q(x_j)$  equals as total  $q(x_j)$ .

*Annotation* It is not compulsory to chose the values of the polynome at position 1 to  $n$ . Another choice  $\neq 0$  is suitable as well.

For large secrets one does not want to calculate modulo as a large number. The secret can be split up into smaller parts. Then the protocol above needs to be applied to all parts separately.

Often the occasion arises, that not only confidentiality has to be guaranteed for our secret. Long term accessibility has to be ensured for its owner. The one who wants to protect himself from loss of his secret  $S$  by supernatural powers, but does not want

---

<sup>69</sup>Watch out:  $p$  must not be chosen in great dependence on  $S$ . On no account: Chose the next greater prime number to  $S$ . Otherwise the attacker gets already much information about  $S$  from the public prime number  $p$ .

to entrust it to a single party, could distribute the parts of the secret to his  $n$  friends, a different part each. In case of a loss of his secret, he requests  $k$  friends to send a copy of their share. From these the owner is capable to calculate  $S$  efficiently (generally quadratic cost in  $k$ ). For plots of maximum  $k - 1$  friends, they cannot reconstruct  $S$  behind its owner's back, no matter how long they compute. An owner who chooses a  $k$  nearly as big as  $n$  might lose the capability to reconstruct his secret in case of only some lost parts at his friends (through supernatural powers or just malice). On the other hand, such a choice would make unauthorized reconstruction nearly impossible. Vice versa, a small  $k$  in relation to  $n$  would decrease the problem of too many parts getting lost on the friend's side and therefore the danger of losing the whole secret. Otherwise it would be easier to reconstruct it without authorization while only compromising some of the secret keepers. Through a suitable choice of  $k$  and  $n$ , the owner can adapt the decomposition of his secret to his needs.

Proofs are given in [Sham\_79], an extension to information theoretical security for the detection of changes of parts can be found in [ToWo\_88].

The owner has to sign all the  $n$  parts of the secret  $(i, q(i))$  to counteract violations of the integrity at complexity theoretical strength. Only parts with their signature tested positively are used to reconstruct the secret. Assuming that the signature was not falsified, nobody can trigger the reconstruction of false secrets or make the reconstruction computational expensive. The first one would apply in the case that only  $k$  parts are available and one of them is incorrect. The latter one would arise if  $k + 1 + \nu$  parts are available, where  $k + 1$  are correct and  $\nu$  are incorrect. Because of the lack of information about the actual parts that are incorrect, all sets of  $k$  out of  $k + 1 + \nu$  have to be checked to determine a majority, which will prove to be very computationally expensive.

Everything said about the threshold scheme is based on the assumption that the holder of the secret is not malicious, i. e. generates  $n$  parts which add up to the secret. Only the environment, particularly some of the secret keepers of the  $n$  parts were treated as potential attackers. For generators joining the group of attackers, additional properties of the threshold scheme are demanded.

1. consistent result
2. predefined result

## Recommended readings

History of Cryptography

An Overview from 1900 to 1999 from a German point of view: [Leib\_99]

Textbooks of the used mathematical background:

A real introduction to Algebra, especially for computer scientist, relatively understandable: [Lips\_81]

Elementary presentation of the basic of elementary number theory, especially the euclidian algorithm: [Lüne\_87]

### *3 Cryptography Basics*

Short and nice to read, some old terms used: [ScSc\_73]

For people who wants to know in detail about the fast algorithms: [Knut\_97]

In close connection to cryptography: [Kran\_86, MeOV\_97]

Textbooks for cryptography

Detailed classical structure: [Denn\_82, DaPr\_89, Hors\_85]

Modern structure but very short: [Bras\_88]

Modern structure, very detailed and practically relevant: [Schn\_96]

Modern structure, very detailed and theoretical profound: [MeOV\_97]

Current research

Journal of Cryptology, Springer-Verlag, Heidelberg

Conference volumes Crypto, Eurocrypt, Asiacrypt, conference volumes at LNCS, Springer-Verlag, Heidelberg

The Theory of Cryptography Library:

access via URL <http://philby.ucsd.edu/cryptolib.html>, email should be sent to [cryptolib@philby.ucsd.edu](mailto:cryptolib@philby.ucsd.edu)

References to standards, patents, products, comments to current development:

<http://www.rsa.com>

<http://www.rsa.com/rsalabs>

<http://www.itd.nrl.navy.mil/ITD/5540/ieee/cipher>  
or subscribe to [cipher-request@itd.nrl.navy.mil](mailto:cipher-request@itd.nrl.navy.mil)

Information about legal situation, according to the use, export and import of cryptography:

<http://cwis.kub.nl/frw/people/koops/lawsurv.htm>

## 4 Basics in Steganography

**Steganography** is the old art and the young science of hiding secret information in larger, harmless looking files.

Figure 4.1 illustrates the differences between steganography and cryptography in terms of the goal of protecting integrity. If good cryptography is used, the attacker notices that he can not understand the ciphertext and will hence presume that the communication is confidential. But if good steganography is used, the attacker will think that the stegotext is a plausible message which he completely understood. He does not notice any confidential communication. In the pictured example, the cover (the original holiday picture balloon over savannah) is as plausible as the stegotext (the slightly modified holiday picture: a just as beautiful balloon over a just as atmospheric savannah in the same photographic quality).

The following applies often to Figure 4.1:  $\text{cover}_* = \text{stegotext}$ , i.e., the receiver can not and does not want to reconstruct the value of  $\text{cover}$  before the embedding and takes  $\text{stegotext}$  as  $\text{cover}_*$  or completely ignores this output parameter of the **stegosystem**.<sup>1</sup>

The old art of steganography is composed, for example, often used:

- secret inks, which become invisible after writing, but which can be chemically or thermally made visible again by an informed receiver,
- microdots, which are photos minimized to the size of dots, which are then pasted in place of i-dots into harmless texts and can be viewed under a microscope, as lastly, - for reasons of human rights and real time requirements of modern telecommunication no longer practical -
- slaves. Their skulls were shaved and the message to be transmitted was written on the skin of their heads. Then the sender waited until the hair had regrown and sent the slave to the receiver together with the order to say the following to the receiver: “Master, shave my hair”.

Alternatively, messages that needed to be transferred secretly were

- swallowed by the carrier<sup>2</sup> or fed to animals<sup>3</sup> as well as

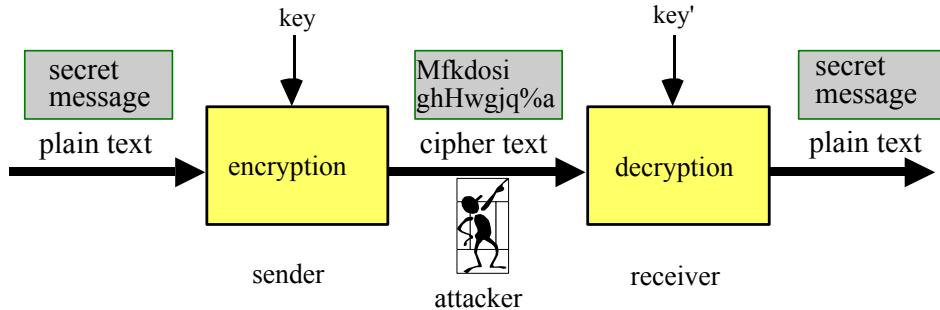
---

<sup>1</sup>Similar to how *cryptosystem* is often said instead of *encryption system*, *stegosystem* is often said instead of *steganographic system of secrecy*. Purists think it should be *steganosystem* and are certainly right. But *stegosystem* is even shorter and is hence usually used.

<sup>2</sup>This is out of fashion for secret communication, but is still common for transporting drugs. Hopefully the packaging does not break up, otherwise, the message in the first case, or the carrier in the second case, is lost.

<sup>3</sup>This has the advantage of speed in comparison to feeding messages to carriers, as one can quickly gain the message by slaughtering the animal.

### Cryptography: confidentiality



### steganography: confidentiality of confidentiality

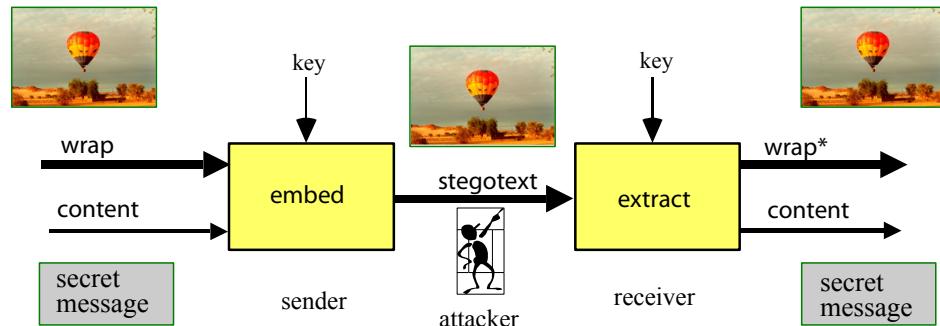


Figure 4.1: Cryptography vs. steganography

- written on the underside of wax boards, so that the insider receiver only needs to remove the wax.

A good historical overview is given in [Kahn\_96].

The young science of steganography uses computers to embed secret data in, for example, digitalized pictures, video signals, or sound signals. This can serve two purposes:

The purpose of **steganographic systems of secrecy** is not just to hide the secret message, but also to hide its very existence (compare Figure 4.2). This is helpful when confidential communication is suspected, unwanted, or even illegal<sup>4</sup>.

**Steganographic authentication systems** strive to encode information about the authors and/or the holder of the right of usage into digital(ized) works in such a way,

---

<sup>4</sup>The development of steganographic systems of secrecy was pushed through the - since 1993 openly led - discussion about the limitation or even eradication of the use of cryptography for confidential communication. The existence of safe steganographic encryption systems is an argument against so called crypto-regulation.

that it can not be removed without destroying the work and hence be used to protect copyright. With large changes to the work, the embedded copyright information extracted is naturally not accurate (cp. Figure 4.3).

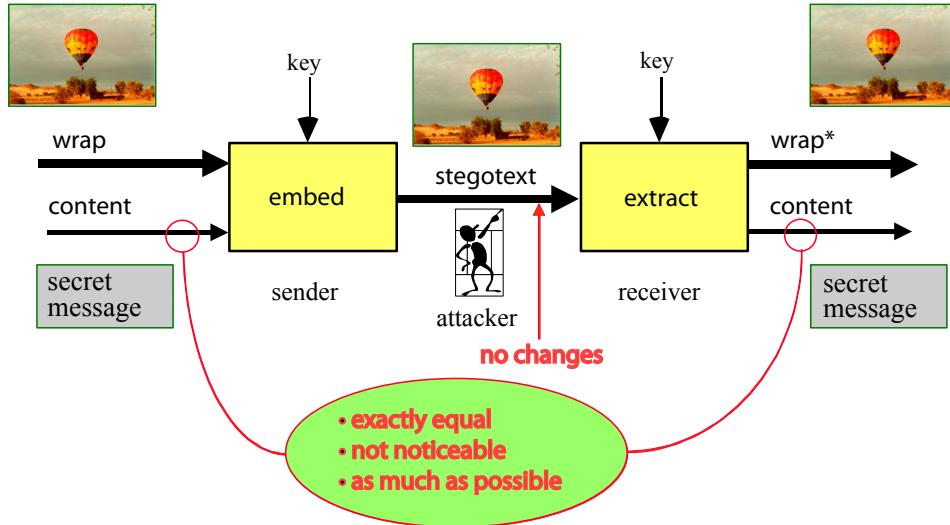


Figure 4.2: Steganographic system of secrecy

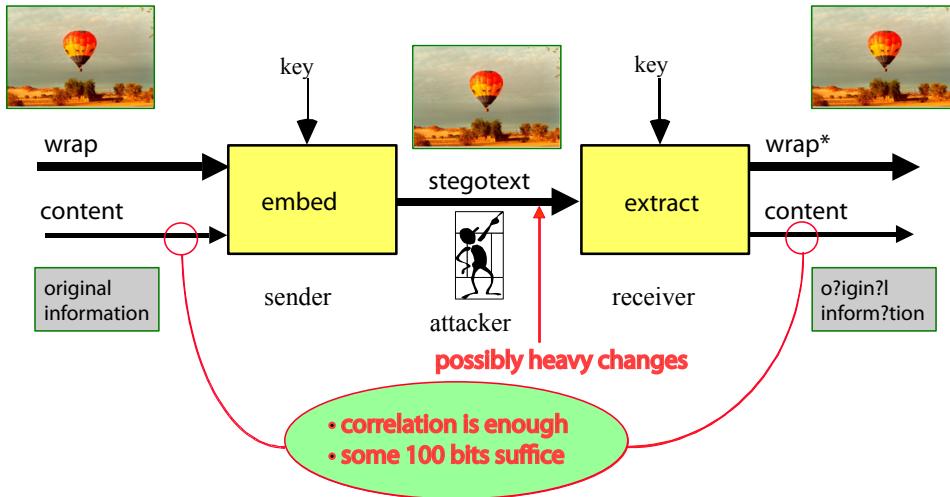


Figure 4.3: Steganographic authentication system

Systematics of steganographic systems is deployed in Figure 4.1. Figure 4.2 deduces necessary conditions for information-theoretically secure steganography. Two opposed stegoparadigms are discussed in Figure 4.3. Figure 4.4 explains the equality encoding

as means of enforcing concealment.

The following sections give examples of how to embed in digital phone calls (§4.5), pictures (§4.6), and moving images (§4.7).

## 4.1 Systematics

First it will be described in §4.1.1, what is meant by a steganographic system. Then, in §4.1.2, there follows a note about the spread of steganographic keys. The basics of steganographic security are in §4.1.3.

### 4.1.1 Stenographic systems

One thing modern steganographic systems have in common with modern cryptographic systems is that their security does not depend on the secrecy of the method, but only on the secrecy of keys which parameterize the method. Wherever the danger of confusion exists, we will speak of **stegokeys** as opposed to **cryptokeys**.

Up to now, only symmetric steganographic systems are well-known, i.e., the same information (so especially the same stegokey) is used for embedding and extraction [FePf\_97].

#### 4.1.1.1 Stenographic secrecy-systems, overview

[ZFPW\_97] cover.

Purpose: “hiding the existence of the message that needs to stay a secret” (simply stated: confidentiality of confidentiality).

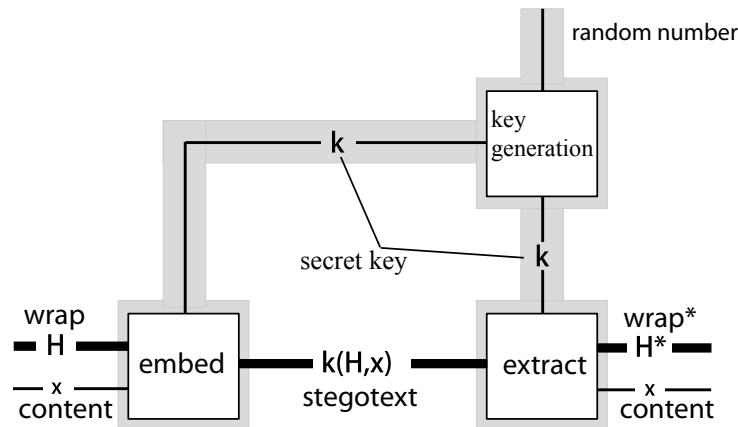


Figure 4.4: Steganographic system of secrecy according to the model of embedding

Composition model explained.

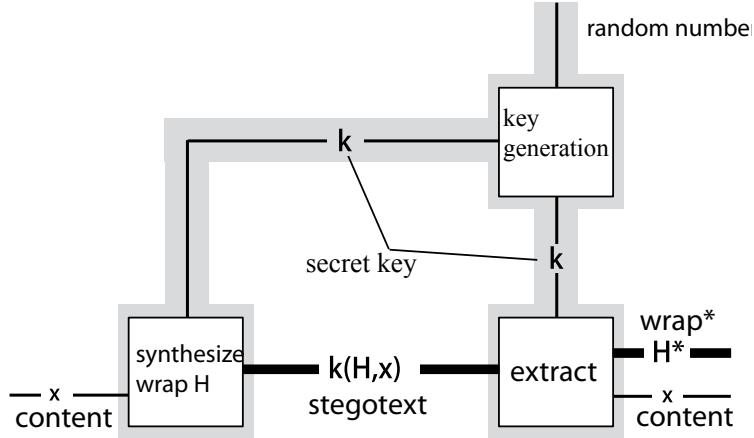


Figure 4.5: Steganographic system of secrecy according to the model of synthesis

The embedding-model is the universal system, since in it, the embedding-function can ignore the input *cover* and synthesize and use another cover, as well as the synthesis-model. Thus, it will be used as *the* model for steganographic systems of secrecy in the following section [ZFPW\_97, Chapter 2.1].

#### 4.1.1.2 Stenographic authentication-systems, overview

Watermarking and fingerprinting:

In contrast to steganographic systems of secrecy, for steganographic authentication systems it is often only required that their usage can not be sensed by humans, so that for instance watermarking and fingerprinting do not change works of art in a disturbing way. This is a much weaker claim than that watermarking and fingerprinting may not be verifiable without knowledge of the stegokey.

Indeed, for watermarking and fingerprinting, heavy and particularly specific changes to the stegotext are to be expected, in order to remove the embedded information about the author. In comparison, steganographic authentication systems need to be **robust**, i.e., the embedded information about the author must be identifiable even after such attacks. This can be achieved, for instance, through a sufficiently strong correlation between embedded and extracted information. In the only realistic scenario in open nets, namely the scenario that the attacker knows all the details of the steganographic authentication system, robustness has so far not been achieved by any steganographic authentication system. My personal assumption is that it will remain this way.

Linartz paper from the IHW98 as borderline of watermarking.

#### 4.1.1.3 Comparison

In the following table, the steganographic systems are compared with each other and with the cryptographic systems.

	<b>steganographic concealation systems</b>	<b>steganographic authentication systems</b>
<b>properties</b>	existence of embedded information must be hidden	hiding the existence of embedded information is not necessary
	cover may be modified drastically, even completely synthesized	embedding shall be smooth, i.e., modifications of the cover (which is the good to be secured) shall not be recognizable by humans
<b>goal</b>	embeds much data as possible	embed given amount of data as robustly as possible
<b>attacker's goal</b>	detect existence of embedded data	targeted change of watermark or fingerprint, e.g., replace, destroy, generate
<b>parameters</b>	stego unmodified	stego may be modified by the attacker to some extent, but not too drastically (this would destroy the good) watermark or fingerprint may be input parameter for <i>both</i> parties the unchanged good (this can be the good both with and without embedded copyright information) may be input to <i>both</i> parties
<b>covering of functionality of the corresponding cryptographic system</b>	yes, because confidentiality of a message's existence always provides confidentiality of a message's content	no, because copyright can be detected, but not small modifications <sup>5</sup>

---

<sup>5</sup>Some authors define so called *fragile* watermarks, i.e., watermarks without robustness, which just have the goal of detecting modifications to the good. I personally think that fragile watermarks are unnecessary, because this security goal can be reached with cryptographic authentication systems *without* modifications to the good.

## 4.1.2 An note on key-distribution: steganography with public keys

Symmetric steganography + Diffie-Hellman key-distribution, compare §3.9.1.

If PGP ex version 5 is used, Diffie-Hellman keys should be created with the setting “Faster key generation”. Then a common  $p$  and  $g$  is used for all participants.

If the participants do not use the common  $p$  and  $g$ , then Diffie-Hellman key-distribution is not usable for steganography, while it is still usable for cryptography, (cp. Exercise 3-28c)).

Then all that is left for steganography is what is called steganography with public keys in [Ande4\_96],[AnPe\_98, page 480](cp. Exercise 4-6): First, the - with the public cryptographic key of the partner encrypted - steganographic session key is, without use of a steganographic key, embedded and transmitted.

For this encryption of a steganographic session key, the Diffie-Hellman key-distribution can be used also without the common  $p$  and  $g$  (cp. Exercise 3-28c)): The sender creates his public parameter for the session, using  $p$  and  $g$  of the receiver, and embeds it without use of a steganographic key. Thereafter, the secret Diffie-Hellman session key can also be computed and used as steganographic key by the receiver.

## 4.1.3 Basics of security

[ZFPW\_97]

### 4.1.3.1 Attack-aims and -successes

### 4.1.3.2 Types of attacks

### 4.1.3.3 Stegoanalysis leads to improved steganography

Scientifically interesting (and significant for the regulation of steganography, cp. §9) is, that every method to detect steganography can be immediately used to improve the stegosystem. This already in [MöPS\_94, chapter 4.3.2] mentioned fact will now be displayed and researched in more detail. For this, mostly steganographic systems of secrecy will be considered.

First, a construction method for the covering model, i.e., for  $S$  and  $S^{-1}$ , is given. The reverse operation is not changed thereby - it had already been assigned in the non-advanced stegosystem to only give an *emb* to the outside, if one was actually embedded. The reverse operation needs enough redundancy, to decide this - with sufficiently high probability - correctly.

Afterwards, constructions of the embedding model, i.e.,  $E$  and  $E^{-1}$ , are developed. The reverse operations each need to be changed here as well.

#### 4.1.3.3.1 construction of $S$ and $S^{-1}$

Let  $S$  be the algorithm "hide" and  $S^{-1}$  the reverse operation to  $S$ , i.e., an algorithm which extracts the embedded data  $emb$ . If nothing was embedded,  $S^{-1}$  gives the result *empty*. Let  $R$  be a guessing algorithm, which guesses for the input *stego* with overwhelming probability *semi-correctly*, if something is hidden in *stego* or not. Let **Semi-correctly** mean: if  $R$  guesses "something\_hidden", then this is correct. But if  $R$  guesses "nothing\_hidden", this can also mean, that something is hidden, but  $R$  does not notice it<sup>6</sup>.

Then an advanced algorithm "hide"  $S'$  and the advanced reverse operation  $S'^{-1}$  can be developed in this way:

```
function S' (in: Source, emb; out: stego);
    stego := S(Source, emb);
    if R(stego) = something_hidden then stego := S(Source, empty);
```

The reverse operation  $S'^{-1}$  is exactly  $S^{-1}$ .

The stegokey was omitted in the notation, as it is not important for the construction. It is not even important whether symmetric or, in case that it exists, asymmetric steganography is used. Also, nothing changes if the guessing algorithm  $R$  gets the additional input parameter *source*, meaning that the attacker changes from a *stego*-attack to a *source-stego*-attack. Furthermore, nothing changes if  $R$  gets not only the current *stego* as input, but also all previous *stegos* (as well as all previous *sources*).

For the hidden transfer of information, the stegosystem (thus  $S$  and  $S^{-1}$  or  $S'$  and  $S'^{-1}$ , respectively) is repeatedly looped until all that information which needs to be transferred secretly is completely transferred. Therefore,  $S'$  needs an additional output parameter which indicates whether the transmission succeeded or needs to be tried again with the same value for *emb*, but with another value for *source*.

**Claim:** For the advanced stegosystem  $S'$  and  $S'^{-1}$  the following holds true:

1. The receiver receives only the data *emb*, which was sent by the sender.
2. The data *emb* is better hidden than through the original stegosystem.
3. The usable bandwidth of the advanced stegosystem is smaller than the usable bandwidth of the original stegosystem.
4. If the usable bandwidth of the original stegosystem is larger than 0, and if it contains nondeterminism<sup>7</sup>, then the usable bandwidth of the advanced stegosystems

---

<sup>6</sup>Such a semi-correct algorithm is exactly, what would be needed to be presented to a court, to prove the use of illegal steganography. It does not necessarily need to detect all steganography, but if it states that steganography was used, then its claim must be correct. Otherwise, no judge would trust this algorithm.

<sup>7</sup>Nondeterminism from the attackers point of view is necessary, so that one can change beyond recognition for him and hence hide. If the stegosystem contains nondeterminism, then it exists in that way especially from the attacker's point of view, and hence one can change beyond recognition and hide. In short: The stegosystem is not made stupid then.

is also larger than 0 and contains the same nondeterminism. The receiver then receives the data  $emb$ , which the sender wants to send, after a finite number of tries.

5. If the effort for the initial stegosystem and  $R$  is acceptable, then the effort is also acceptable for the advanced stegosystem.

Proof:

1. If the original stegosystem has this property, then the advanced stegosystem also has it: Either the sender behaves as in the original stegosystem, then the receiving is exactly the same, or the sender sends a *stego* with nothing embedded. The original stegosystem must also cope with this.
2. As *stego* is only created by the original stegosystem, the advanced stegosystem hides at least as well as the original stegosystem. Since it was assumed that the original stegosystem still creates from time to time a *stego* which can be detected by  $R$ , this can no longer happen with the advanced stegosystem and the advanced stegosystem hides the data better.
3. This does not hold for every single case, though generally for the average case.
- 4.
- 5.

Critical notes to the individual points:

- 2:** Everything is fine as long as  $R$  works only on parts of *stego*, so that one can iterate. If this is not the case, perhaps because dependencies between different parts also need to be considered, then not even an “at least” can be proven let alone a “better”. Nevertheless, I believe that the statements hold for “the average case”.
- 3:** The corresponding holds for the bandwidth.

Research questions:

Can a secure stegosystem of maximum capacity be reached when starting with a stegosystem, which uses all nondeterminism, and using iteration of the construction? (With this there may be no mix up between the iteration of the call of the stegosystem for information transmission and the iteration of the construction to improve the stegosystem.)

#### 4.1.3.3.2 construction of $E$ and $E^{-1}$

So that also single bits can be embedded, the goal of  $E$  and  $E^{-1}$  is weaker than the goal of  $S$  and  $S^{-1}$ , where  $S^{-1}$  needs to decide whether something *was* hidden and only give the  $emb$  to the outside, if something *was* hidden. Therefore  $emb$  must be encoded by  $S$  redundantly, such that it contains at least a few 10 bits.

## 4 Basics in Steganography

The goal of  $E$  and  $E^{-1}$  is:

$E$  shall embed  $emb$ , if it possible to do so undetected.  $E^{-1}$  shall detect if it was *possible* to embed something. If something could be embedded,  $E^{-1}$  shall give the potentially embedded  $emb$  to the outside.

This interface of  $E$  and  $E^{-1}$  can be used efficiently: If the sender wants to transfer a message secretly, then he embeds it sequentially, i.e., whenever possible - otherwise he does nothing. This way, the receiver obtains every secretly transferred message as a compact bitstream, i.e., without being interrupted by meaningless bitstreams. If the sender does not have continuous secret message to transfer secretly, then the receiver gets meaningless bitstreams between the messages.

In the following, we consider the particular case, that  $emb$  is one bit. Thus the sender as well as the receiver can efficiently check whether or not *all* possible values of  $emb$  can be embedded, and if through this *all* messages can be compactly transferred. As in §4.1.3.3.1, let  $R$  be a guessing algorithm, which guesses for the input  $stego$  with overwhelming probability *semi-correctly*, whether something is embedded in  $stego$  or not. Let  $E'$  be the improved algorithm “embed” and  $E'^{-1}$  its reverse operation.

**The obvious construction (correct only if the embedding position is independent from the bit which needs to be embedded):**

The sender checks whether  $E(cover, 0)$  and  $E(cover, 1)$  are respectively **acceptable**, i.e., he checks whether the embedded value is from his point of view not detectable to a steganalyst, and he only embeds, if they are acceptable, i.e., he then sends  $stego := E(cover, emb)$  instead of  $cover$ .

An implementation of the improved embedding algorithm  $E'$  with the least possible uses of  $E$  and  $R$  is as follows:

```
function E'(in: cover, emb; out: stego);
    stego := E(cover, emb);
    stego_alternate := E(cover, not(emb));
    if R(stego) = something_hidden or R(stego_alternate) = something_hidden
        then stego := cover;
```

The receiver checks whether the received  $stego$  is an acceptable value<sup>8</sup> and then - if it is acceptable - computes  $emb := E^{-1}(stego)$ . Now the receiver has to also check whether the other possible value of the bit,  $not(emb)$ , could have also been embedded. Unfortunately, the receiver does not know  $cover$  and can hence not compute  $E(cover, not(emb))$ . In its place, he computes  $E(stego, not(emb))$ . If the equation

$$E(cover, not(emb)) = E(stego, not(emb)),$$

which can be equivalently rearranged by inserting  $stego = E(cover, emb)$  to

$$E(cover, not(emb)) = E(E(cover, emb), not(emb))<sup>9</sup>,$$

---

<sup>8</sup>If the stegosystem is good, then this should *always* be the case - independent of whether something was embedded or not.

<sup>9</sup>This equation says: If two inverse bits are embedded in succession, then the first embedding is overwritten by the second embedding, i.e., the embedding position does not depend on the value of the bit.

holds for the stegosystem, then the receiver can also check whether both possible values of the bit could have been embedded, and thus whether the sender could embed or not. So the receiver only gives the computed  $emb$  to the outside, if  $E(stego, not(emb))$  is an acceptable value for the stegotext.

An efficient implementation of the improved extraction algorithm  $E^{-1}$  is:

```
function invE'(in: stego; out: emb);
emb := invE(stego);
stego_alternate := E(stego, not(emb));
if R(stego_alternate) = something_hidden
then emb := empty;
```

The following question arises: Is the above mentioned goal for  $E$  and  $E^{-1}$  only achievable when the stegosystem fulfills the above written equations? The answer is a constructive “no”:

#### **The inefficient construction:**

The idea is that the sender and the receiver both check whether both the possible values of a bit can be embedded in  $stego$ . Only then is it assumed that the bit has been transmitted. In detail:

The sender checks whether  $E(cover, 0)$  and  $E(cover, 1)$  are both **acceptable**, i.e., he checks whether the embedded values are, from his point of view, undetectable for a stegoanalyst and only embeds if they are acceptable, i.e., he then sends  $stego := E(cover, emb)$  instead of  $cover$ . If  $E(cover, 0)$ , as well as  $E(cover, 1)$  are acceptable values, the sender assumes that  $emb$  has been transmitted, or he just sends  $emb$  again.

The implementation of the improved embedding algorithm  $E'$  with as few as possible uses of  $E$  and  $R$ , is as follows:

```
function E'(in: cover, emb; out: stego);
stego := E(cover,emb);
stego_alternate := E(cover, not(emb));
if R(stego) = something_hidden or R(stego_alternate) = something_hidden
then begin stego := cover; E'(next cover, emb) end
else
if R(E(stego, 0)) = something_hidden or R(E(stego, 1)) = something_hidden
then E'(next cover, emb);
```

The receiver receives  $stego'$  (this can be  $stego$  or  $cover$ ) and then he computes  $emb' := E^{-1}(stego')$ . The receiver checks whether or not  $E(stego', 0)$  and  $E(stego', 1)$  are both acceptable values for stegotext and then receives  $emb'$ , or otherwise nothing.

Why does this work?

**Case 1:** The sender sends  $stego$ , i.e.,  $stego' = stego$ . The receiver thus receives  $emb' = emb$  if and only if the sender assumes  $emb$  has transmitted.

**Case 2:** The sender sends  $cover$ , i.e.,  $stego' = cover$ . Then either  $E(cover, 0)$  or  $E(cover, 1)$  is not acceptable – or both. Because of  $stego' = cover$ , the same holds for the values  $E(stego', 0)$  and  $E(stego', 1)$  tested by the receiver.

## 4 Basics in Steganography

An implementation of the improved extraction algorithm  $E'^{-1}$  is as follows:

```
function invE'(in: stego; out: emb);
if R(E(stego, 0)) = something_hidden or R(E(stego, 1)) = something_hidden
then emb := empty else emb := invE(stego).
```

This inefficient construction omits transmission possibilities, in which  $E(cover, emb)$  is acceptable, but  $E(cover, \text{not}(emb))$  is not acceptable. Therefore the following, more efficient construction is better:

### The efficient construction:

The sender checks whether  $E(cover, emb)$  is **acceptable**, i.e., he checks whether the embedded value is, from his point of view, undetectable to a stegoanalyst, and he only embeds, if they are acceptable, i.e., he then sends  $stego := E(cover, emb)$  instead of  $cover$ . If  $E(\text{cover}, 0)$ , as well as  $E(\text{cover}, 1)$ , are acceptable values, the sender assumes that  $emb$  has been transmitted, or sends  $emb$  again.

An implementation of the improved embedding algorithm  $E'$ , with as few uses of  $E$  and  $R$  as possible, is as follows:

```
function E'(in: cover, emb; out: stego);
stego := E(cover,emb);
if R(stego) = something_hidden
then begin stego := cover; E'(next cover, emb) end
else
if R(E(stego, 0)) = something_hidden or R(E(stego, 1)) = something_hidden
then E'(next cover, emb);
```

The receiver receives  $stego'$  (this can be  $stego$  or  $cover$ ) and then he computes  $emb' := E^{-1}(stego')$ . The receiver checks whether or not  $E(stego', 0)$  and  $E(stego', 1)$  are both acceptable values for stegotext, and then receives  $emb'$ , or otherwise nothing. (This is exactly the same as above, so the above algorithm can be adopted).

Why does this work?

**Case 1:** The sender sends  $stego$ , i.e.,  $stego' = stego$ . The receiver thus receives  $emb' = emb$  if and only if the sender assumes  $emb$  has transmitted.

**Case 2:** The sender sends  $cover$ , i.e.,  $stego' = cover$ . Then either  $E(cover, 0)$  or  $E(cover, 1)$  is not acceptable (as one of these is  $E(cover, emb)$ , and is discarded by the sender as not acceptable), or both. Thus the receiver receives nothing, because – through  $stego' = cover$  – either  $E(stego', 0)$  or  $E(stego', 1)$  (or both) is a non-acceptable value for stegotext.

## 4.2 Information-theoretic secure steganography

### 4.2.1 Information theoretic secure steganographic secrecy

[KIPi\_97]

## 4.2.2 Information-theoretic secure steganographic authentication

Work paper from Herbert Klimant and Rudi Piotraschke

## 4.3 Two opposed stego-paradigms

### 4.3.1 Changing as little as possible

Least significant bit (LSB)

### 4.3.2 Simulating the normal process

[FrPf\_98]

## 4.4 Parity-encoding to improve concealment

Instead of choosing the bits where one embeds with the stegokey, Ross Anderson proposes to choose sets of bits and encode each bit that is to be embedded as their parity [Ande4\_96, AnPe\_98]. This gives more liberty to the embedding process and in this sense intensifies the concealment of the embedding.

The larger the sets of bits over which the parity is built are, the more uniformly distributed the parity is, and the less suspicious the embedding. Conversely, errors with the transmission of the stegotext then affect the embedded data with ever greater probability - be they accidental or specifically created to complicate steganography.

## 4.5 Steganography in digital telephone calls

[MöPS\_94, FJMPs\_96]

## 4.6 Steganography in pictures

[FrPf\_98]

## 4.7 Steganography in video conferences

[West\_97]



# 5 Security in Communication Networks

## 5.1 The problem

### 5.1.1 The social problem

With all communication networks, you should ask yourself how well all participants are protected from damage. The resulting protection goals (and protection mechanisms) can roughly be separated into two categories (cp. §1.2.4):

**Performing desired actions:** A participant can be harmed by refusing, delaying or altering a service, or by establishing a communication relationship in his name and costs without his approval. It can also be harmful to him in the case that he has to prove having sent or received a certain message. These are - in other words - the protection goals *integrity* and *availability* from §1.3.

To actually inflict this damage, a *modifying attack* needs to take place. In general these attacks can be spotted in the IT system (cp. §1.2.5). Or said the other way round: That the desired actions happened can be verified later on.

**Preventing undesired actions:** A participant can be harmed through observation of his communication. The observation may include the *message content* as well as the *communication circumstances*. This was already defined as protection goal *confidentiality* in §1.3.

To actually inflict damage, only an *observing attack* needs to take place. These attacks cannot generally be spotted, as mentioned in §1.2.5. Meeting the protection goal of confidentiality is not possible afterwards - so preventive measures are inevitable.

The measures for “doing desired actions” are explained in detail in [VoKe\_83]. They are almost independent from the measures for “preventing undesired actions”. We will only take a short look at them here.

Therefore we must first answer the question of what outcomes the development of networks mentioned above will have on “preventing undesired actions”.

In fig. 5.1 the final state of that development is shown: All services such as TV, radio, telephone, and the Internet are routed from the local loop (or another relay station) to the participant’s port via fiber optics.

This fiber optic string is non-ambiguously assigned to the participant (or his family or company). And because it is a switching network, only the data specific to him is transmitted. So the physical network address is a type of personal identification number

## 5 Security in Communication Networks

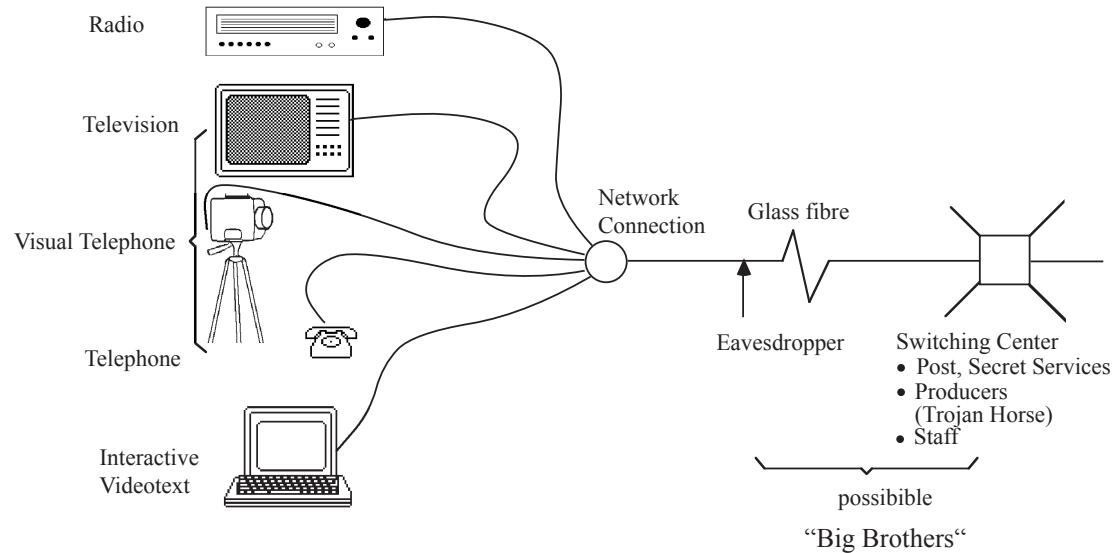


Figure 5.1: Observability of the user in an astral integrated broadband mediating network, which mediates all services.

(an identification for natural or legal persons) which makes it possible to collect data about that participant.

The information inside the fiber optic and the relay stations are *technically* made up of:

- the **payload data** (images, audio, text)
- the **network data** (addresses of the communication partners, the data volume transmitted, the service specifications, the connection time, and if mobile, the currents location)

The resulting person specific data can be *textually* divided into:

- **Content data**, that is the content of confidential (personal or business) messages such as telephone conversations or electronic mail.
- **Selection data**, that is information regarding the interest of the participant in non-confidential messages. These are observations such as which newspaper articles are being received, which database requests are being made, what he is watching on TV, or listening to on the radio.

*If a participant zaps through a TV program, the information about it is given to this TV and his network connection. This is then transmitting the new program instead of the old one.*

The chosen data not only characterizes a natural person but also corporations:

*Which special news articles and patents are requested gives hints about their current research and development interests.*

Before the introduction of public accessible databases (especially teletext) selection data could not be gained throughout communication networks.

- **Traffic data**, this is how long a participant communicated with someone. You can gain this data from the network data. For natural persons, these information are interesting towards developing a personality image (like his consuming habits, his friends, contact with police and medical institutions). Traffic data can harm not only personal rights of a natural person but also the business interests of a legal person. For instance, if the communication between two competitors suddenly increases they may be conducting a joint product development.

One way to get the data is to **wiretap** the participant's **connections** to his local relay station or the connections between the relay stations. The latter one does not allow tracing of the mass communication between the "provider" of TV and radio and the local relay station if you have a corresponding protocol. This also denies access to the individual communication of participants of the same local relay station area. But it allows the tracing of many participants who communicate beyond their local relay station.

### 5.1.2 information-theoretical problem and solution drafts

The protection goal confidentiality from §1.3 extended by one more item is:

**Preventing undesired actions:** Undesired actions are denied.

#### Protection goal confidentiality

- c1 *The message content* shall be kept under wraps except for the communication partners.
- c2 *Sender and/or receiver* of messages shall stay anonymous towards each other and *non participants* (like the carrier) *should not be able to observe*.
- c3 Neither potential communication partners nor non-participants (including the carrier) shall be able to obtain the *current location* of a participant's mobile node without their agreement.

**Doing desired actions:** Desired action shall take place.

#### Protection goal integrity

- i1 Manipulations of the *message's content* (incl. the *senders address*) shall be detected.
- i2 The recipient shall be able to prove that instance *x* has sent message *y* (and even the time of this) to a third person.

- i3 The sender shall be able to prove the *sending* of the correct content and if possible the receiving of the message.
- i4 Nobody can withhold<sup>1</sup> *hires* for services he used from the carrier. Conversely, the carrier can only demand hires for services correctly provided.

### Protection goal availability

- a1 The network allows communication between all partners who would *like* to communicate(unless it is forbidden to them: positive politics).

To give an overview, the protection goals are shown with their corresponding **protection mechanisms**:

#### *Preventing undesired actions*

##### Protection mechanisms for confidentiality

- c1 Usage of one or more encryption systems as *end to end encryption* (cp. §5.2.1.2).
- c2 Techniques within the communication network to protect network data (cp. §5.3 to §5.6.1).
- c3 Techniques within the communication network to protect the mobility behavior of the participant (cp. §5.7).

#### *Doing desired actions*

##### Protection mechanisms for integrity

- i1 Usage of one or more authentication systems.
- i2 Messages are digitally signed by the initiator. The test keys are publicly available.

If signatures should stay valid even if the secret key becomes known (which may even happen through the owner himself: to deny the agreements made with the signature) the signature should be signed with a time stamp by one or more institutions (**electronic notary**) and with their regular signing keys. By doing so, the validity of all signatures made earlier remains even if the secret key is known. Finally, no new signatures with that key should be authenticated by the notaries.

What happens if the signature system used can be broken in the near future is discussed in exercise 3-15b).

Besides the protection of the validity of the digital signature, the time stamp proves that the message was sent after the time of the stamp.

---

<sup>1</sup>This claim is stronger than what the communication network must support. It would be sufficient that the carrier can *prove* the correct provision of his services. How and when a payment is realized is actually not a topic here. But since the claim of confidentiality c2 for collecting the hires seems to make this difficult this stronger claim is required. It is shown in §6 how this can be achieved.

- i3 The communication network (or the partner) gives digital receipts of the message transported (or received). Those receipts may include a time stamp.
- i4 With digital payment systems (§6) hires are paid for services used.

### Protection mechanisms for availability

- a1 Diversive networks (different transmission and relay systems); fair proportioning of all resources.

The **goal** of the systems used for preventing undesired actions is to make it impossible for an **attacker** to gather any sensible data: The communication should be almost *unobservable* towards non-participants and *anonymous* towards communication partners. To make it impossible to gather any unspecific knowledge about a person, the single network events should stay *unrelated*.

Furthermore, you have to show that techniques for preventing undesired actions will not make desired actions unavailable (cp. §5.1.1). Due to their importance for comprehension, I would like to explain the terms introduced more precisely.

Because a protection from an omnipotent attacker who controls all connections, relay stations, and all communication partners is impossible from within the system, all measures are only approximations towards a perfect protection. The approximation is determined by the **strength of the attacker** in a **attacker model**: *How much did the attacker spread and how much computation power* is available to him (cp. §1.2.5 and §3).

(I want to clearly point out that the following discussion is made from an academic point of view according all assumptions about organizations and groups of people. Therefore we will only discuss *if* an organization or a group of people could attack, and not if they did so.)

An **event**  $E$  (e. g., sending a message or making a business) is **unobservable** relative to an attacker  $A$  if the probability of the occurrence of  $E$  as a possible observation  $O$  to  $A$  is truly greater than 0 and truly smaller than 1. Of course this only is true for attackers not involved in  $E$ . In mathematical terms: For  $A$  holds:  $(\forall B)(0 < P(E|O) < 1)$ .

An event is *perfectly unobservable* relative to an attacker  $A$  if the probability of the occurrence of  $E$  is equal before and after every observation  $O$ . For  $A$  holds  $(\forall B)(P(E) = P(E|O))$ . This means [Sha1\_49] that the observation does not give  $A$  any additional information about  $E$ .

An **instance** (e. g., person) in a role  $R$  is **anonymous** relative to  $E$  (e. g., as sender of a message, buyer in a business) and an attacker  $A$ , if for every instance not cooperating with  $A$ , it has the role  $R$  in  $E$  with a probability truly greater than 0 and truly smaller than 1 after every observation from  $A$ . This is also true for attackers involved in  $E$ .

An instance in a role  $R$  relative to an event  $E$  and an attacker  $A$  is *perfectly anonymous*, if for every instance not cooperating with  $A$  it has the role  $R$  in  $E$  with the same

probability before and after the observation from  $A$ . This means [Sha1\_49] that the observation does not give  $A$  any additional information about who has the role  $R$  in  $E$ .

Two **events**  $E$  and  $F$  relative to an attribute  $M$  (like the sender address or the transaction they belong to) and an attacker  $A$  are **unrelated** (unchained) if the probability that they do match in  $M$  is truly greater than 0 and truly smaller than 1 after every observation by  $A$ . This is also true for attackers involved in  $E$ .

Two **events**  $E$  and  $F$  relative to an attribute  $M$  and an attacker  $A$  are *perfectly* unrelated if the probability that they do match in  $M$  is equal before and after every observation by  $A$ . This means [Sha1\_49] that the observation does not give  $A$  any additional information about whether the events do match in  $M$ .

Of all the three definitions of unobservability, anonymity, and unlinkability, unlinkability is the most common:

Unobservability of events can be viewed as unlinkability of observations and the events behind them.

Anonymity can be viewed as unlinkability between instances and events.

The **reliability of the unlinkability, unobservability, and anonymity** is determined by how *strong* the attacker is bound to the previous definitions.

Unobservability and anonymity can be additionally parameterized by making them apply only to certain classes of events (message types) and instances (parts of the communication network (cp. §5.5)). In the previous definitions no classes were defined. Thus they cover the greatest possible degree of unobservability and anonymity. Here is an example of a classified definition:

An **event**  $E$  is (*perfectly*) **unobservable** relative to an attacker  $A$  and a given classification of events, if the conditional probability of the occurrence of  $E$  after every observation by  $A$  of  $B$  is truly greater than 0 and truly smaller than 1.

Both can of course only apply for attackers *not involved* in  $E$ . It is in the case of unobservability as well as perfect unobservability, that for the attacker the probability of some events may change due to his observations. The attacker may gain information about classes, but in the second case nothing about single events within the classes.

## 5.2 Usage and limits of encryption in communication networks

### 5.2.1 Usage of encryption in communication networks

For the usage of cryptographic systems for protecting communication (mainly for concealment, but also for authentication and integrity) you have two strategies at hand which both have disadvantages: link by link encryption and end-to-end encryption.

### 5.2.1.1 Link by link encryption

The first strategy is to encrypt *all* data between two neighboring network nodes [Bara\_64][Denn\_82][VoKe\_83] [DaPr\_89].

There should be a *permanent character stream*, so the observer can not know when a message is sent. A permanent character stream which is statically assigned (e.g., end-to-end connections or radio links) can be established without any extra efforts.

Because of the reasons explained in §3, you could and should use a secure *stream cipher*: If a block cipher is used the observer could recognize that certain message fragments are repeated if the key was not changed properly. The observer could *relate* some messages with certain events (cp. §5.1.2).

If a permanent stream of characters (encrypted with a secure stream cipher) is transmitted the observer does not gain any information. If and for what reason the messages are transmitted is *perfectly unobservable* (cp. §5.1.2).

The disadvantages of link by link encryption are

- Inside the relay stations all data is unencrypted. Of all the attackers only the observers of the links are intercepted.
- As in fig. 5.2, you can see the final stages of a communication network where the participant is connected with fiber optics (at least 560 Mbit/s) to the relay station. This is beyond what today's cryptodevices (they need to be reliable and cheap enough) can achieve (cp. §3). It would be an advantage if high resolution TV (HDTV) which is actually not the subject of interest to observers, would not be encrypted. Only the network and user specific data would be encrypted. Today's PAL TV needs about 34 Mbit/s (compressed) and HDTV needs about four times that much.

In any case, neighboring network nodes will need compatible and powerful cryptographic systems (self-synchronizing stream ciphers) that are directly hardware implemented. Because the connections of the network nodes are relatively statical a symmetric cryptographic system would be sufficient.

### 5.2.1.2 End-to-end encryption

The second strategy is to encrypt the data transmissions between the two participants directly [Bara\_64][Denn\_82][VoKe\_83] [DaPr\_89], so the relay stations can not interpret them (fig. 5.3).

Because of the reasons explained for link to link encryption, the end-to-end encryption should also use a permanent character stream (at least for the time the virtual connection exists) with stream ciphers. The disadvantages of end-to-end encryption are:

- The end-to-end encryption only encrypts the payload data but not the *network data*.

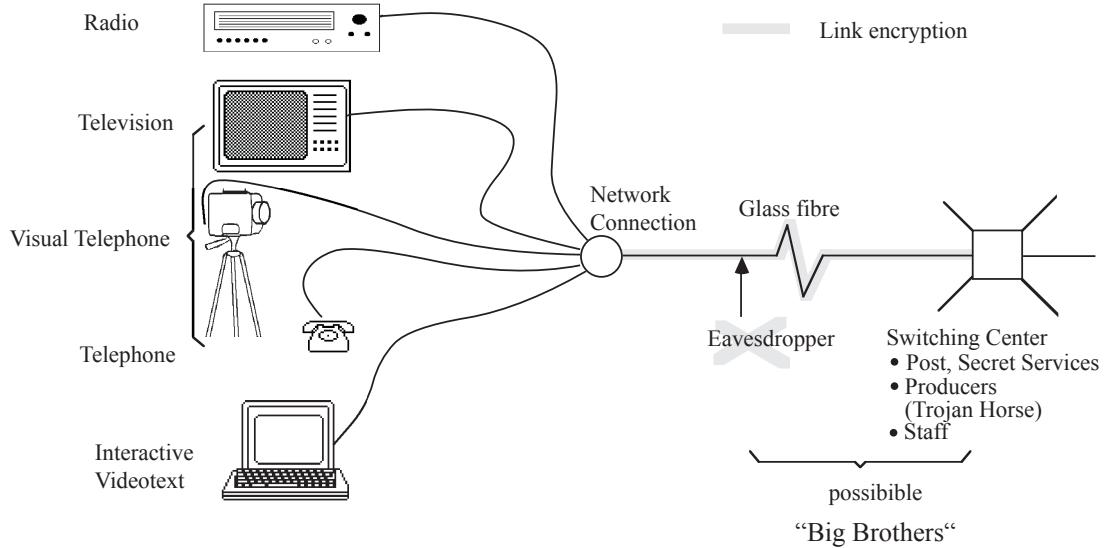


Figure 5.2: Link encryption between network connection and switching center

- Someone who already knows the payload data can now obtain the *selection data* (cp. §5.1.1) from the network data. This may lead to further relations between the traffic data and selection data.

Consequently this technique can not be employed as protection from the carrier and others that have access to the carrier's network when they are at the same time communication partners, or if the payload data is centrally stored in a database. The payload data may be stored encrypted in the database. But this is only useful if the attacker (or the carrier and the others that have access to the database) does not possess the corresponding key and can not obtain it from somewhere. The latter one seems quite unrealistic except for small and closed user groups because an attacker can easily infiltrate a large closed group in the form of a dummy. Inside a large open group the attacker camouflages himself as a normal participant or can collaborate with someone who has access to the data of the relay stations. This could originate from an intelligence service that would gain the network data from the carrier and the payload data from content providers or network participants.

Besides, it is a trivial thought to want to protect oneself with end-to-end encryption from communication partners.

- It is quite difficult to encrypt TV (esp. HDTV) to protect the selection data inside the relay stations using end-to-end encryption in the same way as it would be with link by link encryption.

If end-to-end encryption between any communication partners is desired, they do need a pairwise compatible cryptographic system. If broadband communication is required,

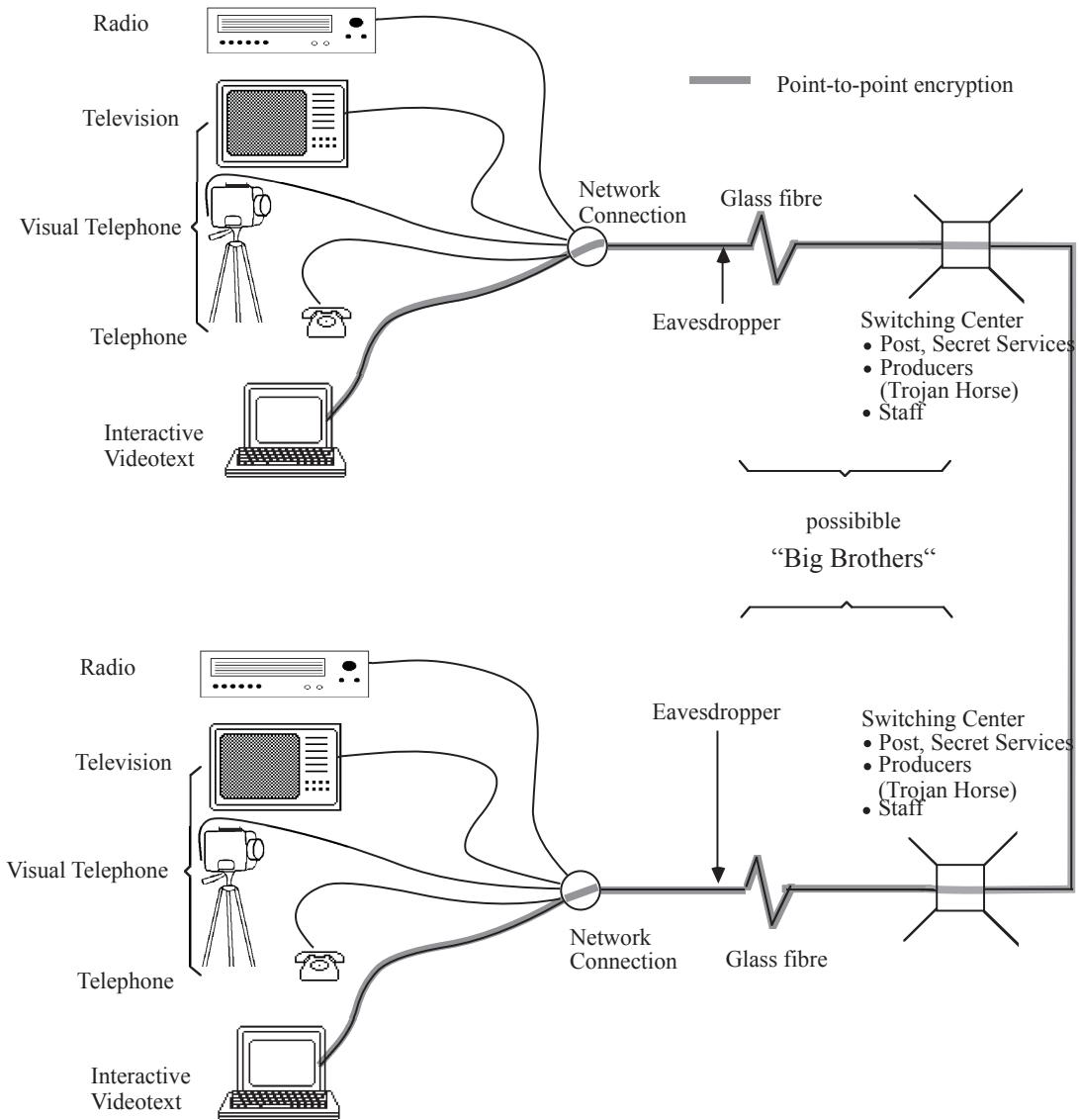


Figure 5.3: Point-to-point encryption between user stations

then at least one cryptographic system needs to be implemented into hardware (cp. §3). Because of the dynamic nature of the communication relations, you can not assign keys statically to every possible connection (that are  $n \cdot (n - 1)/2$  for  $n$  participants). Because of this, there should be a formal standardization of an asymmetric cryptographic system for key exchange and authentication as well as a fast symmetric encryption system for encryption and decryption of payload data. This should form an **open** system [PWP\_90][WaPP\_87] regarding data protection and predictability of legal decisions issues (as it is planned for ISDN). Only on such a basement of standards can cryptographic systems be developed that are efficient and allow *anybody to communicate with everybody else*.

Without such a standardization, confidentiality and integrity can only be achieved within **closed** user groups that have agreed on one cryptographic system. At this point, I want to emphasize what tremendous consequences the plans of the NSA<sup>2</sup> (ZSI<sup>3</sup>) of enforcing the usage of a secret concealment system in the public would have.

### 5.2.2 Limits of encryption in communication networks

Even if (as seen in fig. 5.4) link by link and end-to-end encryption is used [Bara\_64, Denn\_82, VoKe\_83] [DaPr\_89] all disadvantages of the end-to-end encryption mentioned in §5.2.1.2 remain. Only the first disadvantage is softened because observers of the connections will not gain any information from the link by link encryption.

Even the best known cryptographic systems will not grant sufficient and **reliable** protection against many attackers on the relay centrals and the communication partners, at least not with reasonable effort. Especially in pure link by link encryption systems used as service integrated system, you will have to face long-term problems regarding your information freedom.

This is why we try to find other ways to achieve the security still lacking. But you will have to take care to not let other entities become potential attackers.

From a technical point of view, the main goal is to protect the traffic and network data from the carrier and to protect your identity from other communication partners. With this, traffic and selection data is not only protected from them but also from attackers who do not have more access than the carriers and the other participants. Although encryption can not achieve the protection goals in the area of confidentiality inside a communication network, encryption gives us the fundamental techniques to describe protection measures for confidentiality.

The two following sections describe the fundamental techniques of traffic protection and data selection from attackers in relay stations and communication partners:

---

<sup>2</sup>National Security Agency; the USA's intelligence service for cryptographic systems and electronic surveillance

<sup>3</sup>Zentralstelle fuer Sicherheit und Informationstechnik (central agency for security and information processing systems); today BSI: Bundesamt fuer Sicherheit und Infomationstechnik, run by the department of the Interior (Federal agency for security and information processing systems)

## 5.2 Usage and limits of encryption in communication networks

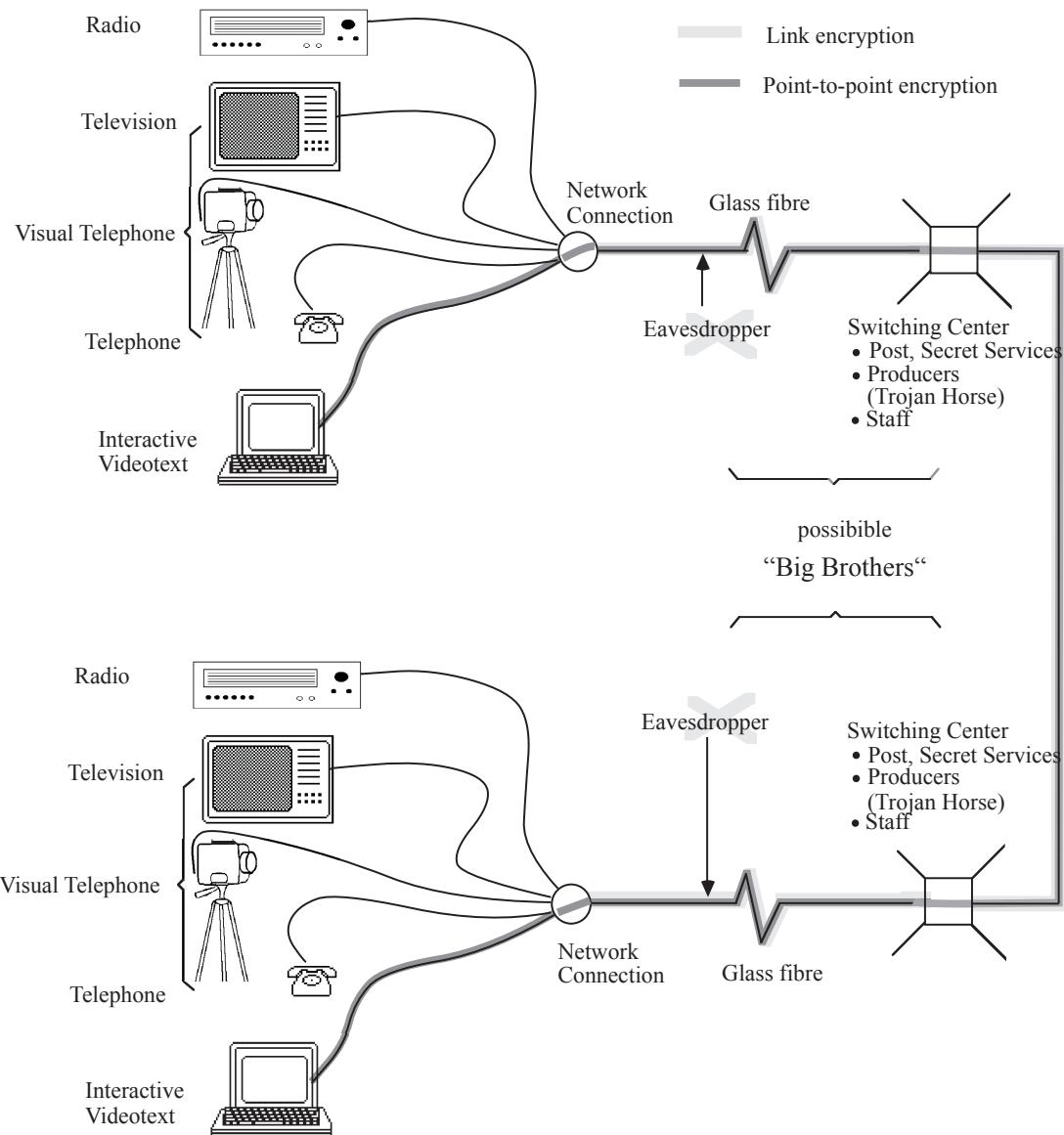


Figure 5.4: Point-to-point encryption between user stations and link encryption between network connections and switching centers, and between switching centers

Protection measures that are settled outside the communication network (which therefore every user meets individually) are described in §5.3. It will also be prove that they are not sufficient.

Protection measures which are settled inside the communication network (which therefore alter the traffic inside the network, but not the usage of the network) are described in §5.4.

## 5.3 Basic measures settled outside the communication network

With protection measures outside the communication network, the sender and receiver addresses are still visible as network data inside the network as well as, naturally, the time when the message goes through the network. You will have to prevent the extraction of traffic and selection data from that.

### 5.3.1 Public nodes

If there are public nodes in the communication network (like public telephone stations) or if the network can be reached via another network which has public nodes, the following measures are practical:

The sender and receiver addresses become meaningless if **various public nodes** are used. But this is only true if you do not have to identify yourself, aka can stay anonymous, for access control or payment issues [Pfit\_85].

The application and effects of the protection measure “using different public nodes” are strongly limited:

Who wants to leave the bureau or apartment for every single phone call (not to mention switching the public phone every time to make a time relation (cp. §5.3.2) difficult)? Who wants to go to a public telephone at an agreed point of time to receive the return call (especially since the location of the telephone is clear through conventional observation)?

However, even if all those questions were to be answered positively, the effects would be limited: Even if everybody switched the telephone with each call it would remain a strong locality (time effort is too big). Even if a *single* person would be anonymous in a group of a thousand of his *communication relations*, they would make it possible to relate sender and receiver data, which could lead to identification. Not to mention the relation between the location of his apartment and the telephone.

### 5.3.2 Time independent processing

The **time** when a message is in a communication network becomes almost meaningless if a network node would **request** information not when the user wants it but at some randomly chosen point of time before the request [Pfi1\_83, page 67,68][Pfit\_83].

*If you want to read a newspaper your network station could receive it when it is printed or when transmission is especially cheap. It would then save it for later reading.*

This protection measure prevents carriers implying the communication partner from time and context of the transmission:

*When a newspaper is requested during the day and you know that of the 10 potential customers 9 work the day and one works the night shift, it stands to reason the night worker is the reader.*

And as well as, your user station could send information **at any randomly chosen point of time**.

If the carrier can discern through other methods which persons are communicating with each other, this measure at least makes it difficult to establish an order of events.

Of course this measure does not work with real-time applications such as the telephone. For all others it makes it difficult to relate network events with other network events as result of a specific person's actions (cp. §5.1.2).

### 5.3.3 Local choice

To protect selection data you can **request information in larger chunks and pick the information that you are interested in later**:

*If a reader orders multiple newspapers of different political directions instead of a specific article, then no one could imply the political interests and opinions of the reader.*

With local choice you can even conceal from your communication partner your interests and how you learn and react, as well as much more.

Requesting large blocks of information from the communication partner is useful as protection for information that is transmitted unencrypted or when the carrier and the partner are identical or are working together.

In future communication networks where there is enough bandwidth this measure will have barely any real costs if the user requests already provided information. But it is a matter of the payment procedure if the user will profit from this (cp. §5.6.2). Even if no acceptable accounting procedure is found to get many newspapers of different political character from different publisher for the price of one, it should be possible today to buy complete newspapers and get them transmitted.

Local choice was invented separately at least three times: First of all with only **one** information provider and without a general classification in the context of BTX [BGJK\_81, page 153,159], in [Pfit\_83] and finally together with timely unrelated processing in [GILB\_85].

## 5.4 Basic measures settled inside the communication network

In contrast to the measures mentioned in §5.3, this section displays measures that reduce the amount of network data inside the communication network (cp. §5.1.1).

With that you could conceal the communication relations between the participants from attackers (evil neighbors, the carrier, producers of software and hardware). That makes the capturing of traffic and selection data impossible.

The goals are:

- The usage of the network should not change nor get more complicated.
- Every participant shall have the opportunity to validate the confidentiality of the system.

Because a protection from an omnipotent attacker that controls all connections, relay stations, and all communication partners is impossible from within the system, all measures are only approximations towards a perfect protection. The approximation is - as mentioned in §5.1.2 - determined by the strength of the attacker in a attacker model.

If a communication network is hiding who sends and receives for all network events it is called *anonymous*. Now we are speaking of **anonymous communication**.

Most of the known systems shown here protect from all attackers seen as realistic, by protecting either

- the communication relation/connection or
- who is sending or
- who is receiving.

Against significantly weaker attackers those basic measures grant a complete protection: Sender *and* receiver are protected (and that protects all communication relations). Against significantly stronger attackers they do not give any protection. Because of this canonical (and therefore independent from details of a realistic aggression scenario) definition, the following sections cover these three options.

#### 5.4.1 Broadcasting: The protection of the receiver

By having the communication network send all information to all participants the receiver can be protected from the communication network and the communication partner [FaLa\_75].

Broadcasting allows the analogy to a perfect (information-theoretical) concealment system (cp. §3). This means, according to the definition of the perfect anonymity from §5.1.2 and the information-theoretical from §3, that, for an attacker who controls all connections and some user stations, for all messages the probability that one station has received the message is the same for all stations not under his control before and after his surveillance. The attacker does not gain any new information about which uncontrolled station has actually received the message. It is even impossible for the attacker to observe whether the message was received at all. So broadcasting guarantees *perfect information-theoretical unobservability* of the receiver. Furthermore, broadcasting does not relate any traffic events so that we even have *perfect information-theoretical unlinkability*.

### 5.4.1.1 Modifying attacks on broadcasting

The realization of broadcasting has no conceptional problems as long as only *observing* attacks are considered. In other case:

- 1) A modifying attacker can *deny the service*. Here the usual fault tolerance techniques help, and if necessary combined with changing the carrier if important subsystems “randomly” fail too often.
- 2) A modifying attacker can *deanonymize some participants at least partially by interrupting the consistency*. For example if the attacker is the sender of a message  $M$  to which the recipient will answer, he wants to deanonymize: He sends  $M$  and prevents it from sending to all but one user station. If he does not get an answer he tries again with another user station. If you expect not just *one* answer (if this simple dialog is part of longer one) you could limit the amount of testings to the logarithm of the amount of user stations: The broadcasting is interrupted for the half of the other user stations. If he receives an answer he halves the group that contained the message. In the opposite case he halves its complement of the previous step [Waid\_90][WaPf1\_89].

This attack can be thwarted by either analogue or digital measures. An **analogue measure** could consist of the usage of media that *guarantees the consistency of the broadcasting (non cellular radio and satellite networks are of this kind)*. Because such an attribute is hard to prove (cp. §2) and radio connections are strongly interference prone, digital measure must be taken into account. **Digital measures** can *only spot inconsistent broadcasting* but not prevent (the attacker could cut through the wires or damage the radio antenna). The recognition of inconsistent broadcasting is either *complexity theoretical* by adding sequence numbers or timestamps to the message (so they become digitally signed by stations that will not cooperate ???) [PfPW1\_89]. Or there can even be achieved *information-theoretical* by using superposed sending (cp. §5.4.5) with inclusion of the distributed messages into the keygeneration (cp. §5.4.5.3) [Waid\_90][WaPf1\_89].

### 5.4.1.2 Implicit addressing

If you want to use broadcasting on routing services every node must decide through an attribute named **implicit address** which messages are actually for it. In contrast to an **explicit address**, an implicit address does not contain any information about the location of the recipient (nor about how he can be found).

If an address can only be interpreted correctly by the right recipient it is called **invisible addressing**. If an address can be interpreted by everybody (and therefore can be tested for identity with other addresses) it is called **visible addressing** [Waid\_85]. With visible addressing, messages with the same address are always *linkable* to the recipient (whose identity is equal to his address).

#### 5.4.1.2.1 Implementation of invisible implicit addressing

The usual implementation of invisible implicit addressing employs redundancy within the message content and an asymmetric encryption system (cp. §3). Every message is encrypted completely or partially with the encryption key of the addressed participant (the first option is an end-to-end encryption). After the decryption with the corresponding key the user station of the participant addressed can determine (with the redundancy inside the message) whether the message was designated for him. Because the implementation of asymmetric encryption systems is pretty complex and slow (with a limited implementation effort (cp. §3)) and every station has to decrypt all messages, this method is much too complex [Ken1\_81]. But it can be improved after the communication begins by exchanging a secret key using a fast *symmetric encryption system* [Karg\_77, page 111, 112] (instead of an asymmetric one): Consider a message that begins with a bit that determines whether symmetric or asymmetric encryption was used for addressing. For the latter case and for long communication relations you may save as much effort (asymtotically) as the symmetric concealation is more efficient than the asymmetric is easy to implement.

If the recipient uses more than one invisible and implicit address he must decrypt all messages with all the decryption keys. While participants using asymmetric encryption for addressing probably only have a few keys, the amount is dramatically larger if using symmetric encryption - which weakens the advantage of symmetric encryption (and may result in additional costs).

The unlinkability of messages using invisible and implicit addressing is only as secure as the most unsecure encryption system for addressing.

Instead of a symmetric encryption system you may use a *symmetric authentication system*: Instead of encoding messages with redundancy and encrypting them afterwards (and test for the redundancy when decrypting) messages are authenticated symmetrically. So the MAC is the appended redundancy. The potential addressees check if the message was authenticated correctly from their point of view. If so, then the message was for them.

#### 5.4.1.2.2 Equivalence of invisible and implicit addressing and encryption

The first invisible addressing of the message between two parties that have not exchanged a secret key is not unlinkable (but more anonymous than with the most secure asymmetric encryption system): With every opportunity for invisible addressing you can produce the functionality of an asymmetric encryption system by transmitting *The secret message is coded by making each message addressed to you translate into a 1 and every message not addressed to you a 0. <addressed message 0>, <addressed message 1>... unencrypted* [Pf1\_85, page 9][PfWa\_86]. This concealment protocol (that is based on receiver anonymity) corresponds to the concealment protocol (that is based on sender anonymity) from Alpern and Schneider [AlSc\_83].

If you assume that both parties have exchanged a secret key, every invisible addressing provides the functionality of symmetric encryption. Both proofs together (with the

		address distribution		
		public address (e.g., phone book)	private address (for designated communication partners only)	
addressing type	implicit	invisible	very costly but necessary to establish contact	costly
	address	visible	should not be used	change after use
	explicit	address	saves bandwidth, strongly dissuaded	saves bandwidth, strongly dissuaded

Figure 5.5: Evaluation of efficiency and unobservability of the addressing type/address distribution combination

implementation shown above of invisible implicit addressing with encryption systems) show that addressing and encryption (symmetric as well as asymmetric) are equivalent.

With these proofs natural limits occur towards efficiency. That means that encryption and invisible addressing are “equally” efficient for the asymmetric as well as for symmetric cases.

#### 5.4.1.2.3 Implementation of visible and implicit addressing

Visible addressing is easier to implement than invisible. You could just give a message an address field and the participants create random numbers as addresses. A station then only must compare the address field with its own addresses. That can be done with an *associative memory* even for large quantities of addresses. Address recognition is far more easier for visible than for invisible addressing.

#### 5.4.1.2.4 Address distribution

For both kinds of addressing you can distinguish between public and private addresses. **Public addresses** are publicly available address indexes (such as the “white pages”). They are used for the first contact. **Private addresses** are assigned to single communication partners. That can be done either outside the communication network or inside as the “comes from” statement in messages. It is pointed out that the categories “public address” and “private address” do not say anything about people behind: Public addresses can be anonymous (at least before usage) from all instances except the receiver. For private addresses it is possible to assign them to a single person.

With visible addressing the network can gain information about the recipient of the message. This also can happen for private addresses if the same address is used several times. So messages that are otherwise unrelated become related because of the equal recipient. The usage of public addresses repeatedly must be avoided by continuous generating and transmitting (e.g., with a generating algorithm). The generating algorithm

“Use the next  $n$  characters of randomly created one-time key that was transmitted with a perfect information theoretical security and integrity keeping method before” allows *perfect information-theoretical unlinkability and anonymity* from non-participants. With the procedure mentioned here anonymity can not be actually achieved due to organizational reasons (cp. §3 for perfect information-theoretical security). In §5.4.5.4 a method for a pairwise superposed reception that bases on sender anonymity for exchanging key is described. If the sender anonymity is information theoretical the key exchange is done with guaranteeing concealment and integrity. Because information-theoretical sender anonymity is possible but a very big effort this key exchange method is quite impractical. In contrast, if the one-time key is generated with a cryptographically strong pseudo random bitstream generator (cp. §3.4.3) you can preserve *perfect information-theoretical unlinkability and anonymity*. And so anonymity between the participants is not a problem at all because the initial value of the generator is exchanged with a perfect information-theoretical asymmetric encryption system.

Using public and visible addresses you can determine with which identifier the recipient is in the index. The usage of such addresses should be avoided as much as possible.

The protection and effort properties of the combinations of addressing kinds and address indexing are shown in fig. 5.5.

With broadcasting and implicit addressing the recipient of a message can be protected completely.

Unfortunately, not all types of communication networks are equally suitable for that. This is discussed in §5.4.9.

### 5.4.1.3 Fault tolerance and modifying attacks on broadcasting

Between broadcasting on the one hand and fault tolerance and modifying attacks on the other there are certain relevant interactions:

recipients that receive *error prone information units or do not receive them at all* should insist on a repeated but error free transmission (even if they have no need of the information) unless the reaction could reveal the recipient. But there are limits with the practical application:

On the one hand user stations may not receive all the information simultaneously. That makes it possible to miss some errors.

On the other hand there is no instance that could give an error free copy at all. The latter could be solved by temporarily saving all information units in one or more global memories. That would not limit confidentiality in any way.

Regarding the identification of error prone broadcasting (resp. altering attacks) remember what was said in §5.4.1.1.

For insisting on error free receipt a sending of the user station is necessary. With radio networks this could be used to easily identified [Pfit\_93][ThFe\_95]. Then you should evaluate the pros and cons.

As with encryption you must take care using *implicit addressing* so that if previous information units are lost or faked the address is still recognized correctly. That can

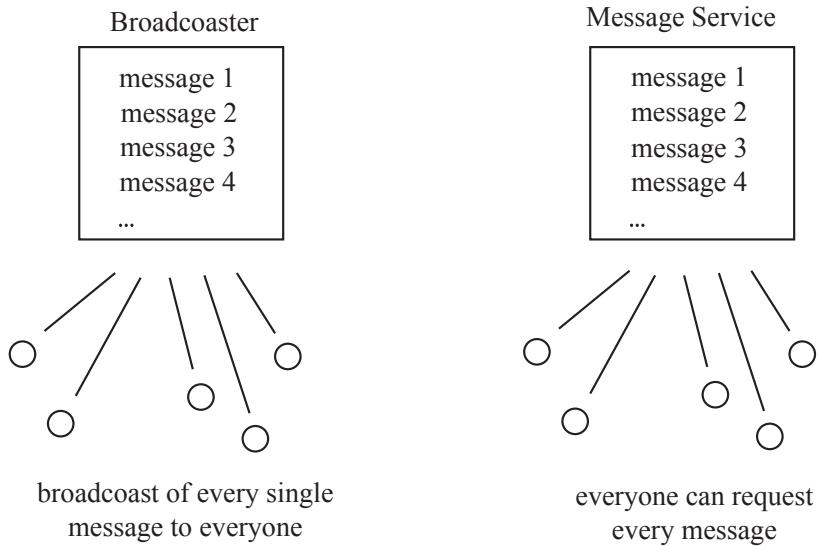


Figure 5.6: Broadcast vs. requesting

be achieved perfectly with a block cipher if using implicit addressing. Visible implicit addressing, using only addresses that have been used once, is not that easy. Remember that the addresses are written into the associative memory for comparison. If a message arrives with one of the addresses it is replaced by the following one (cp. §5.4.1.2). If you have a maximal error size (maximum of information units corrupted or lost in a row) of  $k$  you could write the  $k + 1$  next addresses instead of the next one only into the associative memory. But that would only help against temporary disturbances. With long term disturbances sender and receiver have to synchronize again - with messages with invisible implicit addressing.

### 5.4.2 Query and superpose: Protecting the receiver

By broadcasting the messages the receiver is well protected. Because everybody receives all messages nobody can say who has interest in which data. But broadcasting is very expensive in many communication networks for individual communication.

In [CoBi\_95] the technique **query and superpose** is described that also protects the selection data: The participants of a communication network can request all messages from a distributively implemented **message service** without indicating to others which messages was requested.

In more precise terms, the participants can query the messages *superposed* from the servers. Others are not able to find out which information was queried. That is because the superposed messages are *superposed locally* what produces the final message.

It works like this: The system creates  $m$  server with  $n$  memory cells each ( $n$  is the

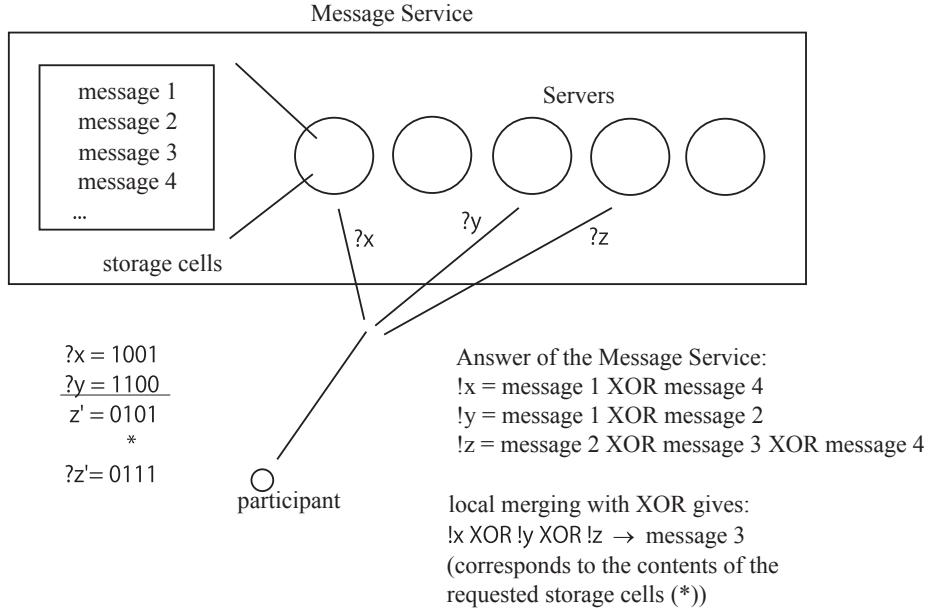


Figure 5.7: Example for a message service with  $m = 5$  and  $s = 3$ , requested is message 3

number of messages that can be saved). All server contain the same messages in the same order.

To cover which message the participants is interested in he queries different information from  $s$  servers ( $1 < s \leq m$ ). He does that by creating  $s-1$  query vectors which are random bit blocks of the length  $n$ . The  $s$ -th query vector is made by XOR over all  $s-1$  vectors. For reading the message the user is interested in from the  $t$ -th memory cell ( $1 \leq t \leq n$ ) the  $t$ -th bit of the calculated vector is negated.

To each  $s$  of the  $m$  servers each one of the query vectors is sent to. The desired messages (therefore the messages where there is a “1” in the query vector) are read from the servers to the memory cells, XORed, and sent to the participants. By XOR linking, the participant can now form the message he is interested in.

The communication between the participant and servers is protected. That is why external attackers do not have to be considered. And even with  $s-1$  servers collaborating, the selection data of the participant is protected:

claim Any  $s-1$  query vectors do not provide any information about the participant's message.

proof Before the negation of the  $t$ -th bit of the calculated  $s$ -th query vector this even holds for all query vectors together. But only if any random  $s-1$  vectors combine to random and unrelated bit strings. So any  $s-1$  vectors can be considered as the ones described above so that the information only goes into the  $s$ -th request vector.  $\square$

It holds that:

Every server used by the participant can protect his selection data.

In certain network environments this technique is more efficient than querying all messages. Though the intended recipient of a message needs to know that there is actually a message for him. And if so he needs to know in which memory cell. That can be done by giving the messages implicit addresses (cp. §5.4.1.2). The message service then distributes the implicit addresses followed by number of the memory cell. Or it puts the message into a cell all participants know. But this is only efficient if the messages are way longer than the numbers of the memory cells.

In [CoBi\_95][CoBi1\_95] this technique is described using mobile communication as example.

### 5.4.3 Meaningless messages: Weak protection of the sender and the receiver

The sender of a message protects himself by sending **dummy traffic**. Protecting means that he protects his identity when sending, and if explicit addressing is used the recipient is protected when receiving. If dummy traffic is used with explicit addressing the meaningless part can be removed automatically by the stations if it is marked (inside the encrypted message content). In case of broadcasting and implicit addressing with non-existent addresses all stations can delete those message contents. If meaningless messages too are sent the network can not determine when and how many meaningful messages a participant has sent [Chau\_81]. Symmetrically to that even in case of explicit addressing the network can not determine how many and when a participant has sent meaningful messages. Because deleting messages addressed to other stations is no more expensive than deleting messages addressed to itself, this technique was not mentioned in §5.4.1. Additionally the first one allows receiver anonymity<sup>4</sup> from the sender. If you do not like or can not afford complete broadcasting you should prefer partial broadcasting (multi-casting) to dummy traffic from the receivers point of view.

The technique of meaningless messages is very costly and does not fulfill the demand for anonymity of the sender for attacks where the receiver of meaningful messages is involved. Through the diction from §5.1.2 this technique only makes the *sending* (resp. *receiving*) *unobservable and unlinkable for non-participants*. Anonymity of the sender from participants can be achieved (against realistic attack models). Despite these obvious weak points of the technique of meaningless messages it is the only one for protecting traffic and selection data that is cited in the newer and widely spread publications about “Data protection and security in communication networks” (especially in the draft for extending the ISO model for “Open Communication”) too that is not from David Chaum or our research group [VoKe\_83, page 157f][VoKe\_85, page 18] [ISO7498-2\_87, page 17,

---

<sup>4</sup>Actually it is *receiver pseudo-anonymity* because the sender knows at least one pseudonym of the recipient.

24-29, 31-33, 36, 39f, 53, 56, 60f] [Rul1\_87][AbJe\_87, page 28][Folt\_87, page 45]. Of course there are widely spread publications that do not mention any solution (cp. [Brau\_87]). Do not these world-famous capacities know their special field - or are not they allow to do (cp. the actions from the NSA and ZSI in the field of cryptographics)?

If the sending (and receiving) of meaningful messages is combined with a technique for protecting the communication relation, it would also withstand attacks in which the carrier or the receiver of meaningful messages is involved. If the combination of the two methods shall protect the sender (or receiver) he must send or receive with a rate independent from sending and receiving wishes. That means that the bandwidth is distributed *statically* under the participants, which is quiet awkward. Furthermore you can only achieve *complexity theoretical* security of the sender. Nevertheless this method can be employed with certain boundary conditions (cp. §5.5.1).

Both classes of techniques for protecting the sender described in the two following sections do not have such drawbacks. They are efficient in a certain way by allowing a **dynamical** distribution of the bandwidth and avoiding meaningless messages and re-coding. And it is proven inside the *information-theoretical* world that the sender stays anonymous even if receiver and carrier work together.

Later on, with the superposed sending, a very powerful but also very expensive technique for realizing sender anonymity on any communication network desired is described.

Before this it is explained how a communication network can be set up so that realistic attacks can only observe the sending of single stations with a very big effort. That can be done by choosing an appropriate connection topology (ring or tree) so that observing is very costly. And if the effort is as big as any other form of individual observation then we can speak of privacy, which is inevitable for democracy.

#### **5.4.4 Unobservability of bordering connections and stations as well as digital signal regeneration: Protection of the sender**

The costly generating (and broadcasting and superposition) of keys and content data in §5.4.5 is necessary because we have to assume that an attacker wiretaps the inputs and outputs of *all* user stations as well as each of the bordering stations so that link by link encryption will not help.

The idea for techniques that are not that costly by far is to shape the communication network physically in a manner which makes equally hard to observe all inputs and outputs of all user stations as to observe the participant directly. This could also be done with link by link encryption but which is substantially more costly. Wherever an attacker can observe messages by listening to active lines or controlling stations, the message should originate from many (best from all stations) and addressed to many (best for all stations not under his control). Both can only apply if the attacker is not sender or receiver of the message.

An essential premise for that is **digital signal regeneration**: Every station regenerates all the signals (bits) received and sent so that they can not be distinguished from signals generated by this station. That also implies that two corresponding signals received from different stations can not be distinguished after regeneration.

But it is not required that corresponding signals sent from *different stations* can not be distinguished. While the latter, because of the analogue character of the sender ???.

In the two following subsections two examples of the this idea are shown.

The **RING-network** is an excellent example because it fulfills the maximum claim from above with minimal effort towards an attacker who controls a station. The effort is minimal in the way that it is required that every station receive the messages at least one time to achieve receiver anonymity (this lower bound is achieved by the efficient anonymous access system from §5.4.4.1.1). And because for reasons of the sender anonymity that every station sends at least with the summed up rate of its actual sending rate (with end-to-end duplex channels both can be halved without significant loss of anonymity as it will be shown on the pairwise superposed receiving on the DC-Network and on a special technique for switching of point to point duplex channels in [Pfit\_89, in §3.1.4]). This lower bound too is reached with the efficient anonymous access systems (besides a little overhead). Additionally it is positive that there is a lot of experience in the local sector with its physical structure (and at least in the USA there are plans for using MANs (Metropolitan Area Network [Sze\_85][Roch\_87])).

The **TREE-Network** is an example for pragmatic reasons: its physical structure is like today's broadband cable networks that can be set up for anonymous communication easily. The TREE-Network only offers anonymity against weaker attacker in comparison with the RING-Network, but it can be extended with *superposed sending*. That would make it grant strongest anonymity against any attacker. §5.4.5.5 shows that the structure of TREE-Networks is more suitable than RING-Network's.

#### 5.4.4.1 Circular cabling (RING-Network)

The most efficient way to physically shape a communication network, in the manner such that is not easier for the attacker to observe all inputs and outputs of a user station than to observe a participant directly, is to position the user stations in a circular form (as it has been done in local networks for a long time). Because of digital signal regeneration in every user station, both neighboring stations (circular ordered stations) would be needed to work together to observe the sending of the station between - alternatively by controlling the two connections. With constructional measures like the direct wiring of apartments, as well as the direct wiring of bordering realties [Pfi1\_83, page 18] and by using transmission medias that are hard to observe (like fiber optics) you can achieve that the latter requires equally hard physical measures as the direct observation of the participant within the apartments. If two ring neighbors are trying to observe the station between without collaborating, they will hardly observe anything because outgoing messages are encrypted and if implicit is used the addresses can not be interpreted.

So you may only expect attackers that have encircled a group of coherent stations. By

comparing the incoming and outgoing signal patterns they can only identify which were sent from members of the group and which were removed by the group. The attacker can not specify a member as an originator or a remover of signals.

If stations are sending uncoordinated on a joint distribution channel “ring” there may be collisions of messages. In contrast to the DC-Network from §5.4.5 and the TREE-Network from §5.4.4.2 the view of a sender and a receiver of a message (that collides in a RING-Network) is naturally not consistent - the collision may occur “after” the receiver so that only the sender will take notice (errors that may cause an inconsistent view in DC-Networks or TREE-Network are not considered here - they hopefully occur much more seldom than collisions and so they are handled separately in §5.4.7).

Under the (realistic) assumption that the encoding of two different messages will never be the same, the sender can see that the receiver received the message. This is true if the sender gets the message back after one round in the circle. By a global agreement that receivers will ignore messages already received (like time stamps) the sender can be assured through resending that the message was received, in that a consistent view about the receiving can be created for sender and receiver but not about the exact time. If this is required or if you want to prevent stochastical attacks to gain the sender by its sending rate, then you will have to employ more sophisticated access protocols or another configuration of the RING-Network.

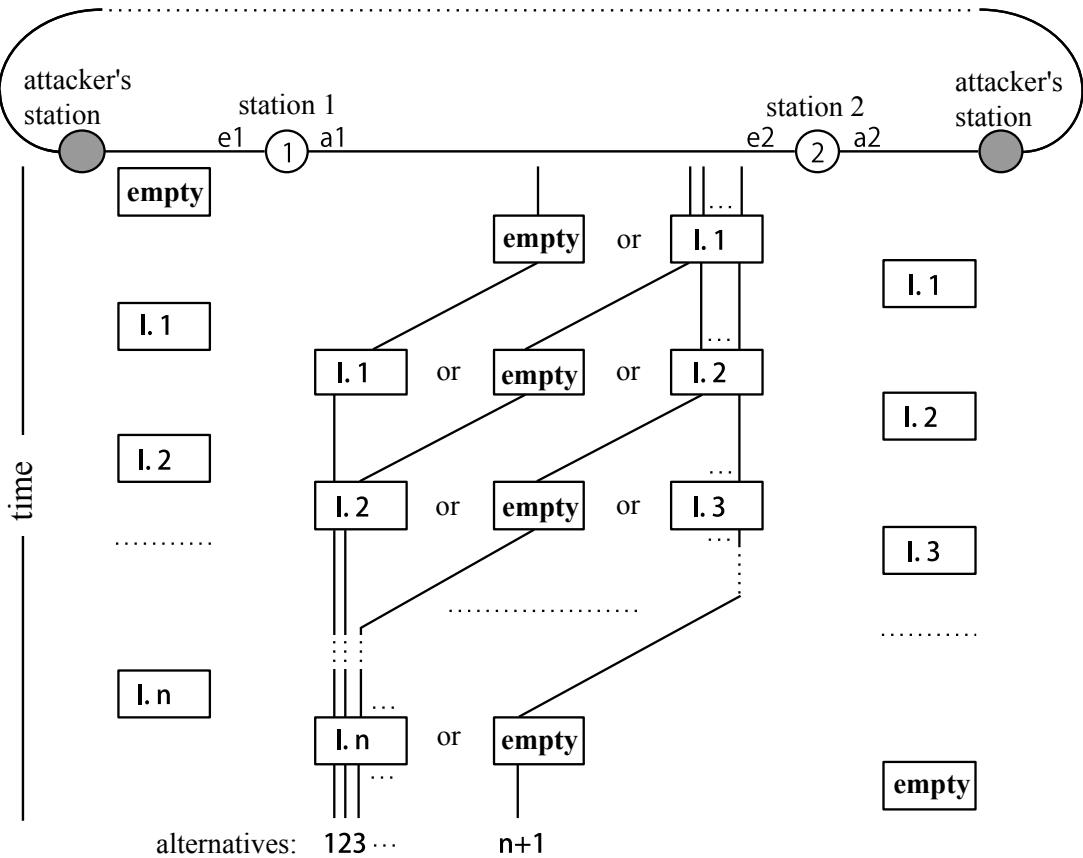
On the one hand, the ring offers (in contrast to DC-Networks) an efficient way to completely avoid collisions by special techniques for multiple access (distributed (§5.4.4.1.1) and unobservable polling [Pfitz\_85, page 19]). Sending of messages as well as switching of channels can be done with these access techniques. With end-to-end duplex channels you do not need the information to circle completely without a significant loss of anonymity. The receiver simply takes the information from the ring and replaces it with information for the communication partner. It doubles the capacity of the ring.

On the other hand, a ring offers two opportunities for uncoordinated sending to produce a consistent view for sender and receiver towards the receiving as such and the time of receiving [Pfitz\_89, page 150f]. For this the messages circle twice. During the first circle collisions may occur, but during the second circle for distributing, the results of the first one have to have no collisions. The difference between both options depends on where the cycle was started: whether it was a fixed and statically marked station or a sender who was marked dynamically. The statically marked stations may be known globally while the dynamically marked station should be remain anonymous.

### 5.4.4.1.1 An efficient 2-anonymous ring access system

A ring with a given access protocol is called ***n*-anonymous** if there is no situation where an attacker encircling  $n$  consecutive stations with *any desired amount of attacking stations* can identify one station as a sender or a receiver.

Some known techniques (slotted ring, token ring) are efficient and suitable because they can be modified in a manner that sending permission is granted without time limit and that every information unit (package, message) circles the ring completely. The former



Efficient ring access methods (token, slotted) hand on a temporal unlimited right-to-send from station to station. If an attacker encircled 2 stations, that send after the receipt of the right-to-send altogether  $n$  infotmation units ( $I_i$ ) and then hand on the right-to-send to the next station, there are  $n+1$  alternatives, which station could have sent which information unit. For each information unit there is at least one alternative, on that station  $i$  ( $i=1,2$ ) sends the information unit.

The example can be generalized on  $g$  encircled stations. There are  $\binom{n+g-1}{n}$  alternatives and at least one alternative for each information unit, on that station  $i$  ( $i=1, 2, \dots, g$ ) sends the information unit.

Figure 5.8: An anonymous access protocol for RING networks ensures: An attacker, who surrounds a sequence of stations, cannot decide, which station sends which message.

causes distributed (and to be shown: anonymous) query. The latter (which causes information units not to be removed by the receiver but by the sender) also executes distribution so that the receiver is protected too.

To prove the uncertainty of the attacker towards the real procedure, for every turn where an information unit is sent alternative turns are constructed. These alternative turns are shown in Figure 5.8. Therefore the ring with this ring access system is proven to be 2-anonymous.

The formal proof from [HöPf.85][Höck.85] corresponds completely to the proof shown

here at the information unit level.

This proof only holds for an attacker who does not know the sender of the information units that have to be send from different stations during the alternative turn. That knowledge could be gained by the attacker if he himself is the receiver of those information units and if the sender identifies himself in the information units. That is not modeled here due to §5.4.9. It is a limitation of possible alternatives on level 2 through higher levels.

#### 5.4.4.1.2 Fault tolerance in RING-Networks

With a RING-Networks the series quality is obvious: All connections and stations have to work to make communication between two stations possible in both “directions”.

Through the cooperation of circular transmission topology, digital signal regeneration, and methods for anonymous multiple access you can achieve anonymity and unlinkability (cp. §5.4.9). But this requires the employment of fault tolerance measures that will not hamper the cooperation.

The following concentrates on the levels 0, 1 and 2 of the ISO OSI Reference Model (fig. 5.40 in §5.4.9).

Transient errors of the access systems (level 2) can be tolerated with the usual techniques for restarting the access system used for rings with circling frames or circling sending permission.

Permanent errors of the access system (level 2) imply that at least one station in the RING-Network is faulty. This station has to be excluded until it is repaired with the techniques described here.

Transient errors of the access systems of the ring (therefore level 0 or 1) either affect only the message content of the transmitted information units or the access system too (level 2). The latter type of error is simple and can be detected and fixed by the end-to-end protocols. There are transient errors that are more difficult to handle. They affect the access system like deleting the sending permission token in a ring with circling sending permission. Those and similar errors in rings with circling transmission frames or circling sending permission can be tolerated with the usual techniques for restarting. Two additional ideas are important for the fault tolerance of the access system (level 2):

1. To tolerate transient errors of the second kind on the levels 0 or 1 (see above) you are introducing indeterminism (*indeterministic protocols*, cp. [Mann\_85, page 89ff]). With this, no attacker can draw sure conclusions with a deterministic attacker model (the attacker must conclude deterministically which station has sent and received). Since there is as of yet no sufficient formal statical attacker model (that is, in which the attacker is satisfied when he knows sender or receiver with a probability of 0,9 (cp. [Höck\_85, page 60ff]) the consequences of indeterminism can not be quantified.
2. Another kind of fault tolerance originated from the idea of guaranteeing anonymity to all but a few stations like protocol transformers in a hierarchical communication

network. Those stations can work with other protocols than other stations and tolerate errors for purpose [Mann\_85, page 99ff]. That is why they are called *asymmetric protocols*. Those are much easier to implement and they are much more powerful. But they do weaken the anonymity of the participant. If the simple access system is  $n$ -anonymous, the modified asymmetric protocols are often only  $(n + 1)$ - or  $(n + 2)$ -anonymous.

Permanent errors in the transmission system of the ring (therefore on level 0 or 1) always results in a *ring reconfiguration* with restart (synchronization, ...) of the technique for anonymous multiple access (level 2). Because the errors to be tolerated are situated on level 0 (media) and level 1 (physical) they can be easily spotted (time bounds, self-tests as well as external tests through many instances so that an attacker that controls a few stations can not easily turn on and off other stations when desired). More problematic is the error correction because you need to prove that it will not violate anonymity. It must be the goal of every error correction to reconfigure all error free stations and connections to a ring that contains all error free stations (and not an “eight”; therefore not two rings connected by one station).

The most simple and least expensive way of ring reconfiguration is the usage of a **bypass installation** per station. That bypass installation is closing the ring physically if the station notifies that it is faulty, turned off, or a blackout occurred. However you can not tolerate the failure of a connection as well as the failures of stations that can not diagnose this on their own. If only the station itself can close its bypass installation, its deployment has no further problematic interplays with anonymity and unobservability of the RING-Network. In particular, an attacker can observe the closing of bypass installations of bordering stations not under his control because the analogue characteristics of the incoming signals will change (cp. §5.4.4 about the signal regeneration). Consequently he knows that he now is surrounded by a smaller group of stations whereby he now gains more information about the sending of those participants. If an attacker can only surround many stations concurrently and if on these stations only a few bypass installations are closed, then it is not that bad.

The usage of *ring-conntector-concentrators* that are bypass installations that are neither located in the stations nor under their control [BCKK\_83][KeMM\_83], is incongruous with goals of RING-Networks from §5.4.4.1 since those concentrators would be ideal observation points providing complete observation of all stations connected.

Another simple but more expensive method of ring reconfiguration is the usage of **many parallel rings** (statically created redundancy). If a connection of the rings fails, another intact ring is used (dynamically activated redundancy). This allows, as long as there is more than one ring intact, a bigger throughput than sending the information unit on all rings simultaneously (statically activated redundancy). Though if no ring is completely intact any more the assignment of connections to stations has to be switched or you will have to send all rings simultaneously. Of course you can not tolerate an error of a station that does not take notice of that on its own.

This is also true for combinations of bypass installations and parallel rings.

To tolerate at least one permanent single error in the transmission system of the ring, a **braided ring** (fig. 5.9) is used. As long as there are no permanent faulty connections connecting neighboring stations, they are run as a RING-Network. For odd amounts of stations the other connections can be run as a second RING-Network which doubles the transmission capacity.

In a braided ring there are three possible *single errors*, namely the failure of

- one *station  $i$* : It is bridged with the connection  $C_{i-1 \rightarrow i+1}$  (cp. fig. 5.9 top right).
- one *inner connection  $C_{i-1 \rightarrow i+1}$* : The outer ring stays intact and is used exclusively until the failure is corrected.
- one *outer connection  $C_{i-1 \rightarrow i}$* : Station  $S_{i-2}$  sends to station  $S_{i-1}$  and  $S_i$  (it copies the data stream onto two connections). To make station  $S_{i+1}$  unable to observe  $S_i$ ,  $S_{i-1}$  sends half of all information units (for example all frames with even numbers as far as  $i - 1$  is even) to  $S_{i+1}$ . Correspondingly  $S_i$  sends the other half to  $S_{i+1}$ .  $S_{i+1}$  rejoins both data streams (fig. 5.9 bottom right). This is to be favored over a reconfiguration of the inner ring (fig. 5.9 bottom left) because so you are able to tolerate failures of inner connections and stations.

Built on these three basic concepts for tolerating a failure (even with the most multiple failures), the braided ring can be reconfigured with the following protocol executed by every station. This places every error-free station in a reconfigured, almost all error-free stations containing ring (therefore on all incoming and outgoing connections, information units are transmitted from almost all error-free stations).

#### **Reconfiguration protocol for braided rings:**

If station  $S_i$  takes notice of a permanent signal loss on one of its incoming connections, it signals this with a message to all other stations on both its outgoing connections. As long as the signal loss is not repaired,  $S_i$  will ignore this incoming connection. If the station sending on this connection is not sending on its other connection that the station is in working condition, we will assume it failed. In this case it is bridged with a corresponding inner connection. Otherwise, a connection failure is assumed and a reconfiguration is done.

With a failure of a station or a inner connection the result is a complete ring of all error-free stations. In regards to the proof of anonymity, you may take it over from the non-error case. In the case of a failure of an outer connection two RING-Networks (in terms of the anonymity) with half the bandwidth are created. In Figure 5.9 one of the RING-Networks goes through  $S_{i-1}$  the other through  $S_i$ . However in both RING-Networks only one station each can receive. But that is irrelevant because the receiver will not change anything on the information units he received if the normal ring access system is used. But if - as seen in [Pfit\_89, §3.1.4.3] - a transmission frame is used as duplex channel with one of the other stations  $S_{i-1}$  or  $S_i$ , the called station can send only (into one of the rings) with a probability of 0.5. The situation that a station located

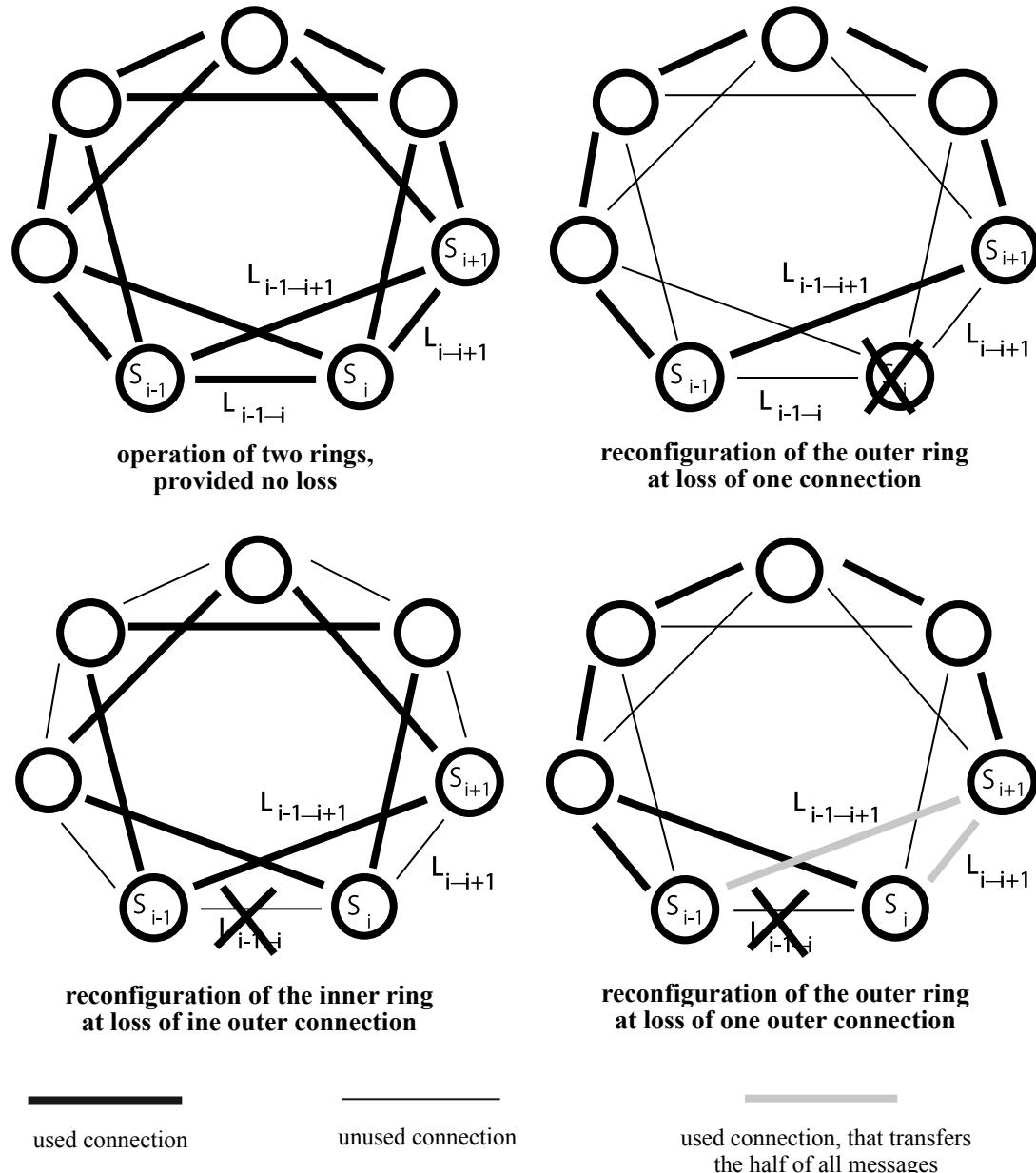


Figure 5.9: In case of a failure, a braided ring can be reconfigured in such a way that anonymity of the station is kept

in only one RING-Network with half bandwidth is asked to send by a transmission frame in which it can not send, may be quite a rare event. However this may thwart the anonymity towards the calling party. But this is only a special case of a general problem, that a receiver can be identified if he can not answer because of failure of his station, which is the only failing station, and the sender knows this (cp. §5.4.8 for an active attack on resource shortage).

Moreover you will have to take care (as with the technique for bypass installing) that a considered attacker in the non fault tolerant case becomes an unconsidered attacker (according the attack model). If the attacker controls the stations  $S_1$  and  $S_4$  (cp. fig. 5.9 with  $i = 2$ ) he can fake the failure of connection  $C_{3 \rightarrow 4}$  or just wait. In both cases  $S_2$  sends to  $S_3$  as well as to  $S_4$ . That makes  $S_2$  encircled by the attacker and therefore observable. It is mentioned that in some cases only one station alone can observe another. If  $S_1$  fails in a way that it sends the same on both its outgoing connections  $S_3$  can observe  $S_2$  on its own.

An exact description of fault tolerance in RING-Networks as well as many examples and protocols can be found in [Mann\_85]. There you may also find a more precise analysis of reliability improvements done with the shown fault tolerance techniques. The results are fairly positive.

More current formulas for calculating reliability can be found in [Papa\_86].

It is remarkable that superposed sending can be implemented in a braided ring in a manner that after any single error a working DC-Network with the full bandwidth is created [Pfit\_89, page 264].

A combination of bypass installations and and braided rings is possible and useful.

#### 5.4.4.2 Collision avoiding tree network (TREE-Network)

Almost all broadband cables in the end user area are ordered in a form of a tree. For pragmatic reasons an important example for the idea of “unobservability of bordering connections and stations as well as signal regeneration” is the collision-preventing TREE-Network [Pfit\_86, page 357].

The inner nodes are equipped with **collision-avoidance circuits** ([Alba\_83]; collision-avoidance switches [SuSY\_84]). A collision-avoidance circuit routes an information stream coming from the leaf nodes to the root only if the channel leading to the root is free. If more than one information stream from the leafs races for the free channel to the root, the collision-avoidance circuit picks one randomly and ignores the others. If the stations are directly connected to collision-avoidance circuits so that there are inner stations of the TREE-Network, these information stream are treated as if they were coming from the leafs.

The information stream from the root to the leafs is routed to all stations (and therefore through all collision-avoidance circuits too).

To observe the sending of a station on the leafs of the TREE-Network, a special connected has to be observed (in a RING-Network you have to observe two special connection

each); to observe the sending of a inner station of a TREE-Network all its incoming and outgoing ports have to be observed. Usually almost all stations are leafs of the TREE-Network. An attacker would partition the TREE-Network by observing a connection or by controlling the collision-avoidance circuits towards the sender anonymity, the resulting anonymity is weaker than in a RING-Network. The same applies to those who observe their neighbors without the order from a third party (therefore without collaboration with the big communication brothers). He will gain almost no information because all messages are encrypted and addresses are, if implicit addressing is used, not interpretable.

Evaluations [Alba\_83][SuSY\_84] show an excellent behavior of TREE-Networks with collision-avoidance circuits under random or uncoordinated access.

If the TREE-Networks shall be enhanced with *superposed sending* (cp. §5.4.5), the collision-avoidance circuits must be replaced by modulo adder. As seen in [Pfit\_89, page 105] every connection not observed by the attacker is like an exchanged key towards the superposed sending which also holds for TREE-Networks. The same applies to every networks with digital signal regeneration and with superposition topology corresponding its transmission topology.

By using modulo adders instead of collision-avoidance circuits you can achieve twice the throughput for certain kinds of traffic with superposed sending (the remaining traffic is transmitted as fast as before; cp. §5.4.5.4). This is an argument for considering the collision-avoidance circuits as “outtaken” by superposed sending.

## 5.4.5 Superposed sending (DC-network): Protection of the sender

After a short outline of the generalized superposed sending in §5.4.5.1, the anonymity of the sender is defined and proved in §5.4.5.2. Afterwards we show a method to ensure the anonymity of the receiver by snap-snatch-distribution in §5.4.5.3.

In §5.4.5.4 a method is introduced, which allows much more efficient use of the superposed sending. This method is called superposed receiving.

Finally, some considerations about the optimality of superposed sending in terms of the anonymity of the sender, as well as its complexity and possible implementations, are presented in §5.4.5.5. Concluding, fault tolerance methods for the DC-network are described in §5.4.5.6.

### 5.4.5.1 A first outline

David Chaum specifies a method for anonymous sending in [Cha3\_85, Cha8\_85, Chau\_88], and calls it DC-network (DC-network as the abbreviation for Dining Cryptographers network, according to his introducing example; also, DC represents David Chaums initials). This method for anonymous sending (and unobservable receiving as in §5.4.1) was generalized in [Pfi1\_85, Pages 40,41] to the method described in the following.

It is well known that every finite alphabet is an Abelian group with respect to a serial numbering of its elements starting with 0, and with respect to addition (modulo the cardinality of the alphabet) concerning this numbering. As usual, let the subtraction (of a character) be the addition of its inverse element of the group. The **generalized superposed sending** then happens as follows:

Member stations create – randomly and according to an equipartition – one per several key-characters for every use-character which they want to send. Then they communicate every key-character to exactly one other member station via a channel (still to be discussed) which guarantees secrecy. (The data – who is communicating with whom via such a channel – does not need to be kept secret, and should even be publicly known, in order to simplify countermeasures against attacks against the supplying of this service, compare to §5.4.7.) Every member station adds (modulo alphabet size) locally all self-created key-characters, subtracts (modulo alphabet size) locally all key-characters, which were communicated to it, from that, and then adds (modulo alphabet size) locally the use-character that it wants to send – if it wants to send a use-character. This addition (or subtraction respectively) modulo the cardinality of the used alphabet is called *superposition*. Every member station sends the result of its local superposition (hence the name superposed sending). All the characters that were sent are now being globally superposed (i.e., added modulo alphabet size) and the resulting sum-character is distributed to every member station.

As every key-character was exactly once added and exactly once subtracted, and the key-characters hence erase each other after the global superposition, the sum-character is the sum (modulo alphabet size) of all the sent use-characters. If no member station wanted to send a use-character, the sum-character is the character that corresponds to 0. If exactly one member station wanted to send a use-character, the sum-character is equal to the sent use-character.

If one chooses the binary characters 0 and 1 as an alphabet, then one obtains the – for practical purposes important – special case of the **binary superposed sending**, which was specified by David Chaum. In this case, one does not need to distinguish between addition and subtraction of characters (Figure 5.10).

Of course, (digital) *superposition collisions* may occur, if several member stations want to send simultaneously: all stations receive the (well-defined) sum of the simultaneously sent characters. Collisions are a common problem in distribution channels with multi-access, and there are a large number of access methods to solve this problem. All published access methods are designed for (analog) *transmission-collisions*, for instance in bus-systems (e.g., ethernet), radio- and satellite-nets. In these systems, there exists no well-defined “collision result”. So the “working conditions” of the access methods for superposed sending are considerably better than the “working conditions” for normal distribution channels. Indeed, one may of course only use such access methods which preserve the anonymity of the sender and which – if possible – also preserve the impossibility to concatenate sending-events. Alongside, the access methods should – for expected traffic allocation – take advantage of the channel which is available. Examples for anonymous and not-concatenating access methods are simple, but not very efficient method slotted ALOHA, and an efficient reservation method which was designed for

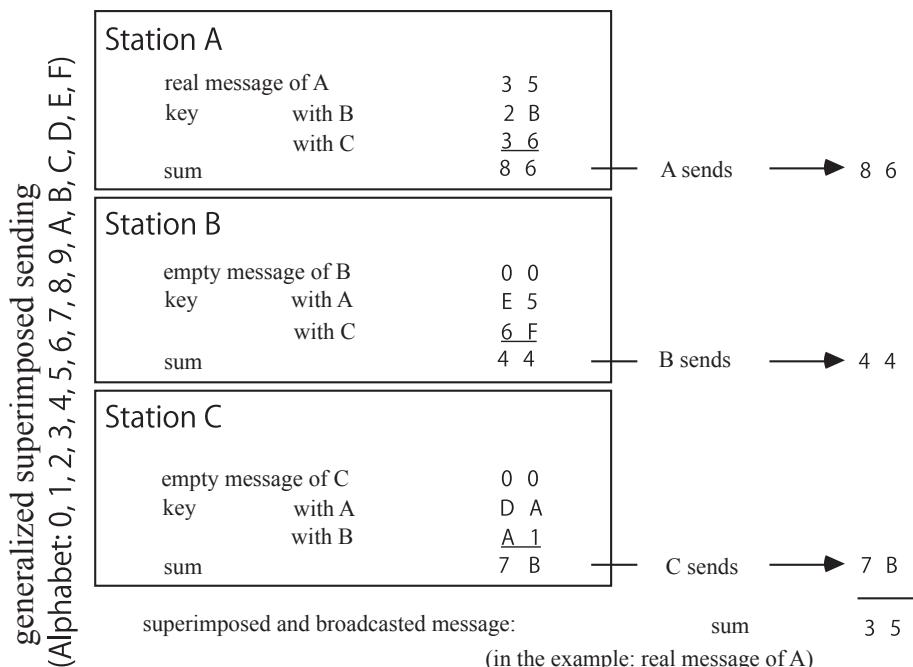
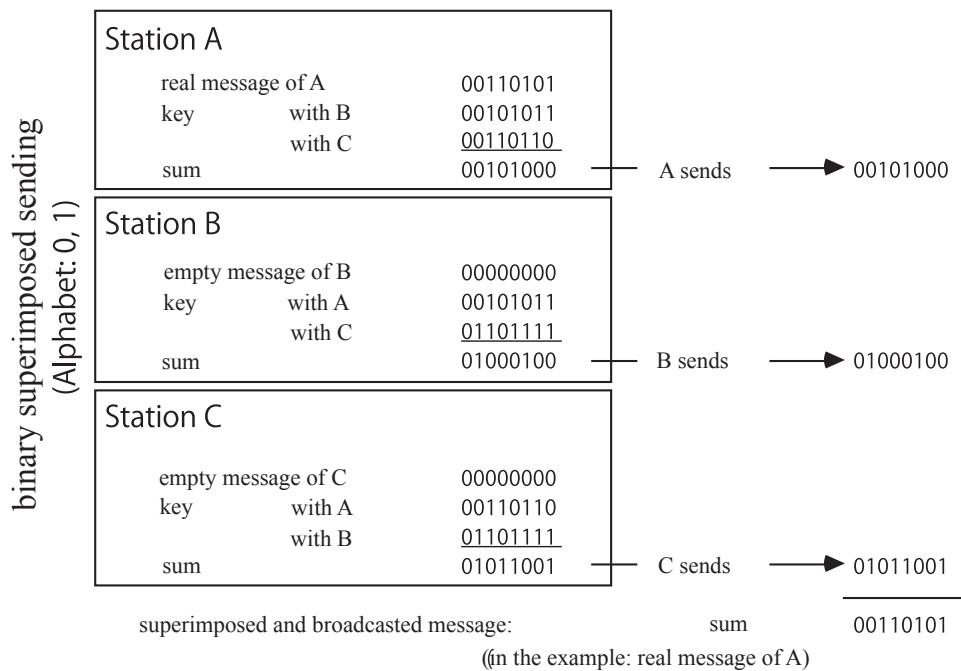


Figure 5.10: superposed sending

channels with large time delays [Tane\_81, Page 272], [Cha3\_85].

If one declares that a participant may continue to use a transmission-slot in which he has already sent without collisions and if one further declares that other participants may not send in this slot until it was once not used, then one can switch very efficient channels by using several transmission slots [Höck\_85],[HöPf\_85]. Of course, everything that is sent over such a channel can be concatenated – on the other hand, such channels are typically used for services which by nature contain concatenation. This is discussed in detail in §5.4.9.

#### 5.4.5.2 Definition and Proof of sender anonymity

After this short overview of superposed sending, *sender anonymity* will first be defined exactly and then proven for the most common case of superposition (arbitrary Abelian group).

Afterwards, a way – namely *superposed receiving* – is shown, how to make use of the fact that superposition collisions have a well-defined value that is visible to every member station that participates in the superposed sending. Both are in contrast to transmission collisions in bus systems (e.g., ethernet) and wireless networks and only the former in contrast to satellite networks.

Several annotations to the implementation of the key distribution and to the implementation of the superposed sending close this section.

David Chaum proves in [Cha3\_85] for binary superposed sending (using the properties of GF(2)), that other members can not learn (not even by gathering all their information) who is sending in a group – as long as this group of member stations passes keys around only in the described way, and as long as this group is connected with respect to the distributed keys. Being “connected with respect to the distributed keys” means that for two arbitrary group member stations  $M_0, M_1$  there exists a (possibly empty) sequence of member stations  $M_2$  to  $M_n$  of the group, such that for  $1 \leq i \leq n$  the following holds:  $M_i$  has exchanged a key with  $M_{i+1} \bmod (n + 1)$ .

In the following, a proof of this is given, which is simplified and generalized to generalized superposed sending:

The proof only uses the fact that the characters and the addition defined by them constitute an *abelian group*. It does not use the fact that the addition after the enumeration of the characters is modulo alphabet-sized (i.e., that it is a *cyclic group*). Thus we have an implicitly given ring-structure in generalized superposed sending, which is not used in the proof, thus the proof is as general as one could wish:

1. In order that a station can send no sign, there must exist a *unit element*.
2. In order that the keys neutralize pairwise after the superposition, there must exist an *inverse element* for every character.
3. In order that one can freely organize the local and especially the global superposition, the addition should be commutative.

In order that a key-distribution between arbitrary pairs of member stations is possible, the addition must be *commutative*. Otherwise, there would be a - by the global order of superposition defined - linear order on the characters which are sent by the member stations, and thereby also on the member stations. Because key-characters do not commute with use-characters, every member station could only exchange keys with the one or two neighbor-stations (neighbor with respect to the linear order).

Also, the proof is a bit more general than the definition of the generalized superposed sending - indeed, there is no usable application for this larger generality. This is the case because the main-proposition of finite abelian groups (compare [Waer\_67, page 9]) states that every finite abelian group can be written as a direct product of cyclic groups. But the direct product, i.e., the component-wise execution of the addition in the particular cyclic group, corresponds in detail to the collateral, i.e., serial or parallel, operation of several generalized superposed sending-methods.

**claim:** If member stations that are connected with respect to the secretly distributed keys superpose their messages and the respective keys and only output the result, then an attacker who observes all the results, *only* gets to know the sum of all messages. The latter means, that for the attacker, for all probability-variables  $Z$ , the following holds:  $p(Z | \text{allresults}) = p(Z | \text{sumofallsentmessages})$ .

**annotation:** Depending on the previous knowledge of the attacker, he thereby also possibly knows the single messages, which build up the sum as well as - if single messages together with the previous knowledge of the attacker allow for drawing conclusions about the sender - the sender of these messages. But the attacker does - by knowing all results, i.e., the outputs of the single member stations - not receive more *additional* information which exceeds the knowledge of the sum. He thus does not need to observe the outputs of the single member stations, as that does not give him any additional information about the sender of a particular message. According to the definition in §5.1.2, the senders of messages thus remain anonymous with respect to the attacker and with respect to the event of sending.

**proof:** Without loss of generality, in the following the proof will be restricted to such situations, where  $m$  member stations are *minimally*<sup>5</sup> connected with respect to the distributed keys. Let this mean, that for each two arbitrary member-stations  $T_0, T_1$ , there exists *exactly one* possibly empty sequence of member stations  $T_2 \dots T_n$ , such that for  $1 \leq i \leq n$ , the following holds:  $T_i$  has exchanged a key with  $T_{(i+1) \bmod (n+1)}$ . If there exists more than one sequence, then the attacker gets to know more keys, which only makes him stronger. He then can subtract these keys from the outputs of the affected member stations, whereby he respectively receives exactly the “output”, which is in the following generated by completely omitting these keys.

---

<sup>5</sup>This means, that the member stations are connected with respect to the distributed keys, but no key can be removed without harming connectedness, i.e., after removing one key, at least two member stations are not connected anymore. For the graph which consists of the member stations as nodes and of the keys as edges, this means: The graph is acyclic.

The proof itself is by **complete induction** on the number  $m$  of member stations. Let a *message-combination* be the assignment of each particular message to every member station, let a *key-combination* be the assignment of values to all keys secretly exchanged between member stations and to the keys that are unknown by the attacker. We prove for every  $m$ , that for every message-combination which gives the sum of all sent messages, there exists exactly one key-combination, such that message-combination and key-combination are compatible with the outputs of all member stations. Because the keys and thereby also the key-combinations are generated at random and according to an equipartition, the observation of the outputs does not deliver any information which would exceed the sum of all messages (according to Shannon's definitions [Shan\_48, Sha1\_49]) to the attacker.

Let  $S_{i \rightarrow j}$  be a key that was generated by  $T_i$  and communicated secretly to  $T_j$ , let  $N_i$  be a message which was sent by  $T_i$ , and let  $A_i$  be its output. According to the description of the generalized superposed sending, the following holds:

$$A_i = N_i + \sum_j S_{i \rightarrow j} - \sum_j S_{j \rightarrow i}$$

#### **Induction base: m=1**

The claim holds trivially, as the attacker gets to know exactly the output of the only member station.

#### **Induction step from m-1 to m:**

As the member stations are *minimally* connected with respect to the distributed keys, there is always a member station which is only connected to one other member station by an exchanged key. Let the former be without loss of generality called  $T_m$  and let the latter be called  $T_i$  with  $1 \leq i \leq m-1$  and let the key be without loss of generality be called  $S_{i \rightarrow j}$  (compare fig. 5.11).

$T_i$  creates  $N_i$  and  $A_i = N_i + \dots + S_{i \rightarrow m}$ ,  $T_m$  creates  $N_m$  and  $A_m = N_m - S_{i \rightarrow m}$ .

The attacker observes the sending of  $A_1, A_2, \dots, A_m$ .

Let  $N' = (N'_1, N'_2, \dots, N'_m)$  be an arbitrary message-combination, whose sum is equal to the sum of all sent messages, i.e.,  $N'_1 + N'_2 + \dots + N'_m = N_1 + N_2 + \dots + N_m (= A_1 + A_2 + \dots + A_m)$ .

Now it remains to show, that for the message-combination  $N'$ , there exists exactly one fitting key-combination  $S'$ . The between  $T_i$  and  $T_m$  exchanged fitting key  $S_{i \rightarrow m}$  has because of  $A_m = N_m - S_{i \rightarrow m}$  the value  $S'_{i \rightarrow m} = N'_m - A_m$ .

The rest of the fitting key-combination  $S'$  is determined by the induction hypothesis, whereas the value  $A_i - S'_{i \rightarrow m}$  is used as output of  $T_i$ . The induction hypothesis is applicable, as the remaining  $m-1$  member stations are again minimally connected with respect to the remaining keys.

Thereby it is proven, that the generalized (and thus of course also the binary) superposed sending creates *perfect information-theoretic anonymity* of the sender inside of the group. The group is connected by randomly selected and in a perfect information-theoretic secrecy-guaranteeing manner distributed keys.

Also, messages can not be concatenated in any way by the sending, as long as we ignore the influences of the multi-access method, compare §5.4.5.4.1 and §5.4.9. Superposed

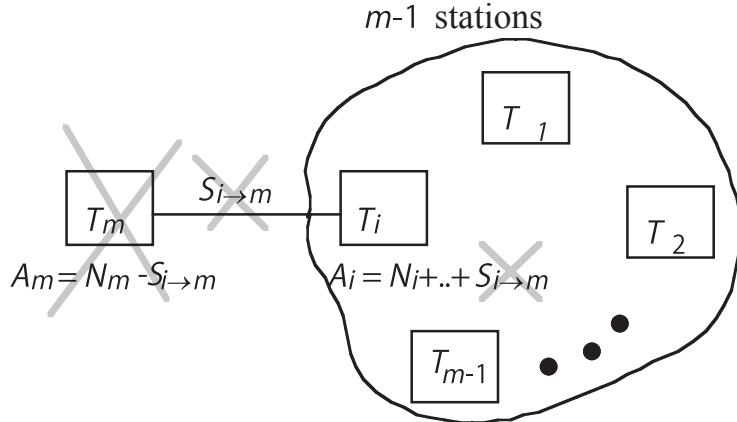


Figure 5.11: Illustration of the induction step

sending thus also makes *perfect information-theoretic impossibility to concatenate* of sending-events possible. If a message suffers a superposition-collision, then it must indeed be again and thereby in another way (end-to-end) encoded, before it can be sent again. Otherwise, an attacker could deduce for instance - if the messages are sufficiently long - that the messages, whose sum was already sent previously, were probably not sent by the same member station, as this member station otherwise would waste transmission-bandwidth of the DC-network by creating superposed-collisions.

David Chaum's proof for the binary superposed sending and the here given proof for the generalized superposed sending holds not only for *observing*, but also for coordinated *modifying* attacks - in the here given proof for instance, the behavior of the attacker-stations is not even mentioned, it can thus be arbitrary. (behavior of the attacker-stations which is not allowed by the protocol is only an attack on the service-performance, i.e., availability, which is treated in §5.4.7).

Nevertheless, both proofs only consider the anonymity of the *sender*. If the sending of a message  $N$  depends on the receiving of previous messages  $N_i$ , the modifying attack on the anonymity of the *receiver* of one of the messages  $N_i$  (compare §5.4.1) can - even for perfect protection of the sending - identify the sender of  $N$ . As already mentioned in §5.4.1, this problem can - by appropriately including the distributed messages into the key-generation - be perfectly solved, even inside the information-theoretic model-world [Waid\_90],[WaPf1\_89].

### 5.4.5.3 Receiver's anonymity achieved by the snap-snatch-distribution

The aim of the snap-snatch-distribution is to stop the message-transmission, as soon as two not-attacking participants receive different characters as a global result of superposition in a round of the DC-network.<sup>6</sup>

<sup>6</sup>According to the organization of §5.4 with respect to the protection-aims "protection of the receiver", "protection of the sender", and "protection of the communication-relation" this does not belong to

If a non-attacking participant  $T_i$  notices such an inconsistency, then it is easy to stop message transmission:  $T_i$  just interferes with the superposed sending in the next rounds by choosing its output-character randomly and according to an equipartition. Thus the global result of superposition of these rounds is independent from the use-characters.

In §5.4.5.3.1 an obvious but inefficient implementation of this idea is discussed using a comparing-protocol.

Snap-snatch-key-generation-methods are described in §5.4.5.3.2 and §5.4.5.3.3. They generate keys for the superposed sending depending on the previously received global result of superpositions and assure that two participants which have received inconsistent results of superpositions will afterwards use completely independent keys and thus they stop the message transmission.

#### 5.4.5.3.1 Comparing the global result of superpositions

In order to discover inconsistencies in the distribution, the participants can explicitly compare the received global result of superpositions by using an additional protocol: After each round of the superposed sending every member station  $T_i$  sends its global result of superposition  $I_i$  (input) to all other member-stations  $T_j$  with  $j > i$ . If a non-attacking member station  $T_j$  receives a global result of superposition which is not equal to  $I_j$  or if it receives nothing from another member station  $T_i$  with  $i < j$ , it interferes with the superposed sending in all the following rounds.

Such testing phases are well known from byzantine accordance protocols.

In order to make these tests reliable, the communication between  $T_i$  and  $T_j$  must be protected with a perfect authentication-code [GiMS\_74],[Sim3\_88], i.e., with a code, which allows the successful faking of a message at most with the probability  $\frac{1}{\sqrt{|F|}}$ , where  $F$  is the key-space, compare §3.3. An additional message and an additional key are hence necessary for every test.

The number of tests can be chosen depending on the assumed strength of the attacker: Let  $G^*$  be an undirected graph with the member stations as nodes. Two member stations  $T_i$  and  $T_j$  are directly connected in  $G^*$  if  $T_i$  and  $T_j$  compare their global result of superpositions. In correspondence to superposed sending, the following Lemma 1 holds:

**Lemma 1** Let  $A$  be the subset of the member stations, which is controlled by the attacker and let  $G^* \setminus (T \times A)$  be well-connected.

If two non-attacking member stations  $T_i$  and  $T_j$  receive two different global superposition results  $I_i$  and  $I_j$ , then there is a pair of non-attacking member stations  $T_{i'}$  and  $T_{j'}$  which are directly connected in  $G^*$  and which also receive both different global result of superpositions.

**Proof:** Because of the good connection of  $G^* \setminus (T \times A)$ , there is a path  $(T_i = T_{k1}, \dots, T_{km} = T_j)$  with  $I_{k_z} \notin A$  and  $(T_{k_z}, T_{k_{z+1}} \in G^* \setminus (T \times A))$ . As  $I_i \neq I_j$  by assumption, there exists an index  $z$  with  $I_{k_z} \neq I_{k_{z+1}}$ . Choose  $(i', j') = (k_z, k_{z+1})$ .

Obviously, the good connection of  $G^* \setminus (T \times A)$  is a necessary requirement.

---

“protection of the sender” but to “protection of the receiver”, i.e., in §5.4.1.1. But this method would not be understandable there, as it is based on the sender anonymity-method “superposed sending”. Thus it is described here.

The protocol needs in every round  $|G^*|$  additional messages, which is usually in the size  $\mathcal{O}(n^2)$ . If  $G = G^*$  and if we assume, that one key from  $F$  is necessary for the authentication of each testing-message, the number of keys, which must be exchanged in a secrecy-guaranteeing manner, is doubled in comparison with normal superposed sending.

If we deal with a physical distribution network<sup>7</sup>, the number of pest-messages can be reduced to  $\mathcal{O}(n)$  by using a digital signature-system instead of an authentication-code [DiHe\_76],[GoMR\_88]. The resulting scheme is of course at most complexity-theoretical secure.

#### 5.4.5.3.2 Deterministic snap-snatch-key generation

A more efficient implementation of snap-snatch-distribution is achieved by combining the two tasks: to detect inconsistencies and to stop the superposed sending. If the current keys  $K_{ij}$  and  $K_{ji}$ , which are used for the superposed sending, depend completely on the previous global result of superpositions, which were received by  $T_i$  and  $T_j$ , then an inconsistency between  $I_i$  and  $I_j$  automatically stops the superposed sending because the global result of superposition is now random and not the sum of all use-characters anymore.

Let  $\delta_{ij}^t := K_{ij}^t - K_{ji}^t$  and  $\varepsilon_{ij}^t := I_i^t - I_j^t$  for all  $i, j, t$ . (Whereas  $t$  is not an exponent but an “above” written index.) A snap-snatch-key generation-scheme for superposed sending should for all directly in  $G$  connected member stations  $T_i$  and  $T_j$  guarantee the following:

**SupSen** *Superposed sending*: If for all rounds  $s = 1, \dots, t-1$  the equation  $I_i^s = I_j^s$  holds, then the keys  $K_{ij}^t$  and  $K_{ji}^t$  for round  $t$  are *equally* and randomly chosen from  $F$ . More formally (let  $x \in_R M$  mean, that the element  $x$  is randomly - according to an equipartition - and independently of all others selected from the set  $M$ ):

$$[\forall s \in \{1, \dots, t-1\} : \varepsilon_{ij}^s = 0] \Rightarrow K_{ij}^t \in_R F \text{ and } \delta_{ij}^t = 0.$$

Then the superposed sending works as usual.

**SnaSna** *Snap-snatch*: If there is an index  $s < t$  with  $I_i^s \neq I_j^s$ , then the keys  $K_{ij}^t$  and  $K_{ji}^t$  for round  $t$  are chosen *independently* and randomly from  $F$ . More formally:

$$[\exists s \in \{1, \dots, t-1\} : \varepsilon_{ij}^s \neq 0] \Rightarrow K_{ij}^t \in_R F \text{ and } \delta_{ij}^t \in_R F.$$

The superposed sending is stopped by every such pair of keys, i.e., the result of the global superposition is independent of the sent use-characters. Because of the well-connectedness of  $G \setminus (T \times A)$ , this realizes the snap-snatch-property from Lemma 1 (with  $G = G^*$ ).

In the rest of this section, an arbitrary but fixed pair of keys  $(K_{ij}, K_{ji})$  with  $T_i \notin A$  and  $T_j \notin A$  is considered. Thus the indexes  $i, j$  are often omitted.

The strongest attacker is assumed: he can directly observe the values of  $K_{ij}^t$  and  $K_{ji}^t$  for every round  $t$  and he can supply  $T_i$  and  $T_j$  with arbitrary global result of superpositions

---

<sup>7</sup>in which the consistency of the distribution is not guaranteed, otherwise we would not need to bother.

## 5 Security in Communication Networks

$I_i^t$  and  $I_j^t$ . The member stations  $T_i$  and  $T_j$  are assumed not to be synchronous, so that the attacker can wait for  $K_{ij}^{t+1}$  until he supplies  $I_j^t$  to  $T_j$ .

Let  $(F, +, \cdot)$  be a finite field and let  $a^1, a^2, \dots, a^{t_{max}}$  and  $b^1, b^2, \dots, b^{t_{max}-2}$  be two sequences whose elements are randomly selected from  $F$  and whose elements were exchanged  $T_i$  and  $T_j$  in a secrecy and authenticity guaranteeing manner. Let for  $t = 1, \dots, t_{max}$  the following be defined:

- (1)  $K_{ij}^t := a^t + \sum_{k=1}^{t-1} b^{t-k} \cdot I_i^k$ .
- (2)  $K_{ji}^t := a^t + \sum_{k=1}^{t-1} b^{t-k} \cdot I_j^k$ .

**Lemma 2** The by equation (1) defined key generation-scheme fulfills the two above requirements SupSpn and SnaSna.

**Proof:** As  $a^t \in_R F$  and as  $\Sigma := K_{ij}^t - a^t$  does not depend on  $a^t$ ,  $K_{ij}^t \in_R F$ . let  $\varepsilon_{ij}^s = 0$  for all  $s < t$ . Then obviously  $\delta_{ij}^t = 0$  and the requirement SupSen is fulfilled. Now let  $s$  be the first round with  $\varepsilon_{ij}^s \neq 0$ . For the sake of simplicity, let  $\varepsilon^u := \varepsilon_{ij}^u$  and  $\delta^u := \delta_{ij}^u$ . The differences  $\delta^u$  are computed according to the following system of linear equations:

$$\delta^u = 0 \text{ for } u = 0, \dots, s$$

$$\begin{pmatrix} \delta^{s+1} \\ \delta^{s+2} \\ \vdots \\ \delta^{t-1} \\ \delta^t \end{pmatrix} = \begin{pmatrix} \varepsilon^s & 0 & \dots & 0 & 0 \\ \varepsilon^{s+1} & \varepsilon^s & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \varepsilon^{t-2} & \varepsilon^{t-3} & \dots & \varepsilon^s & 0 \\ \varepsilon^{t-1} & \varepsilon^{t-2} & \dots & \varepsilon^{s+1} & \varepsilon^s \end{pmatrix} \cdot \begin{pmatrix} b^1 \\ b^0 \\ \dots \\ b^{t-s-1} \\ b^{t-s} \end{pmatrix}$$

As  $\varepsilon^s \neq 0$ , the matrix is regular and defines a bijective mapping. As  $b^1, \dots, b^{t-s} \in_R F$ , the  $\delta^{s+1}, \dots, \delta^t$  are equally and independently distributed in  $F$ . The independency of all  $K_{ij}^1, \dots, K_{ij}^t$  and  $\delta^{s+1}, \dots, \delta^t$  follows from the independency ov all  $a^1, \dots, a^t$  and  $\delta^{s+1}, \dots, \delta^t$ .

The additional costs of this key generation scheme is the following:

- The altogether  $2 \cdot t_{max} - 0$  key-characters  $u^t, b^t$  (instead if only  $t_{max}$  for normal superposed sending) which need to be exchanged in a secrecy and authenticity guaranteeing manner between every pair  $T_i$  and  $T_j$  of member stations which are directly connected in  $G$ .
- Storage of all  $t_{max} - 1$  received global superposition-characters.
- The each  $t - 1$  additions and multiplications (in the field) in order to compute the key for roqnf  $t$ .

From the mentioned above it follows that the scheme needs each  $t_{max}/2$  additions and multiplications in average. Thus the scheme is little use in practice.

If there is no additional communication between  $T_i$  and  $T_j$  about their current state, then the scheme is *optimal* in terms of the number of exchanged key-characters and the additionally needed memory. This is shown in [WaPf\_89, page 13f].

The scheme for the deterministic snap-snatch-key generation becomes usable, if one makes the sums in equation (1) each not start at 1, but if one makes them for a fixed  $s$  start at  $t - s$ . At this,  $s$  needs to be chosen in such way, that every member station can send its own use-character - which can not be guessed by the attacker with sufficiently high probability - after at most  $s - 1$  other sent use-characters. If the member station receives its own use-character as global result of superposition (which nobody would have noticed, if there was interference by inconsistent distribution in one of the last  $s$  rounds), then there was - with corresponding probability - no inconsistent distribution in the previous  $s$  rounds. As the attacker does not get to know anything about the  $b^u$  if the distribution is consistent, one can reuse each of them after  $s$  rounds. Thereby not only the number of field-operations is reduced, but also the number of necessary key-characters is reduced. Thus one can gain a tolerably practical scheme by adequately choosing  $s$ .

{It is described in [LuPW\_91, §2.2] how the complexity of the deterministic snap-snatch-key generation approximately can be halved.}

#### 5.4.5.3.3 Probabilistic snap-snatch-key generation

A considerably more efficient implementation of the snap-snatch-key generation is obtained if one weakens the requirement SnaSna: Inconsistent distribution need not interfere with probability 1 forever, but it can only interfere the next  $r$  rounds (e.g.,  $r = 10^{100}$ ) with the probability  $p$  (e.g.,  $1 - 10^{-10}$ ). This should suffice for practical purposes.

The details of this method can be found in [WaPf\_89]. For each round, two additions and two multiplications inside the field - which indeed needs to be chosen large - are necessary. Find information on how to replace one of the two expensive multiplications by a cheap addition in [LuPW\_91, §2.1]. The complexity of the probabilistic snap-snatch-key generation can be approximately halved by this.

If every member station sends after at most  $s - 1$  characters again an own use-character (see previous section),  $s$  key-characters  $b^u$  can be used cyclically.

#### 5.4.5.4 Superposed receiving

The bandwidth of a DC-network can be used considerably better if the member stations use the fact that whoever knows the global result of superposition  $\ddot{u}$  of  $n$  simultaneously superposed sent messages as well as  $n - 1$  of the single messages, can compute the missing message by superposed  $\ddot{u}$  with the  $n - 1$  single messages. This **superposed receiving** can take place **globally**, i.e., all member stations superpose directly and thus receive the same characters, or it can take place **pairwise**, i.e., exactly two member stations can superpose:

In the *global superposed receiving*, all member stations store the (at first) unusable message after a superposition-collision, in order that for  $n$  collided messages only  $n - 1$  need to be sent again: the last message is gained by subtraction (modulo alphabet-size) of the  $n - 1$  messages from the (at first) unusable message.

If the member stations receive globally superposed, then a member station - which occasionally superposes several messages simultaneously and then sends every single

message except for one again - does not waste any bandwidth of the DC-network. Thus, it is possible without problems and can be generally known. Thereby, the above conclusion, that colliding messages do not come from the same member station, becomes invalid, so that the superposed receiving leaves sending-events - despite the enhancement of the efficiency - information-theoretically not linkable, if the probabilities for the simultaneous superposition of several messages are chosen appropriately. Additionally to the better utilization of the bandwidth of the DC-network, the globally superposed receiving also saves the retried and thus changed (end-to-end) encoding of collided messages.

*Pairwise superposed receiving* is possible, if two possibly completely anonymous member stations (for instance by using an anonymous reservation-scheme, compare §5.4.5.4.3) agreed with all others in the superposed sending participating member stations on the question that, and when both member stations have the exclusive sending privilege: Although *both* member stations are sending simultaneously, both can receive the data which was sent by the other respective member station, namely by superposing the data which they sent with the global result of superposition (Figure 5.12). This fact can be used for two purposes:

If both member stations each send a message for another member station, then the bandwidth of the DC-network is utilized twice as good as with the access-methods which are described in [Cha3\_85],[Chau\_88],[Pfi1\_85]. For example, a point-to-point-duplex-channel can be hosted within the bandwidth of a simplex-distribution-channel.

If one of the two member stations sends a completely randomly generated key, then it can receive a message from the other station in a perfect information-theoretic secrecy guaranteeing manner, even though both member stations can be mutually completely anonymous, if the underlying DC-network serves perfect information-theoretical anonymity (against the attacker) inside of some group which contains both member stations. The latter is an especially efficient implementation of the secrecy-protocol (which is based on sender anonymity) of Alpern and Schneider [AlSc\_83]. This protocol was already published before communication networks with sender anonymity were invented. With the - by Axel Burandt developed - code [Bura\_88], one can achieve, that both partners can check with information-theoretical security, whether third-party stations interfere with the message-transfer, thus *perfect information-theoretic integrity* is achievable.

#### 5.4.5.4.1 Criteria for the preservation of anonymity and unlinkability

In order to preserve the anonymity of the sender and in order not to link communication-events multi-access methods should

- *treat (or let operate) all stations in exactly the same way*(for instance, stations should not have a unique number as in the bit-map-protocol in [Tane\_88, page 296] or as in the group testing protocol in [BMTW\_84, page 721]),
- *not create a connection between communication-events* (for instance, “if this was sent by station A, then that was sent by station B.” or “if this was sent by station A, then that was not sent by station B.”, whereas in practice in both cases, we often have  $A = B$ ),

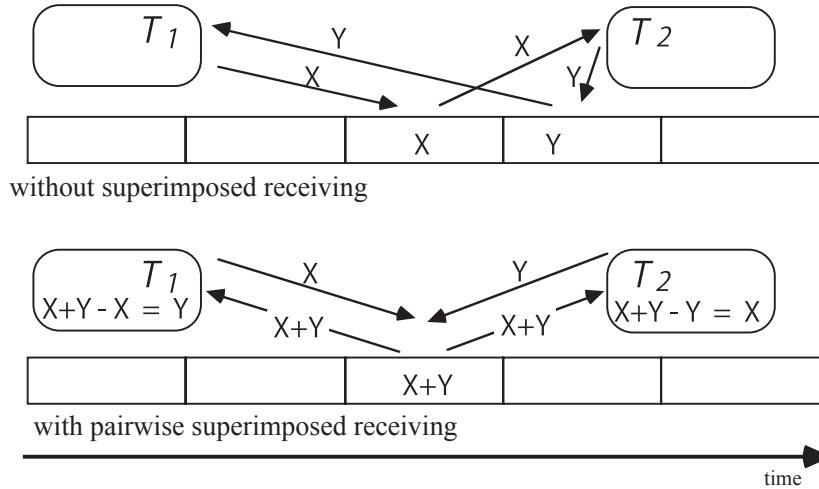


Figure 5.12: Pairwise superposed receiving of the user stations  $T_1$  and  $T_2$

- allow every station to use the same bandwidth, if no other station is sending.

1. is necessary and sufficient for *perfect preservation of the anonymity* of the sender. With the in §5.4.5.2 presented proof for superposed sending, 1. is also sufficient for perfect information-theoretic anonymity of the sender, as long as the sender does not identify himself explicitly by the content of the messages or by linkable messages.
2. is necessary and sufficient for *perfect preservation of unlinkability*. With the in §5.4.5.2 presented proof for superposed sending, 2. is also sufficient for perfect information-theoretic unlinkability of sending-events, as long as the sending-events are not explicitly linked by the content of the messages or by addresses.
3. is necessary for *perfect preservation of unlinkability*. If not every station was allowed to use the complete bandwidth, all simultaneously happening sending-events would be linkable in the respect that they do not all come from the same station (if only some stations may not use the complete bandwidth: ... they do not all come from this station).

#### 5.4.5.4.2 Algorithm for resolution of collisions with averaging and superposed receiving

The modulo- $g$ -addition of the superposed sending is used as (real) addition up to the value  $g - 1$ , in order to compute with this addition the rounded average  $[\emptyset]$  of all information units which are involved into a collision. Every station decides depending on the size of its information unit compared to  $[\emptyset]$ , when it will send again (**averaging**). In order that the modulo- $g$ -addition is a (real) addition, the sum of all involved information units must be smaller than  $g$ . This is for  $s$  stations, of which every station is allowed to send maximally  $m$  information units simultaneously, and for maximal size  $G$  (in the sense of the interpretation of the binary coding of the information units as dual-number) of information units with certainty always then the case, if  $g > s \cdot m \cdot G$ . As long as not

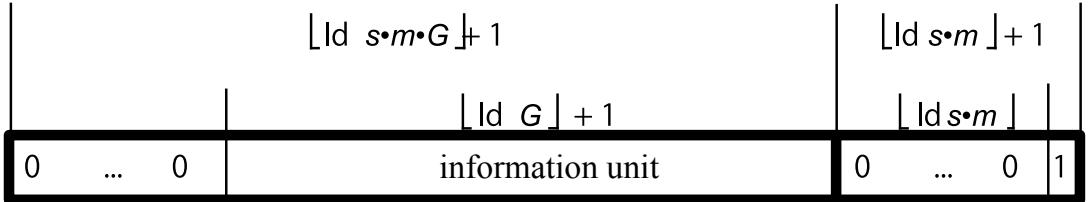


Figure 5.13: Message format of the algorithm for resolution of collisions with averaging and superposed receiving

all colliding information units are equal, there is always at least one unit lesser and one unit greater than  $\lfloor \emptyset \rfloor$ . Then it is never the case that all stations get the same result from averaging. Thus accordingly, it is never the case that all stations directly send again or that no station sends directly again. Hence no bandwidth is being wasted.

Figure 5.13 shows a fitting message-format for a sending station. For a non-sending station, all the bits of the message - even the last bit - are 0. The leading zeros in Figure 5.13 are necessary in order to prevent an overflow in the addition. Figure 5.14 shows a more detailed example with the message-format which is specified in Figure 5.13. The fact that additionally the pairwise exchanged keys are superposed when the multi-access method is used in the DC-network is illustrated neither in Figure 5.13 nor in Figure 5.11 nor in the following text. Because thus the leading zeros are generally not transmitted as zeros, the characters which correspond to them must actually be transmitted.

The number of leading zeros in the message-format can be reduced during the execution of the collision-resolution algorithm, if only sufficiently few information units are sent superposed at a time. This dynamic reduction saves, on the hand, transmission-effort or allows for a given transmission rate a higher throughput respectively. But on the other hand, one should evaluate carefully for each implementation, whether the effort for this dynamic adaption of the message-format and of the cyclic group which is used for the superposition pays off.

It is of rather theoretical interest, that not only “determinism” with exponentially scaling probability of failure can be achieved for the comparing of the mean value, but that even “real” **determinism** can be achieved: If either nothing is transmitted or if the same is transmitted again after a superposition collision, then all the collided information units are equal. As the number of collided information units must already be known to everybody before the averaging (perhaps by the already earlier mentioned accumulation of the characters which correspond to the 1), everybody can divide the previously saved collision result by the number of collided information units and then receive the resulting information unit the according number of times.

Thus, this multi-access method achieves a throughput of 100% under the usual (idealizing) assumptions.

If a sufficiently large alphabet and the just described comparing of the mean value

5.4 Basic measures settled inside the communication network

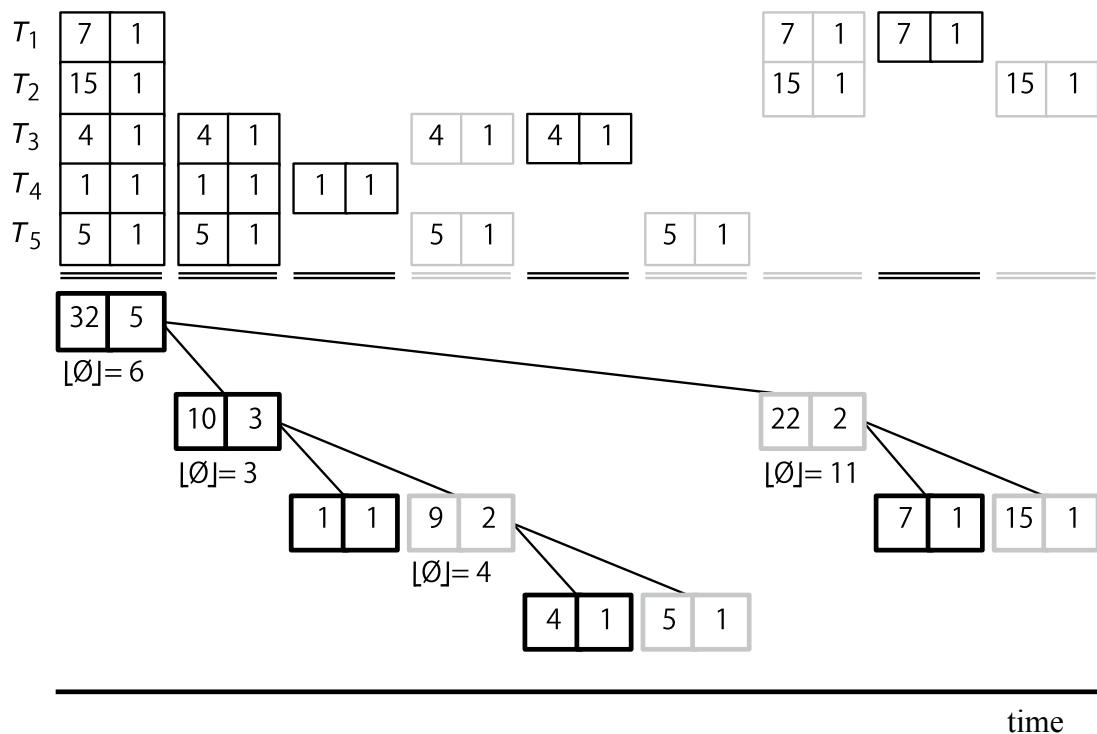


Figure 5.14: Detailed example of the algorithm for resolution of collisions with averaging and superposed receiving

is used, a real throughput of almost 100% can be achieved. The throughput of the multi-access method for binary encoding is

$$\frac{\text{number of bits which are necessary for encoding of the information units}}{\text{number of all bits which are to be transmitted}} \cdot 100\%.$$

If the encoding of the information units is redundant, then the resulting loss of throughput affects the source-encoding or the binary channel-encoding, but it does not affect the multi-access method. This is the case, because redundant encoding does not limit the encoding of the information units in any way.

For array-wise binary encoding according to Figure 5.13, the real throughput of the multi-access method is hence at least

$$\frac{\lfloor \log_2 G \rfloor + 1}{\lfloor \log_2 s \cdot m \cdot G \rfloor + \lfloor \log_2 s \cdot m \rfloor + 2} \cdot 100\%,$$

because exactly  $\lfloor \log_2 s \cdot m \cdot G \rfloor + \lfloor \log_2 s \cdot m \rfloor + 2$  bits are needed for the binary encoding of the integers of the interval  $[7, x]$ . Whereas  $\log_2$  (logarithmus dualis) means the logarithm to base 2 and  $\lfloor y \rfloor$  means the largest integer  $z$  with  $z \leq y$ .

Furthermore, it is guaranteed that every station can transmit  $m$  information units in  $s \cdot m$  for the transmission of one information-unit necessary time-units: Every station has hence, as long as it has something to send, as well a “real” deterministic guaranteed fair share of the bandwidth of the DC-network as an upper bound, after which it could at the latest successfully send. A single station can, on the other hand, use the bandwidth of the DC-network completely alone, if the other stations have nothing to send.

The average delay time can be minimized by always continuing with the subset which has the lesser cardinality. This can and should be combined with the comparison of the mean values. The average delay time of this multi-access method is investigated in detail in [Marc\_88].

If for the fulfilling of requirement  $g > s \cdot m \cdot G$  necessary alphabet size is too large, then a smaller value  $k$  can be used instead of  $s \cdot m$  in the inequality, for large  $s \cdot m$ , for instance  $k = 10$ . The mean value statement is then possibly invalid for collisions of more than  $k$  information units. If  $k$  was chosen appropriately, the bandwidth wasting cases, that the wrong, namely modulo  $g$  computed, mean value is lesser or greater than all involved information units, occur sufficiently seldom. These cases are detectable for all participants and are resolved by coin-flip.

Another possibility for using a smaller alphabet-size is the following: Only execute the mean-value-comparison on parts of the information units (perhaps in the beginning), and not on complete information units. Indeed, the probability that all the information units which are involved in a collision have respectively equal parts (for example equal beginnings), increases thereby. If the used parts each have a length of at least 20 bit and if the values are only approximately distributed according to an equipartition (for instance implicit addresses or end-to-end encoded parts), then this probability should be sufficiently small though, such that the usable performance is only imperceptibly decreased.

The loss of the “real” determinism is the price to pay for the smaller alphabet size in both cases.

The **algorithm for resolution of collisions with averaging and superposed receiving** is following: if we desist from the (indeed slight, compare §5.4.5.5) effort for the realization of a very large alphabet and from the effort for the global superposed receiving - in every aspect the **optimal multi-access method for the DC-network** - unless pairwise superposed receiving or conference circuits are possible, compare [Pfit\_89, §3.1.2.5 and §3.1.2.6].

Every communication network that is suited for superposed sending with large alphabet, works so efficient for so many services with this (and possibly further) multi-access methods, that its employment seems convenient at least in the local area, even for purposes which do not need sender anonymity – key generation and synchronized superposition can then of course be omitted.

#### 5.4.5.4.3 Booking-scheme (Roberts' scheme)

As already mentioned in §5.4.5.1, the use of this anonymity perfectly preserving multi-access method, with which a throughput of nearly 100% can be achieved [Tane\_81, page 272], was already suggested in [Cha3\_85] with the following implementation in the digital world of binary superposed sending:

The bandwidth is divided in frames whose length lies between minimal and maximal value. In order to simplify the following functional description, let “the next” frame not be the next physical frame, but the  $n$ -th next physical frame, whereas  $n$  is chosen fixed and minimally, such that before it is sent, all the stations have received the result of superposition of the “previous” frame (also for frames of minimal length in between) so in that time they are able to execute the multi-access method. Every frame starts with a data transmission part of variable length and ends with a booking part of fixed length. The result of superposition of the booking part of a frame defines the length of the data transmission part of the next frame. This is achieved by reserving space (in more detail: time) for one information unit for every signalized booking, in which exclusively only the station that made the booking is allowed to send. If no booking is made in a frame, then the data transmission part of the next frame will be empty. David Chaum proposes in [Cha3\_85],[Chau\_88] to regard the booking part as a bit-strip, whereas every 1 corresponds to a booking. The data transmission part can hence maximally contain as many information units as the booking part contains bits. To make one (or several) bookings, a station sends one (or several) 1 to arbitrary places in the booking part. If the result of superposition in one (or several) of these places is equal to 1, then it sends in the corresponding places (in more detail: times) in the data transmission part of the next frame.

Unfortunately it is now not guaranteed that there is no collision: If stations 8, 5, 7... have reserved in this place, then they all assume that only they send in the corresponding place of the next frame. The probability of this fallacy cannot be reduced to 0 in binary superposed sending, but it can be considerably reduced by using  $x$  bits for every

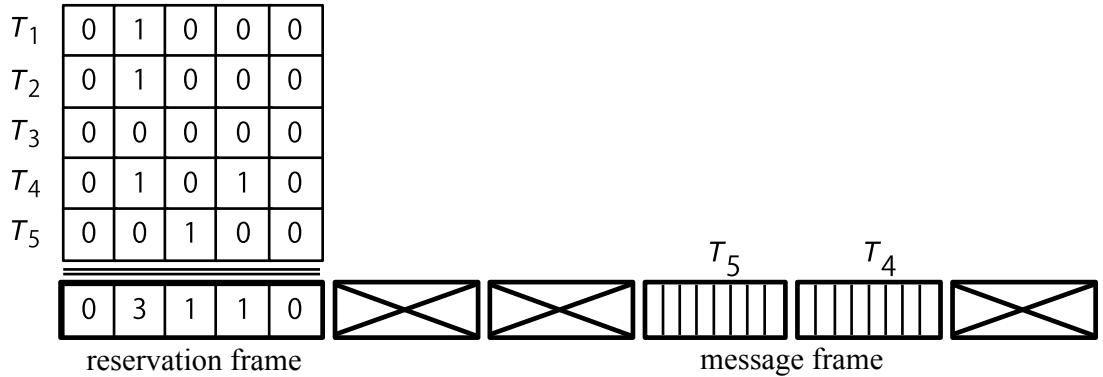


Figure 5.15: Booking scheme with generalized superposed sending for the stations  $T_i$

booking, of which  $y$  ( $y < x$ ) are sent as 1 to signalize a booking. Though the probability that a collision is not being detected during the booking can be arbitrarily reduced by appropriately choosing  $x$  and  $y$ , also  $x$ -times as many bits are used for the booking. To find optimal values for  $x$  and  $y$  for a given traffic-distribution, is a worthy task.

Instead we stress here again, that generalized superposed sending with  $g > s \cdot m$  can be used for reducing the probability of collisions during the superposition of information units to 0 as it was explained before in the description of the single multi-access methods. This multi-access method, which is depicted in Figure 5.15 is the oldest application of generalized superposed sending [Pf1\_85, page 40].

A combination of these booking-methods with superposed receiving seems to be mostly unnecessary. Only in the case that a booking results in the value 2 in generalized superposed sending can it be without loss of throughput, but with reduction of the average waiting time till successful transmission, declared, that both stations send their information unit in the corresponding place of the next frame and that only the station that sent the greater information unit will send in the frame after the next frame. This is exactly like the already described method of deterministic resolution of collisions for two information units.

#### 5.4.5.5 Optimality, complexity and implementations

Potentially all the member stations shall get to know the (end-to-end encrypted) messages for reasons of receiver's anonymity (and thus - except for pairwise superposed receiving - also every realistic attacker can get to know the messages). Hence, the superposed sending with exchange of one key between every pair of member stations and with new encoding of collided messages or global superposed receiving with appropriately frequent simultaneously superposed of several messages respectively, is the *optimal* method in terms of sender anonymity.

However, the use of the method of superposed sending is very, very expensive, as large quantities of keys must be exchanged in a secrecy guaranteeing manner: *every* pair of

member stations which exchanges keys, needs a secure transmission channel for this. This channel has to have as much bandwidth as the communication network offers all the users *together* for the exchange of their messages.

This *complexity of the key-distribution* can be reduced by using pseudo random number generators, or in the case of binary superposed sending by using pseudo random bit sequence generators. Then only relatively short keys have to be exchanged secretly. Very long keys which look randomly generated to an external observer can be generated from these short keys.

The method is then not information-theoretical secure anymore, but only more or less complexity-theoretic secure; if cryptographically strong pseudo random number generators or pseudo random bit sequence generators are used, the method creates *perfect complexity-theoretic anonymity* of the sender and *perfect complexity-theoretic unlinkability* of sending-events.

Unfortunately, the known fast pseudo random number generators or pseudo random bit sequence generators are all easily breached or they are at least of doubtful security (e.g., backcoupled shift registers), whereas the pseudo random number generators or pseudo random bit sequence generators which have been proven to be cryptographically strong so far (§3.4.3,[VaVa\_85],[BiMi\_84],[BiBS\_86]) are very expensive and very slow.

David Chaum proposes in [Cha3\_85] to *implement* the superposed sending *on a ring*. The breaching of fast (and possibly easy to break) pseudo random number generators or pseudo random bit sequence generators is thereby made more difficult for an attacker, because the attacker can only get to know the outputs of sequenced member stations in the ring by very expensive physical means, thus we can assume realistically, that the attacker will not get to know the outputs (compare to §5.4.4.1).

Every bit circulates once for means of sending by successively superposed and once for means of receiving around the ring in this implementation.

This implementation seems quite efficient, as it causes in average only a transmission-complexity which is four times as high as in the traditional sending-method on a ring, whereas an implementation on a star-shaped network causes for  $n$  member stations a transmission-complexity which is  $n$  times as high as in the traditional sending-method on a star-shaped network.

But as the transfer-amount on every line, i.e., the required bandwidth, is equal for all the implementations, the implementation of a *star-shaped network* (or more general: *tree-shaped network*) can be more efficient nevertheless. The shorter the delaying-time, i.e., the time that is elapsed between a sending-attempt and the feedback, whether a transmission-collision happened, the better the implementation of a channel. Nodes of star- and tree-networks can be considerably easier for this method than the traditional switching centers, and they can also be designed in such a way, that the sum of the circuit times grows only logarithmic with the number of members. The pure runtime grows roughly with the square root of the number of members, whereas both grow in ring network always linearly proportional with the number of members.

So far, the following aims for the realization of the (sub-)layer of the DC-network, which creates anonymity, have arisen:

- A1** The *delay*, i.e., the time, which elapses between the sending of a character and the receiving of the sum (modulo alphabet-size) of all sent characters, should be as small as possible, as this is convenient for all the communication-services and as it is necessary for some multi-access methods. The delay must trivially be smaller than the minimum of the acceptable delays for all the services which need to be transacted.

From this follows, at least for a service-integrating network, the requirement that the delay of the DC-network must be smaller than the response time which is sensed as negative by humans. As mentioned in the “Assumptions and notations with respect to the transfer-amount” in [Pfit\_89, §3.2.2.4], a suggestive value for the delay is in my opinion under 0.2 seconds, whereas CCITT allows for maximally 0.4 seconds.

- A2** If the multi-access channel of the DC-network shall be assigned using a booking scheme, then the *alphabet size* should be at least inside the booking-channel *large enough*, especially larger than 2, compare §5.4.5.4.3.

The corresponding holds for the algorithm for resolution of collisions with averaging, which was described in §5.4.5.4.2, which is for many services the best possible multi-access method.

A large size of the alphabet which is used for the superposing is also in the restricted way favorable for an as efficient as possible processing of conference circuits, compare [Pfit\_89, §3.1.2.6].

- A3** The member community should be provided with a *high bit rate* for the payload transmission.

- A4** The realization should cause as *few expenses* as possible.

The *designing decisions*, which are relevant for achieving these aims, namely the specification of the alphabet size, the implementation of the modulo-adder and of pseudo random number generators, choice of a suitable topology for the global superposing and synchronization of the superposition of information units and keys are treated in the following in the corresponding order.

**Specification of the alphabet size**(concerning mainly A1, A2, A4): Small expense and small delay on the one hand, and large size of the alphabet that is used for the superposition on the other hand are obviously only slightly contradictory aims, because for an appropriate, i.e., for the encoding of the information units, keys, and transmission fitting, choice of the size of the alphabet which is used for the superposition, parts of an alphabet-character can already be output by the member station, *before* the rest of the alphabet-character is defined, for instance by the use-message. The corresponding holds for the global superposition: if only the beginnings of all the characters which are to be superposed have arrived yet, the beginning of the result of superposition can already be output. For a short sketch on how to choose or rather execute such a fitting encoding and fitting, fast and inexpensive adders respectively subtracters (modulo the size of the alphabet which is used for the superposition), compare Figure 5.16:

As usual, let the encoding of the information units, of the keys, and of the transmission be binary. Then it is very convenient to choose the alphabet size  $2^l$  with a fixed natural number  $l$  and to also encode the alphabet characters in the usual binary way: the unit element as 0 and so on. If the binary digits of the alphabet characters are now transmitted in ascending valences, then a total adder respectively a total subtracter, an AND gate and for instance a shift register of length  $l$  suffices to execute the superposition of two bitstreams binary-digit-wise modulo  $2^l$ : there is only a 0 at one position in the shift register. The position where the 0 is in the beginning is linked subjunctively with the carry of the total adder respectively total subtracter by using the AND gate. The output of the AND gate serves as carrier of the total adder in terms of total subtracter. By this we achieve that the carrier of the total adder and respectively total subtracter is always 0 at the beginning of the binary-digit-wise superposition of a character.

As the total adder is respectively total subtracter, the AND gate and the shifting register work just as fast as adders modulo 2 (XOR gates), the delay of the just sketched DC-network which works modulo  $2^l$  is exactly as large as the delay of a DC-network which works modulo 2. Only the expense of the adders and respective subtracters grows linearly with  $l$ , or in other words, it grows logarithmic with the size of the alphabet which is used for the superposition. As the circuit-complexity for the superposition is - for realistic values of  $l$  (e.g.,  $l \leq 32$  should always hold) - always dimensions smaller than the complexity of the generation of cryptographically strong pseudo random bit sequences or pseudo random number sequences (compare §3) and as the assumption of optimally short delays is unrealistic for realistic bit rates for the only multi-access method which requires a DC-network which works modulo 2 (it is described as last method in [Pfit\_89, §3.1.2.3.6]), there is from my point of view no real reason to construct a DC-network which works modulo 2. We demonstrate by the following example that the assumption of optimally short delays is unrealistic for realistic bit rates: Assume that the DC-network has only a bandwidth of 64,000 bit/s and assume that the transmission media has a signal-spreading speed of 200,000 km/s. Then a DC-network with optimally short delay can - for reasons of the signal-spreading speed – only have a diameter of 3.15 km. Non-optimal line management etc. allows only for a substantial smaller diameter.

And also a combined DC-network which works in the largest part of its bandwidth modulo 2 and which works in a small part of its bandwidth - which is used for the booking - modulo  $2^l$  seems to be not very useful, as a far reaching decision - which would probably increase the design-complexity of the higher layers - about the ratio of the bandwidths must be made in an early phase of the construction of the DC-network.

**Implementation of the modulo-adders** (concerning mainly A1, A4): We have already shown in the specification of the alphabet size, that modulo-adders can be realized for all possible alphabet sizes with small expense in such a way that they have the technology-depending minimal gate delay as the delay of the modulo-adder. Thus every modulo-adder can process the whole bitstream, so that we do not need to think about parallel operation of several modulo-adders.

Conversely, it does not seem to be remunerative if every member station is connected to several DC-networks, to use existing modulo-adders for several DC-networks by multiplexing, because multiplexors do not cause considerably less expense than the described

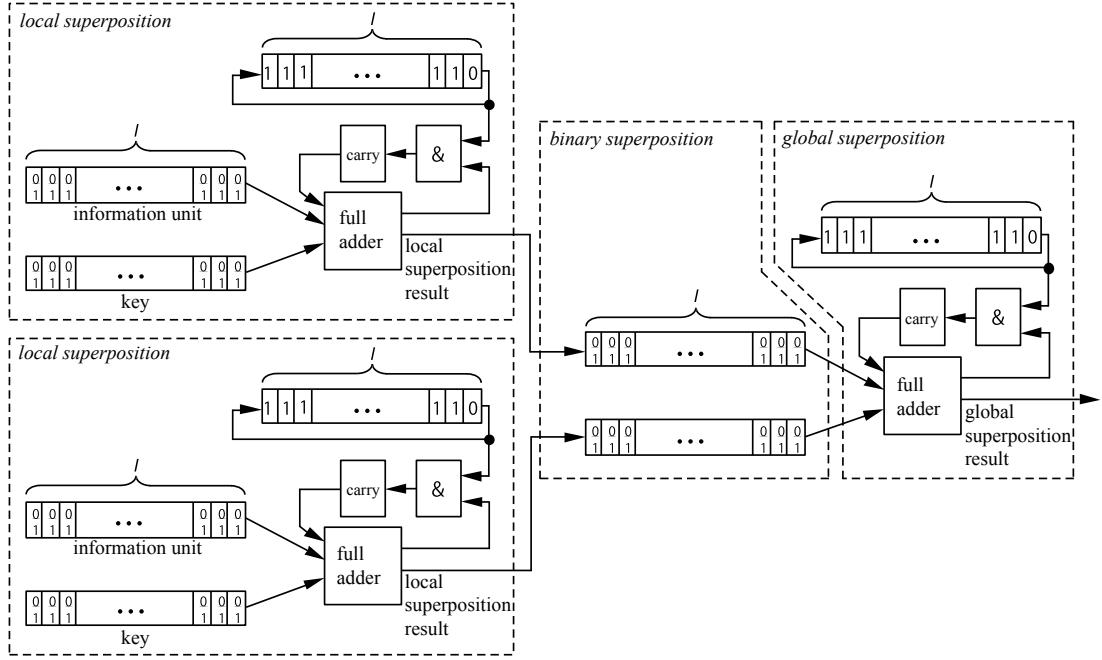


Figure 5.16: Practical encoding of information units, keys, and local and global superposition results during transmission: unique binary encoding

modulo-adders. Also it will be demonstrated in detail in §5.4.5.6, that (as far-going as possible) abandonment of common parts makes the simultaneous failure of several DC-networks less likely, and is thus desirable.

**Implementation of the pseudo random number generators** (concerning mainly A3, A4): As the generation of cryptographically strong pseudo random bit sequences or pseudo random number sequences are nowadays needed for every DC-network with appreciable transmission rate (compare §3), but as the generation is considerably slower than the superposition and the transmission, several pseudo random number generators must – if necessary – be practiced in parallel and their outputs must be nested with a multiplexor into a bitstream which is the number of pseudo random number generators times faster.

Neither the design nor the implementation of the pseudo random number generators needs to be uniform inside of the DC-network. In theory, two members can agree upon the design of a pseudo random number generator, and they then can obtain two equally fast (otherwise the faster implementation needs to work slower) implementations and they then only need a common and secret initial value as well as possibly a publicly known exact point in time for the synchronized commencing of the superposition of their keys. The advantage of this completely peripheral choice of pseudo random number generators is the following: more secure and more efficient pseudo random number generators can be employed bit by bit in the DC-network and that probably a large number of them

will be provided, so that it “is not so bad”, if some of them should be breached. The disadvantage is obvious: a market works only then well, if the consumer is able to estimate the quality of the goods (if necessary with help of an expert) in a cheap way. I have already expressed my skepticism against the cryptographic systems which are nowadays offered in the market and which are mostly based on “secret” algorithms which were each (if the secrecy succeeded) only analyzed by a few “experts” which are mostly not known by name in [Pfit\_89, §2.2.2.2]. Similarly, I have offered several suggestions and substantiations for a standardization in [Pfit\_89, §2.2.2.3 (and in the annex) and §2.2.2.4], which allows in the case of the DC-network for the short term exchanging of keys between arbitrary participants and which thereby allows for a more flexible and more purposeful arrangement [Cha3\_85],[Chau\_88] of the key-distribution. Furthermore, some discussed problems of reliability become easier to solve by using standardized implementations.

The today still high expense of cryptographically strong pseudo random number generators can make it necessary for DC-networks with high bandwidth to compromise with respect to the cryptographic strength of the pseudo random number generation. These compromises can consist in separate or combined application of the measures, that

- member stations exchange only very few keys, that
- some keys are generated with more efficient (and possibly cryptographically weaker) pseudo random number generators or that
- a part of the bandwidth is superposed with more weakly generated keys.

A separate DC-network would be created in every part of the bandwidth by the last measure, whereas the anonymity of the sender would be different in the different DC-networks, and could be — as discussed in [Pfit\_89, §4.2] — used for differently sensitive communication. Stronger pseudo random number generators could be upgraded additionally to the more efficient (and possibly cryptographically weaker) pseudo random number generators later.

**Choice of a suitable topology for the global superposition** (concerning mainly A1): the delay of a DC-network consists of the needed time for the transmission and of the needed time for the superposition. Hence as well the *transmission topology* as the *superposition topology* need to be chosen in a mutually matching way, so that the sum of all delays and thus the delay of the DC-network is as small as possible. The *key topology* can be chosen independently of these both topologies because the order and summary of the addition is irrelevant in an *abelian group*. The three topologies of the DC-network which can be distinguished are shown in Figure 5.17.

There are two extreme cases for the superposition topology (and also for the transmission topology):

1. The characters are forwarded along a ring from member station to member station. As the superposition lasts at least a gate delay in each member station, thus the delay which is caused by the superposition grows for  $m$  member stations with  $\mathcal{O}(m)$ , i.e., at least linearly proportional to  $m$ .

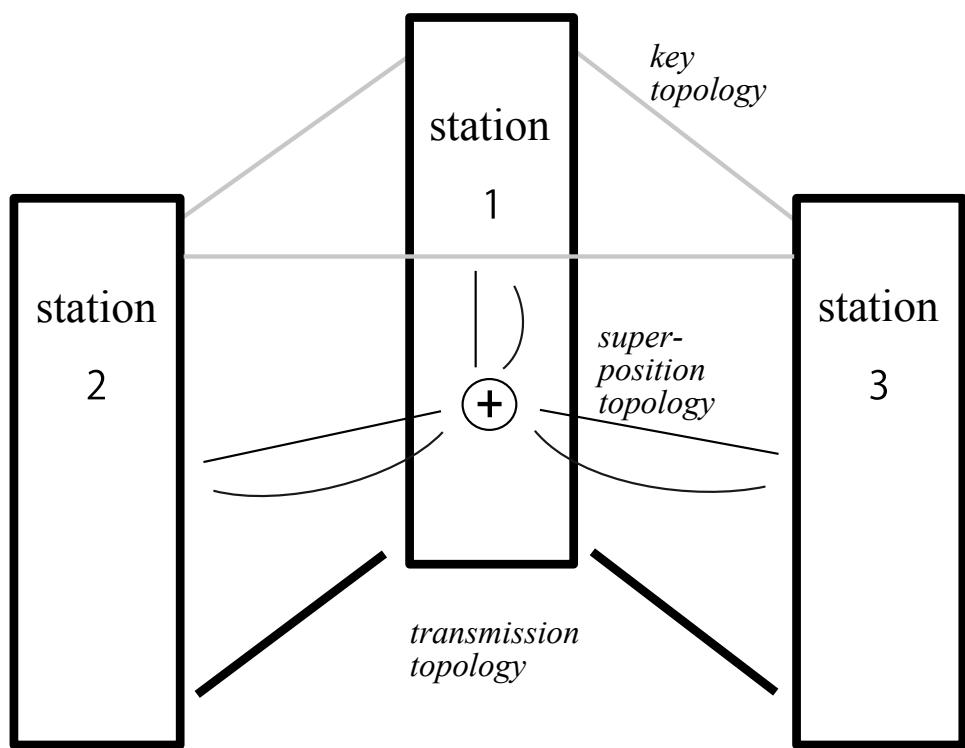


Figure 5.17: The DC network's three topologies

2. The characters are transmitted to a central station (according to a present day switching center) and are superposed there, whereas - as shown in Figure 5.18 - the delay that is caused by the superposition grows - for  $m$  member stations and gates with limitedly many inputs and with limited driver-performance - with  $\mathcal{O}(\log(m))$ , i.e., at least proportional to the logarithm of  $m$ . This small growth is being preserved, if the superposition is executed peripherally but still tree-shaped, such that not only a star-shaped topology, but also a tree-shaped topology is suited for the global superposition.

A counter (for instance, implemented by a shifting register) modulo  $l$  is necessary for suppression of the carrier at the borders of the characters, if the superposition is modulo  $2^l$ , as described above under “specification of the alphabet size”. This counter needs only to exist once in a central realization of the tree-shaped superposition, because it can control all the total adders, whereas it must of course exist in every superposition position in a peripheral implementation.

Transmission topologies which are suited for these superposition topologies are treated in detail in [Pfit\_89, §3.3.3] as well as the growth of the sum of all delays and thereby the delay of the DC-network.

**Synchronization of the superposition of information units and keys** (concerning mainly A3, A4): As we already stressed several times, information units and keys must be superposed synchronized. If the synchronization is lost even only between a pair of pairwise exchanged keys, then no information units can be successfully transferred on the affected DC-network anymore. The actions which will then need to be taken are treated in §5.4.5.6.

If the transmission-network works in a global way synchronous, as for instance planned for ISDN, then this synchronization of the transmission-network can also be used for the synchronization of the superposition. But the bit rates for instance of the planned ISDN are small in comparison with the bit rates of a DC-network of comparable usage-performance. Also the rate of acceptable synchronization errors in the planned ISDN is high compared with a DC-network of comparable reliability. Additional measures thus seem to be inevitable in the general case, whereas all the synchronization problems of the superposition are trivially solvable in special cases, for instance the implementation of a peripheral ring-shaped superposition topology on a ring-shaped transmission network which has high bit rate and high reliability.

A conceptually simple, but very expensive method in execution, is to equip character-streams with time stamps (e.g., sequence numbers) and to buffer them before they are globally superposed as well as to employ the usual flow control mechanisms for computer networks with respect to the buffers.

Methods which are less expensive in execution can be achieved by exploitation of specific superposition topologies and transmission topologies. If the transmission topology of the distribution channel of the DC-network is hierarchical and if the clock of transmission network is passed from top to bottom in a phase-steady manner, every station which is involved in the superposition can deduce its sending-clock from the clock of the distribution channel. If all stations do this in the same manner, then their sending-clocks

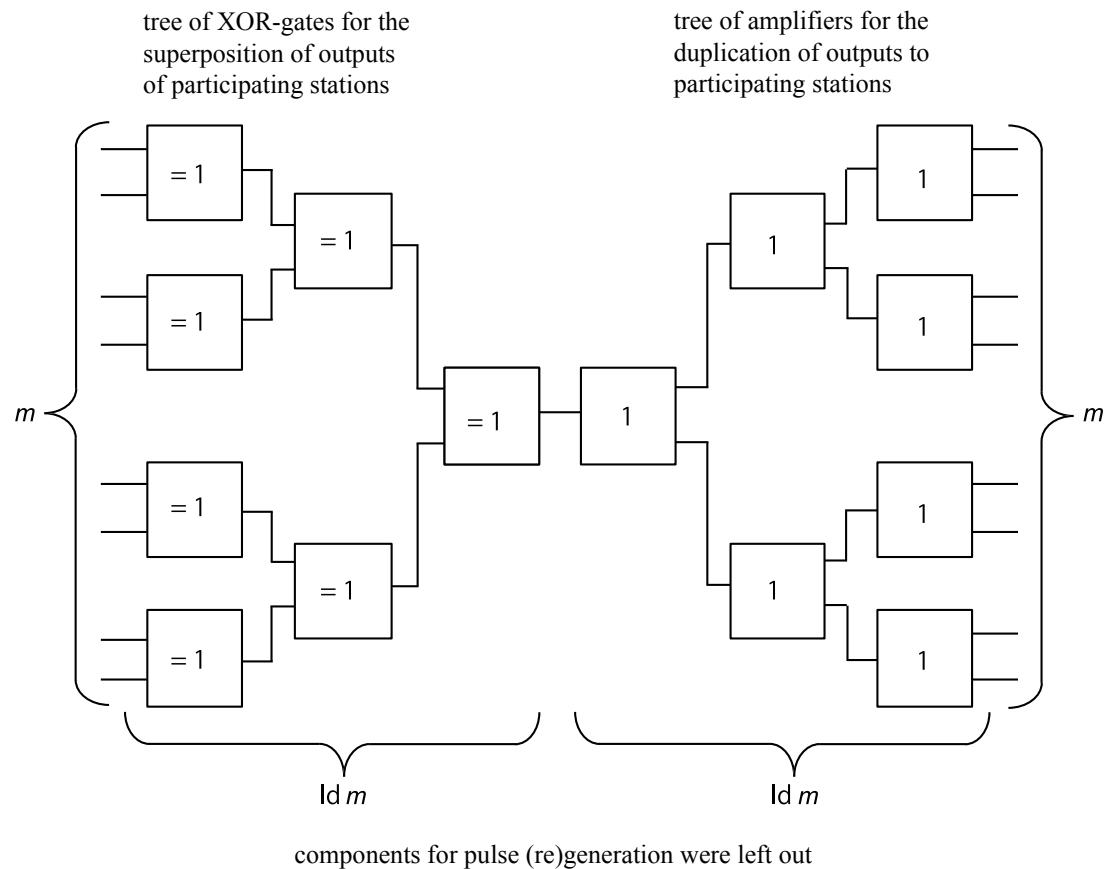


Figure 5.18: Superposition topology with minimum delay for gates with two input channels and a driver power of two for the example of binary superposed sending

are phase-steady. If the superposition topology is also hierarchical and if the delays on the corresponding transmission lines are constant, then it can be achieved by choosing the position of the phase relatively to the (receiving-)clock of the distribution channel of the DC-network individually for every station, that all the sent character-sequences traverse the hierachic superposed network synchronously.

#### 5.4.5.6 Fault tolerance of the DC-network

An arbitrary bit-transmission network which includes layer 0 and the lower sub-layer of layer 1 of the ISO OSI reference is possible for the DC-network with conventional fault tolerance measures without regarding anonymity, unobservability, and unlinkability, compare §5.4.9. Thus we will mostly consider the upper sub-layer of layer 1 and the layer 2 in the following.

The seriality property of the DC-network is, that all keys must have been faultlessly exchanged, that all *pseudo-random-number-generators* (PRGs) and all modulo-adders most work faultlessly and the synchronization must be preserved.

In contrast to the MIX-network, in which fault tolerance measures can always be executed in the background, we can clearly distinguish two phases - we will name the phases modes later - in the DC-network. All the member stations usually transmit information units anonymously. If a permanent error occurs either in the method for the anonymous multi-access or in the DC-network, then no member station can transmit information units anymore, until this error is tolerated by releasing or repairing the defect component(s) - unless there are several independent DC-networks (*statistically generated parallel redundancy*, which itself can be *activated statistically or dynamically*).

The **realization of several independent DC-networks** is not considered in detail here, as it does not cause any special difficulties in the construction.

Klaus Echtle points out that it should be convenient to allow every station to receive on all of the independent DC-networks, but allow every station only to send on relatively few DC-networks (*sender-partitioned DC-network*). Thereby, it prevents an error which affects only a few stations, respectively a modifying attacker who controls only a few stations, can interfere with the DC-networks which would otherwise not be independent with respect to just this error respectively modifying attacker. The Figures 5.19 and 5.20 show a DC-network for ten stations which is configured for the error-guideline that arbitrary faults of an arbitrary station shall be tolerated. [Nied.87] contains a detailed and precise rating of the reliability, i.e., the probability, that all not faulty stations still can communicate with each other, and of the anonymity of the sender, i.e., under how many stations a sender is anonymous for sender-partitioned DC-networks which are configured in such a way.

As already obvious from the example, some multi-errors can be tolerated in this sense even by sender-partitioned DC-networks which were configured for the error-guideline of an arbitrary single error, for example arbitrary faulty behavior of stations 1,2 and 5. For instance, arbitrary faulty behavior of the stations 1 and 8 can not be tolerated in this sense, because all the stations that are not connected to the DC-network 5 - namely stations 2, 3, 5 and 6 - can then no longer send undisturbed.

The configuration principle of the example can be extended to the fault-guideline that arbitrary faulty behaviors of  $f$  stations can be tolerated. That the sending-possibilities of no set of  $f$  stations overlaps the sending-possibility of any other single station, is a necessary and sufficient condition for this.

The complexity of this fault tolerance measure is - contrary to the first impression - not vast, as several (in the extreme case all) DC-networks can be realized on one single underlaying bit-transmission network (which must be at least in the extreme case able to tolerate its own faults) by time-multiplexing. The fact that the sender anonymity is clearly lower on each of the DC-networks than on a single DC-network which contains all the stations, is disadvantageous (for instance, an attacker who observes a message on DC-network 1 in Figure 5.19, can even without cooperation of the stations 5, 6, 7, 8, 9, and 10 know that none of them sent the message). If a sender-partitioned DC-network which is configured in such a way is used, all the member stations need to take care that linkable information units are only sent on the same DC-network, which restricts the load-balance on the DC-networks drastically. Otherwise, the sender is identifiable in sender-partitioned DC-networks which are configured for the error-guideline that an arbitrary error of an arbitrary station shall be tolerated, and also the anonymity of the sender is again drastically restricted for a DC-network which is configured for a more universal error-guideline. Thus a *statistical activation* of the redundancy of sender-partitioned DC-networks is not trivially possible within reason.

The advantage of this fault-tolerance measure is, that a fault-diagnosis is not necessary or at least not necessary for only a few errors, and that a complete *fault-covering* is achieved, i.e., there are no errors which can not be tolerated in this way.

If the fault-guideline is exceeded, a fault-diagnosis which can be executed with the methods which are described in the following might of course become necessary, so that both fault-tolerance methods are then combined.

**Error-detection, -localization, and -correction in a DC-network:** Transient as well as also permanent errors in the bit-transmission network can - as already mentioned - be tolerated through separate fault tolerance measures, or otherwise they lead to transient respectively permanent errors in the DC-network. The transient errors in the DC-network are tolerated by end-to-end-protocols. Thus only the permanent errors remain in the DC-network (loss of PRGs, of modulo-adders, loss of consistency or synchrony of the keys etc.). These errors are easy to remedy, once they are detected and localized. The corresponding keys (of defect PRGs) are just not superposed anymore, adders are being replaced, keys are again distributed or they are again synchronized. Thus the main-problem is the *error-detection* and *-localization*.

Errors in the upper sub-layer of layer 1 can be detected by the addition of additional redundancy - which can be accessed by every station – to every encrypted information unit by the sender, for instance a linearly created checksum, for example CRC, at the end of the information unit. This can always happen, because the redundancy spans only a fraction of the length of the information units, and because the usable transmission performance is thus hardly reduced. If the checksum was created linearly, a valid checksum will arise also for superposition collisions. If an invalid checksum arises, then there is an error in the superposition.

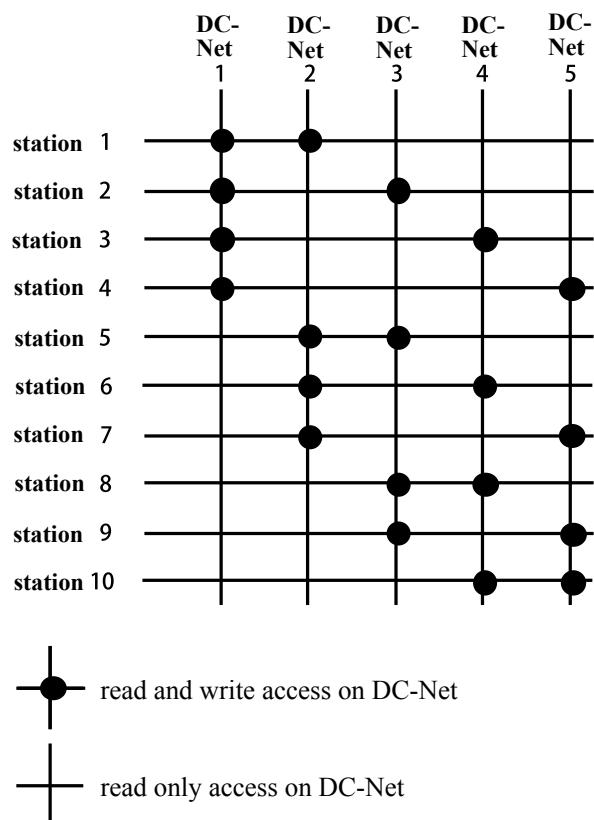


Figure 5.19: Sender-partitioned DC network with 10 stations, configured so that any desired faulty behavior of any desired but one and only one station can be tolerated without fault diagnosis

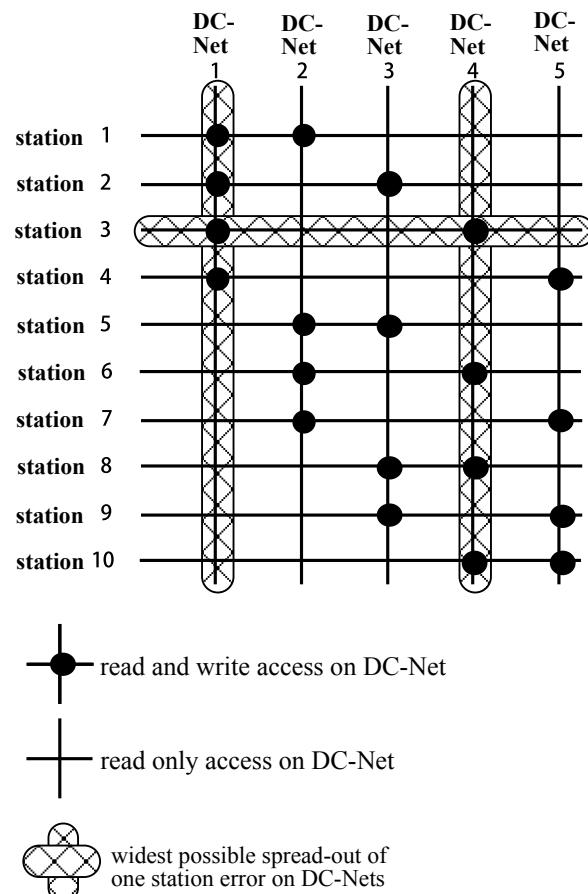


Figure 5.20: Farthermost propagation of an error (or a modifying attack) of station 3

The detection of errors of the multi-access method depends on the method which is used, but it is possible for all the recommended methods with high probability.

The multi-access method will be re-initialized if an error of it is detected in order to tolerate transient errors of the bit-transmission network or of the upper sub-layer of layer 1. If this fails or if a constant error of the upper sub-layer of layer 1 is detected in another way, then the network will switch from the anonymity guaranteeing *anonymity-mode* (A-mode) into the *fault-tolerance-mode* (F-mode) in which error are being localized and tolerated.

All stations execute the following **fault-localization- and -removal-protocol** [Pfi1\_85, page 123]:

Every station saves to the current state of its pairwise exchanged keys respectively PRGs (in technical terminology: creates a recovery point [AnLe.81]) and afterwards executes the following self-diagnosis: it loads pairwise random keys in its key-memory respectively PRGs and superposes them locally with information units which were randomly chosen as well.

The station is defective and sends a message which signals the defectiveness over the DC-network, if the result is not the random information units itself. All other stations dump the keys which they shared with this station and change (under assumption of a single error) back to the A-mode.

The station uses the recovery-point to restore the previous state of its pairwise exchanged keys respectively PRGs, if the result is the random information unit itself.

At this point in the protocol there are three likely types of errors:

- a station is so defective that it can not make a diagnosis about itself and can not communicate the result to the other station, or
- the synchronization of the key(-generation and-)superposition was lost between at least two stations, or
- the underlying communication network or the global superposition are defective.

In order to distinguish these types of errors and in order to localize errors, the number of superposed key pairs is being successively halved (the corresponding key-distribution-graph has always just half as many edges) and for instance 100 new and not even yet used key-characters are being superposed.

The error is with probability  $1 - g^{-100}$  in the other half if the global result of superposition is 100 times the character which corresponds to 0 (as long as a stuck at zero fault in the last global superposition-device can be ruled out. This can be tested by letting all the stations previously send 100 random characters, which will give with equal probability a result  $\neq 0$ , as long as there is no stuck at zero fault.).

If the global result of superposition is not 100 times the character which corresponds to 0, then there is at least one error in the storage or generation of the keys or in the synchronized superposition of the keys or the underlying communication network or the global superposition is defective. Thus continue to halve.

If the key-distribution-graph has only one edge left and if the result of global superimposition is not 100 times the character which corresponds to 0, then both stations exchange a new key or PRG-starting-value in order to remedy the synchronization error. Afterwards both repeat the test.

If the global superposition result is again not 100 times the character which corresponds to 0, then both stations dump the just exchanged key respectively PRG-starting-value and exchange a key or PRG-starting-value with a third and fourth station. Both new pairs of stations superpose successively 100 key-characters.

If the result is not 100 times the character which corresponds to 0 in both cases, then the underlying communication network or the global superposition is with high probability defective. To further examine both of these cases, is a typical problem of fault tolerance.

If the result is not 100 times the characters which corresponds to 0 in just one case, then the station which was originally suspected to be defective is now assumed to be defective and is taken out of commission. This is communicated to all the stations, so that they dump the keys which they shared with this station and change (assuming a single error) back into the A-mode.

The method for anonymous multi-access is re-initialized by changing back from the F-mode to the A-mode, in order to annul effects of errors in layer 1 (physical) on layer 2 (data link). All the just described steps are summarized in Figure 5.21.

Of course, there are hundreds of variations of this fault-localization- and -removal-protocol. One can judge their expediency as soon as the probabilities of (multi-)errors are known. If for example the probability of (multi-)errors is significant, also the halves of the key-distribution-graph, which are not being tested by the just described protocol, should be tested, which at most doubles the expense of the protocol.

As the probabilities are not known, the details of the fault-localization- and -removal are consequently not interesting, but it is interesting that they are possible with logarithmic time-complexity (in the number of the stations).

It is worth mentioning that the anonymity is not weakened in “error-detection, -localization, and -correction in a DC-network” despite the fault-tolerance, if either genuine randomly generated keys or cryptographic strong PRGs are used. By introducing the two modes and by using key-characters in either the first or in the second mode, the anonymity remains always guaranteed in its original extent in the A-mode.

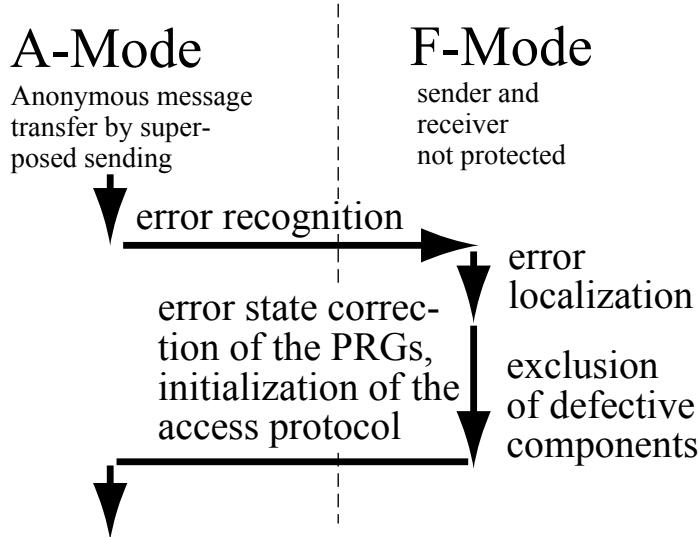


Figure 5.21: Error detection, localization and correction in a DC network

#### 5.4.6 MIX–networks: protection of communication relations

Instead of concealing a sender and a receiver one can try to keep a connection between them in secret. This way one can try to produce anonymity for the communication.

This idea is realized in the procedure of the **MIX that change the encoding** [Chau\_81, Cha1\_84].

This procedure was suggested by David Chaum in 1981 for electronical post messages that are not necessarily sent along the shortest path to the receiver. They are conducted over a lot of intermediate stations which are as independent as possible in terms of their concept, their production [Pfit\_86, page 356], and their carrier [Chau\_81], [Cha1\_84, page 356]. Additionally, these intermediate stations have to *buffer messages of same length, re-encrypt and change order* of messages. These intermediate stations are the so-called MIX. Each MIX encodes messages of same length, i.e., it encrypts or decrypts them using encryption systems so that their ways can not be retraced over their length and their coding (their outlook).

In order to not allow tracing of the route over temporal or spatial connections, each MIX has to wait for a greater amount of messages of the same length at a time from sufficient different senders<sup>8</sup> (or, if necessary, produce messages by itself or let participant stations produce them – as long as there is nothing useful to send, thus meaningless

<sup>8</sup>The demand “messages from sufficient different senders” is not easy to fulfill: 1. In many communication networks, e.g., Internet, unique identities of the participants are not given, i.e., participants are able to have many different identities. MIX-variations that are even in these cases deployed in a useful way are described in [Kesd\_99]. 2. Even when unique identities are given from the second traversed MIX their proof is non-trivial, cp. exercise 5-25.

messages that are from the participant station of the receiver or with convenient structure of encoding and addressing are ignored by one of the following MIXes), additionally they have to buffer the messages and emit them in the changed order after the re-encryption, i.e., in different chronological order respectively or on different channels. An appropriate order would be the alphabetical order of the encoded messages. (Such a given order at the outset is better than a random one because the hidden channel of the “random” order is closed in this way for a possible Trojan Horse in the particular MIX. Since the number of the messages that have to be mixed simultaneously and the time ratios can be given exactly and it is possible to forbid the MIX the produce of messages the MIX does not need room for maneuver and this means that there is no need to have a possibility where a Trojan Horse that probably exists in a MIX can send hidden messages to a not-entitled receiver, cp. §1.2.2 and [PoKl\_78], [Denn\_82, page 281].)

Messages of same length that are mixed together, i.e., buffered (or produced) together, re-encrypted and emitted sorted (e.g., in alphabetical order) in order to hide temporal and spacial connections are called *batch* [Chau\_81, page 85].<sup>9</sup>

In addition to buffering, re-encrypting, and changing the order of messages with the same length, each MIX must be aware of *mixing every message only once* – or in other words: messages that repeat must be ignored by the MIX. An input message represents a repetition if and only if it has been processed before and if the output messages could be linked up by non-involved persons (it is discussed in details below when this is the case because it depends on the scheme used for changing encoding).

If a message is processed more than once within one batch, then there appear undesirable counterparts that are proportional to the frequencies of the in –and output messages: an input message which appears  $n$  times correlates with one output message which appears  $n$  times too. If all input messages of one batch appear with different frequency, then re-encryption of this batch does not protect at all.

If a message is processed in *more than one* batch, then one can create non-empty averages (and differences) in which one can build respectively in– and output messages and in a general case one is able to prove the linkability between messages. Both operations can be applied to results of these operations. For messages that belong to the average(difference) of cardinality 1 the correspondence between input and output message is clear – MIX does not contribute anything to the protection of the communication relations.

The basic functions of a MIX that were derived and explained in this section are plotted in Figure 5.22.

---

<sup>9</sup>a MIX-implementation (MIXmaster) that is spread on the Internet is working a little bit different: messages are collected in a *Pool* up to a certain number and after the mixing in a newly arrived message it is decided randomly between this new message and the messages in the Pool which message will be emitted. Is the Pool magnitude +1 chosen equivalent to the batch the delay at the implementation is in average double that size – therefore the unconcatenation of input– and output–messages is higher. Since it must often be guaranteed in communication networks that a maximal value is not exceeded and at the Pool implementation can not be a guarantee given, the batch implementation seems advantageous to me.

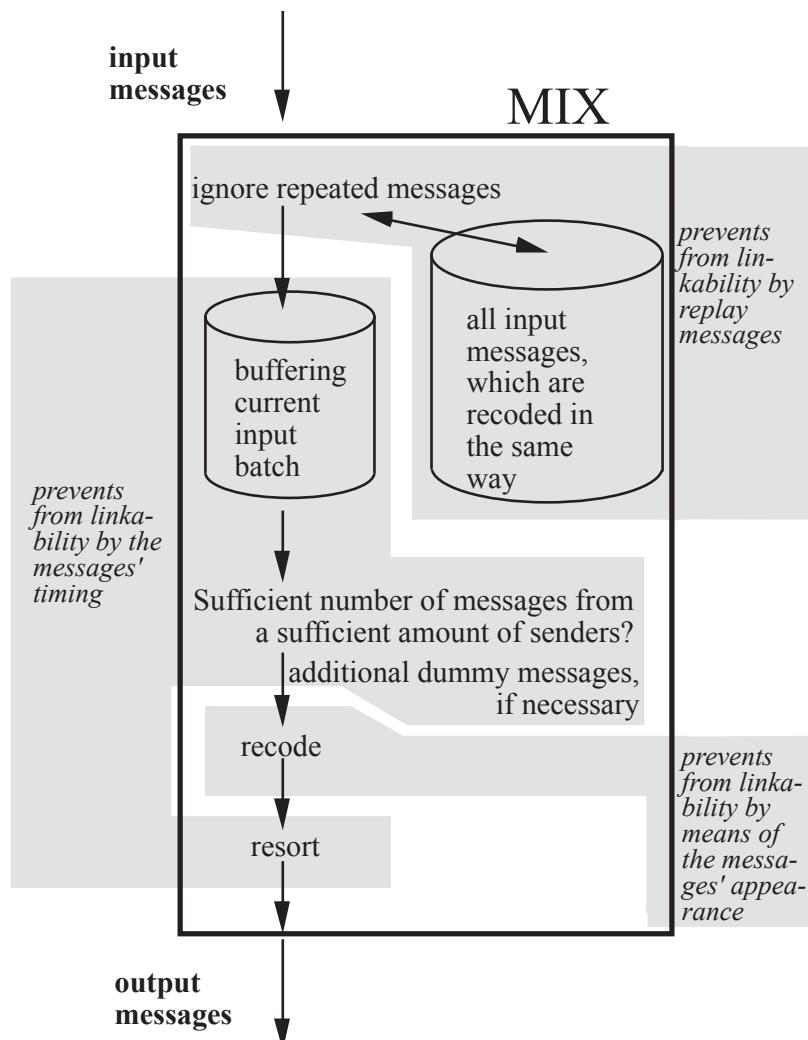


Figure 5.22: Basis functions of a MIX

#### 5.4.6.1 Fundamental considerations about possibilities and limits of re-encryption

The **aim** of the re-encoding on MIXes is that

- all other senders and receivers of messages that were mixed together in the batches of the MIX [Pf1\_85, Page 11] or
- all MIX that were processed by one message [Chau\_81, Page 85]

have to work together in order to reveal the communication relations against the will of the sender and receiver.

As long as both sender and receiver are unknown, from an attacker's point of view each message that was sent by a sender in a certain time interval should lead to each message of corresponding length that was received by a receiver – unless the attacker is the sender **and** receiver of the message. The length of the time interval is determined by the maximum of tolerable delay of the services: each message that was sent can arrive only after being sent (of course), but it has to arrive to the receiver before the maximum tolerable delay has elapsed.

This aim that divides messages and members of communication in classes according only to their sending and receiving abilities with time intervals and message length is presented in §5.1.2, with the relation to these classes and described attacker models, but guaranteed *unobservability of the communication relationships* for uninvolved parts and what is highly desired – *anonymity of communication partners from each other* and also *unlinkability of communication relationships*, that is the maximum of what can be achieved:

- If all other senders and receivers work at messages that have been emitted, buffered, re-encrypted, and their order changed by one MIX (i.e., being mixed in one batch), then on principle a bridge can be created between the messages sent by another user. It is particularly bad if an attacker is able to provide  $n$  of  $n + 1$  messages that were mixed together in one batch by a lot of MIXes. From what is said in [Pf1\_85, Page 11], it follows that they can completely observe one further sender and receiver in spite of using any desired MIX that can not be controlled by the attacker if all senders and receivers of messages with the same length that were sent during a certain time interval work together.
- That the method of the re-encryption MIX is not used for this message is trivial, since all MIXes that process a message work together.

From the aim explained above it follows that re-encryption of a message (or as explained later of message's parts) with relevant content never consists of encryption followed by decryption, so long as redundant re-encryption is omitted and the encryption system has only properties described in §3.1.1.1: if there is an encryption followed by decryption, the decryption has to be done with the suitable key, otherwise the message cannot be understood in future. This means that the message is presented in both MIXes in

the same form, so both MIXes are able to uncover the communication relations with or without this en- and decryption. Thus the en- and decryption is irrelevant to the protection of the communication relations in terms of the aim described above. It is even more so as the participation of sender and receiver does not change.

From the aim described above it also follows that all messages of same length in the considered time interval have to pass the MIX's at the same time (and therefore in same order, too): If messages of the same length do not pass the MIX's at the same time in the considered time interval then there is a MIX  $m$  which is not passed by two messages  $N1$  and  $N2$  at the same time. If all MIX's are working together now they are able to observe this and differentiate between the senders and receivers who belong to  $N1$  and  $N2$ , cp. picture 5.23.

If all messages of same length pass the MIX's in the considered time interval at the same time and if each participant station sends and receives at least one message of each possible length in the considered time interval, then the aim of the procedure of the MIX's that change the encoding is achieved even without classification of the participants.

If, in addition to the case mentioned above, each participant station sends and receives a constant number of messages of each possible length in the considered time interval *perfect computational secure in-observability* of the relations of communication according to uninvolved persons is achieved. And – so long as it is desired – *perfect computational secure anonymity* for the communication partners as well as *perfect computational secure unlinkability* of the relations of communication. (cp. §5.1.2)

Re-encryption that is done by the MIX's has to be constructed in order to guarantee that the receiver is able to understand the message: if he receives an encrypted message he has to **know** all **keys** that are necessary for decryption as well as the right order of their usage or he has to get to know them by himself during the process of decryption. Additionally, each MIX must be able to perform the expected re-encryption, thus it has to know the key for en- and decryption or uncover it while the process of re-encryption.

An encrypting re-encryption of a message must be specified by the receiver for the enforcing MIX, a decrypting re-encryption of a message must be specified by the sender.

Additionally the keys used by the MIX must not reveal anything about the identity of the sender or receiver of the message. It is probably acceptable that the first MIX of a message knows its sender and, if there is no distribution used for the protection of the receiver, that the last MIX knows the receiver. The used key should not reveal anything about the sender or receiver to a mid MIX and this means that it is not possible in practice to use a statically determined exchanged secret key and therefore an information-theoretical encryption system. (In §5.4.5.4 was described a method for key exchange that is based on sender anonymity by pairwise overlaying receiving. If the sender anonymity is information-theoretical, the key exchange takes place in a way that guarantees secrecy and integrity as well as anonymity – likewise in the information-theoretical model world. Since the information-theoretical sender anonymity is possible

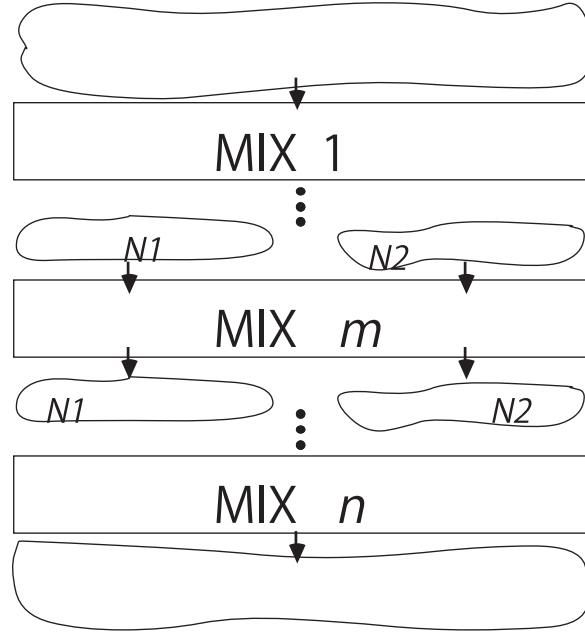


Figure 5.23: Maximum security: passing through the MIXes simultaneously and, because of that, in the same sequence

but it is connected with huge effort the mentioned key exchange procedure is surely possible but hardly applicable.) Public keys and therefore asymmetric encryption systems must be used for “middle” MIX’s.

Thus messages to and therefore messages from “middle” MIX can only be concealed in a way that is guaranteed by perfect computational secure encryption systems. The anonymity that is achievable in this way is only of a computational secure nature.

After basic deliberations concerning possibilities and limitations of changing the encoding, concrete schemes of re-encryption can be derived systematically. At first this will be done by generalizing the three re-encryption schemes described in [Chau\_81] where they are described in a simple way and not discussed in detail according to their limits and necessity.

#### 5.4.6.2 Sender anonymity

One is looking for a scheme of changing the encoding where the sender is permitted to send a message to a receiver while being completely anonymous against the receiver and all (instead of the first) MIXes. Additionally, the relations of communication are kept in secret as long as not all MIX’s that are passed through by the message do work together or all other senders and receivers of messages do work together in the same time interval. According to that, the sender is not able to use any secret key with the receiver and the

MIX against whom he wants to be anonymous. Thus an asymmetric encryption system must be used and the sender must know the encryption key.

With the keys that are suitable for encryption the sender is able to encrypt the message directly (*direct scheme of re-encryption in order to guarantee anonymity for the sender*) or the sender encrypts a key that specifies another re-encryption of a message (*indirect scheme of re-encryption in order to guarantee anonymity for the sender*, this is described below in the general terms)

**Direct re-encryption for the sender anonymity:**

The sender of a message encrypts the message with public known encryption keys of an asymmetric encryption system (resp.: for the first MIX in the order of the MIX that must be passed through if necessary with a secret key of a symmetric encryption system), so that it must be decrypted in the order of MIX's chosen by him with their dedicated and secret decryption keys (resp.: if necessary with the agreed secret key with the first MIX). Thereby the appearance of the message changes with every step of the way, i.e., its path can not be followed (unless all MIX it passes through work together or the sender is cooperating or all other senders and receivers of messages work together in the same time interval (more exactly: in the same batch)).

May  $A_1, \dots, A_n$  be the sequence of the addresses and  $c_1, \dots, c_n$  the sequence of the publicly known cipherkeys (or encryption keys) of the MIX sequence  $MIX_1, \dots, MIX_n$  that was chosen by the sender, whereby  $c_1$  can also be a secret key of a symmetric encryption system.  $A_{n+1}$  may be the address of the receiver who is called for simplification  $MIX_{n+1}$ , and  $c_{n+1}$  is his cipherkey.  $z_1, \dots, z_n$  may be a sequence of random bit strings. If  $c_1$  is a secret key of a symmetric encryption system then  $z_1$  can be an empty bit string. If  $c_i$  is an encryption key of an asymmetric encryption system that encrypts indeterministically then  $z_i$  can be an empty bit string, too. The sender creates messages  $N_i$  that will be received by  $MIX_i$ , on the basis of the message  $N_1$ , which the receiver ( $MIX_{n+1}$ ) is supposed to receive:

$$\begin{aligned} N_{n+1} &= c_{n+1}(N) \\ N_i &= c_i(z_i, A_{i+1}, N_{i+1}) \quad (\text{for } i = n, \dots, 1) \end{aligned}$$

The sender sends  $N_1$  to  $MIX_1$ . After the decryption each MIX receives the address of the next MIX and the message that is dedicated for the next MIX.

The additional encryption of random bit strings is necessary for the application of asymmetric encryption systems that are deterministic because an attacker would be able to guess and test not only short standard messages with the publicly known cipher key but also the whole output of the MIX (completely without guessing).

With this direct scheme of re-encryption in order to guarantee anonymity for the sender, the re-encryption of a message (at least with all except the first MIX) is completely

determined by each public key. In order to disallow any correlation between frequencies of in- and output messages that are proceeded by MIXes, as long as each MIX possesses a key pair, it processes different incoming messages with different corresponding secret key. If a MIX receives a message several times during a time interval in order to decode it with its secret cipher key, it will ignore it.

The direct re-encryption scheme that guarantees the anonymity of the sender which was explained before is plotted in the picture 5.24 by means of two MIX. Though the addresses were edited out and the indexes are only for distinction between messages and random bit strings, because a double index would have become necessary by using the convention of indexes of the recursive scheme of creation.

There is a larger example ( $n = 5$ ) shown in the picture 5.25. It emphasizes who knows which key and in which order messages are encoded, transferred, and decoded.

#### 5.4.6.3 Receiver's anonymity

Besides the possibility to be able to *send* a message anonymously it is also necessary to be able to *receive* a message anonymously in order to keep the communication relations a secret. This is not merely achieved by the scheme of re-encryption just described before in the sense that a sender does not know the message  $N_{n+1}$  and therefore he is able to identify a receiver in a usual network as long as

1. the sender himself or someone who collaborates with him is able to *intercept* the corresponding line or he has to cooperate with an operator or with a producer of the communication network which underlines the MIX network,
2. the address  $A_{n+1}$  is a *public or explicit address* which evinces a reference of persons or which scheme of creation (oriented on the topology of the relations of communication) is known in general resp. which is able to be exploited or the “location” of this address can be disclosed to a creator of the communication network or
3. in addition to the 1st and 2nd aspects, the sender works in *cooperation with  $MIX_n$*  if he is able to choose it (which is probable).

This problem can be solved

- considering the first threat by encrypting the connection virtually between  $MIX_n$  and receiver [Pfi1\_85, page 12, 14]
- considering the second threat by using implicit and invisible addresses (for example  $A_{n+1}$  could be an agreed implicit and invisible address between  $MIX_n$  and receiver and could then be replaced with an explicit and public address by  $MIX_n$ )
- considering all threats with the procedure (described in §5.4.1) for protecting the receiver by using implicit addressing and distribution.

$c_j$  is encryption key and  $d_j$  is decryption key of MIX  $j$ .  
 $z_i$  are random bit sequences,  $N_i$  are messages.

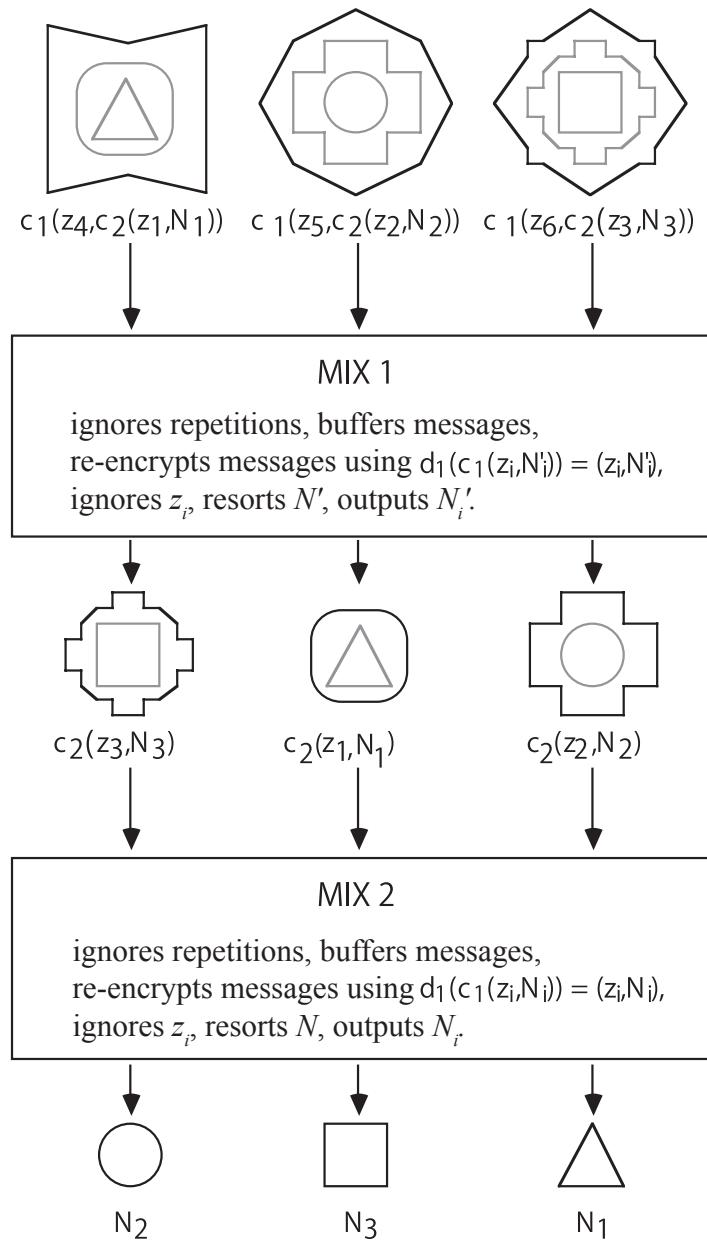


Figure 5.24: MIXes hide the connections between incoming and outgoing messages

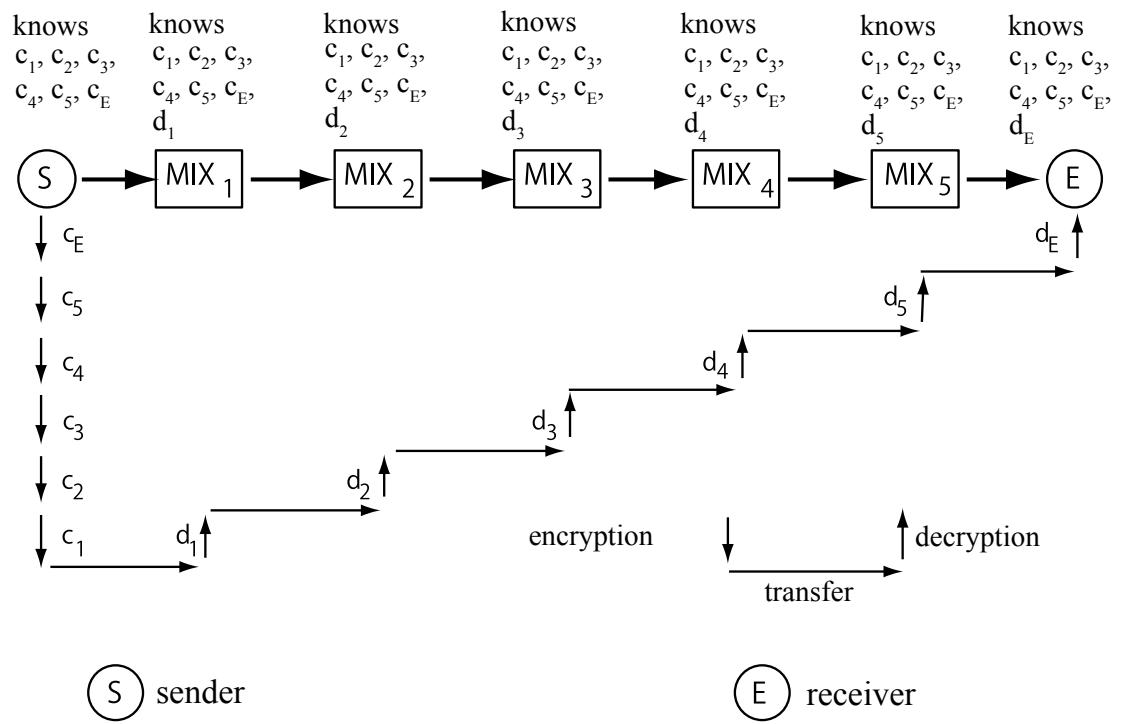


Figure 5.25: Transmission and encryption structure of messages in a MIX network, using a direct scheme of re-encryption for sender anonymity

The latter can protect even more than the relations of communication. Depending on the structure and productivity of the underlying communication network the distribution of “last” (i.e., directed straightly to the sender) messages can reduce the productivity of the communication network in an unacceptably way. Thus it is searched for less debiting kind of protection of the receiver according to his relation of communication to the sender. This kind of protection is not supposed to protect a sender from a receiver because it could be combined with a scheme of re-encryption in order to guarantee anonymity for the sender: a sender sends its message to any instance  $X$  by using the re-encryption scheme in order to guarantee anonymity for a sender, for example to any  $MIX$  with the request to send it to a receiver by using the scheme of protecting a receiver from a sender. For the protection of the communication relation between a sender and a receiver (more exactly: their *mutual anonymity*) it is now absolutely not critical that the sender is able to observe the receiving of his message by  $X$  and that the receiver is able to observe the sending of his message by  $X$ . Having the instance  $X$  having fulfilled its purpose – the most simple way possible to explain the principle – it is removed now: The concatenation of the scheme of re-encryption can be modified easily just in that way that the last  $MIX$  of the scheme for anonymity of a sender sends the message (which was sent by  $X$  up to now) directly to the first  $MIX$  of the scheme for anonymity of a receiver.

If one ignores the protection of the sender from the receiver regarding communication relations (similarly as it was discussed concerning the re-encoding schema for sender anonymity), then we are looking for a re-encryption scheme which allows a receiver to receive messages from a sender in a way that he holds anonymity against himself and all  $MIX$ es (except of the last one) as well as hiding the relations of communication as long as not all  $MIX$ es that are passed through by the message or all other senders and receivers of messages do not collaborate at the same time. In order to keep the anonymity against the sender, the receiver has to specify re-encryption of the  $MIX$ es that will process the message – but it has to be done differently from in the re-encryption scheme for the sender anonymity. Talking about a message created by the sender, in order to continue the re-encryption every  $MIX$  should possess keys that were applied before it in the sequence of  $MIX$ es. In accordance with what was already said, a receiver is not able to exchange the secret keys with a sender and  $MIX$ es statically if he wants to keep his anonymity against them. This delivering of the key is often done by appending a second part to a message created by a sender – the so-called **return address part**. This return address part is part of an *untraceable return address* that was created by the receiver [Chau\_81]. The return address must be decrypted by each  $MIX$  using its decryption key which is kept in secret and which corresponds to the publicly known encryption key (when needed, one can also exchange a key of a symmetric encryption system between the last  $MIX$  and the receiver) in order to known not only the return address to the next  $MIX$  but also the key he is supposed to use for the encoding of the message that was created by the sender. Thus it follows:

A scheme of re-encryption for receiver anonymity must be an indirect scheme of re-encryption.

**Indirect re-encryption for receiver anonymity:** let  $A_1, \dots, A_m$  be the sequence of the addresses and may  $c_1, \dots, c_m$  be the sequence of the publicly known cipherkeys of the sequence of MIXes  $MIX_1, \dots, MIX_m$  which was chosen by the receiver, whereby  $c_m$  can also be a secret key of a symmetric encryption system. The message that is accompanied by an anonymous return address will pass MIXes in the ascending order according to their indexes. May  $A_{m+1}$  be the address of the receiver who is called  $MIX_{m+1}$  for simplicity. Similarly, the sender will be called  $MIX_0$  for simplicity. The receiver creates an anonymous **return address**  $(k_0, A_1, R_1)$  where  $k_0$  is specially created for that purpose symmetric encryption key (it is also possible to use asymmetric encryption system but it would be more expensive in the sense of computation).  $MIX_0$  is supposed to use this key to encrypt the message, so  $MIX_1$  cannot read it.  $R_1$  is a part of the anonymous return address that is encrypted using  $k_0$  and transmitted with the message.  $R_1$  is created by a receiver using randomly chosen unique name  $e$  of the return address according to the following recursive scheme, where  $R_j$  refers to the return address that MIX will have, and  $k_j$  refers to a key of a symmetric encryption system (it is also possible to use asymmetric encryption system but it would be more expensive) using what a MIX should encrypt the part of the message that contains its content.

$$\begin{aligned} R_{m+1} &= e \\ R_j &= c_j(k_j, A_{j+1}, R_{j+1}) \end{aligned} \quad (\text{for } j = 2, \dots, m+1)$$

These return address parts  $R_j$  and the message's content  $I$  generated by the sender (if necessary already several times encoded), called *message's content part*  $I_j$  constitute messages  $N_j$ . These messages  $N_j$  are created by  $MIX_{j-1}$  and sent to  $MIX_j$  – according to the following recursive scheme they are created and sent by the sender  $MIX_0$  and then they pass through MIXes in a sequence  $MIX_1, \dots, MIX_m$ :

$$\begin{aligned} N_1 &= R_1, I_1; & I_1 &= k_0(I) \\ N_j &= R_j, I_j; & I_j &= k_{j-1}(I_{j-1}) \end{aligned} \quad (\text{for } j = 2, \dots, m+1)$$

The receiver  $MIX_{m+1}$  receives  $e, N_{m+1} = e, k_m(\dots k_1(k_0(I))\dots)$  and can decode and obtain the message  $I$  without any problems since he can assign all secret keys  $k_j$  (in case of an asymmetric encryption system: all decryption keys) to the unique name  $e$  of the return address part in right order.

The additional encryption of random bit strings using deterministic encryption systems is not necessary because the encrypted parts of messages  $R_j$  that were encrypted with publicly known encryption keys and with not published  $k_i$  of the MIXes, contain information the attacker does not know. Above it, when re-encrypting with keys that are not known by an attacker no tests are possible anyway.

In this indirect scheme of re-encryption for receiver anonymity only re-encryption of the part that contains the return address (at least with all except the last MIX) is decided

by each publicly known cipher key. In order to not let the frequency of the incoming and out-coming messages of MIXes reveal any connections, *each MIX processes (as long as it possesses a key pair) only different parts that contain return addresses*. That is why:

Each untraceable return address part can be used only once.

A repetition of a message's content part  $I_j$  is noncritical as long as a different key for re-encryption always is used. In a case when each anonymous return address is used only once, it is always a case that every time when a new anonymous return address is created a new key  $k_j$  is used. If a receiver does not pay attention to that while creating anonymous return addresses he only hurts himself. If  $MIX_j$  uses an old key  $k_j$  to create a return address then it is not able to harm the generator of  $k_j$  more than revealing the context of the messages which had to be concealed by re-encryption with  $k_j$ . Fig. 5.29 shows this scheme graphically.

#### 5.4.6.4 Mutual anonymity

If this re-encryption scheme that guarantees receiver anonymity is used alone, the relations of communication are completely concealed from non-involved participants (as in the re-encryption schema for sender anonymity). But since the receiver knows the return address  $R_1$ , he is able to observe the sender so long as

1. a receiver or somebody who cooperates with him is able to *wiretap the corresponding line* or a receiver cooperates with a provider or a manufacturer of the underlying communication network of the MIX or much easier in many cases,
2. a receiver who cooperates with  $MIX_1$  (which is very probable if he is able to choose it) and who assigns *public or explicit* addresses of senders to messages that were sent to the underlying communication network of the MIX (and who notifies users of the network – like it is planned in ISDN) that are evincing a reference to persons or whose scheme of creation (mostly oriented on the topology of the communication network) is either publicly known resp.: which can be developed easily or the sender is informed by a carrier or a producer of the communication network about the “place” that is associated to the address, or
3. in addition to the 1st point, a receiver who cooperates with  $MIX_1$ , which is – as already mentioned – very probable if he is able to select it.

This problem can be solved regarding to the first threat by virtual link encryption [Pf1.85, page 12, 14], regarding to the second threat by a suitable configuration of the protocol in the underlying communication network of the MIX network, and regarding to all other threats described in §5.4.4 by a procedure for protection of the sender. The latter protects even more than communication relations. Depending on structure and productivity of the underlying communication network these methods can reduce its performance in an unacceptable way which means that – as long as an application requires mutual anonymity – only the already described combination of a re-encryption

scheme for sender anonymity and receivers anonymity is possible [Pfi1\_85, page 14f]. (This idea is probably contained in [Chau\_81, page 85] as well, but the description in its universality is wrong). This mutual anonymity must already be provided with the first message and it is achieved by the former described indirection step by using an (actually necessary) additional instance  $X$  (at least functionality that can be achieved by an instance that is already available), here a directory of addresses with anonymous return addresses. Since each anonymous return address can be used only once, the usage of printed directories is very inconvenient. In contrast to that, an electronic directory that emits each address only once does not cause any problem.

#### 5.4.6.5 Re-encryption maintaining length of messages

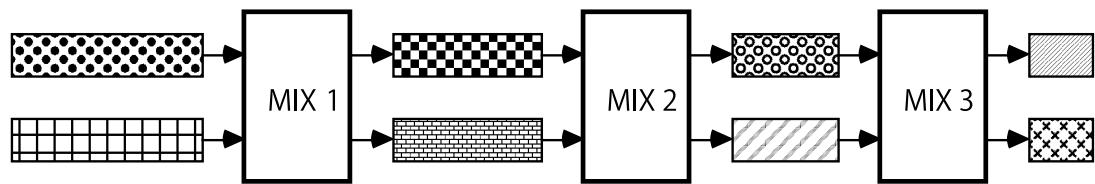
After we derived the easiest possible schemes of re-encryption to guarantee anonymity for both sender and receiver and described their field of application, we should show another property which they have in common: the length of the message is changed in both re-encryption schemes: messages become shorter. This is no problem if all messages pass through the same MIXes in same order which is necessary in order to achieve maximal possible anonymity (like it is shown in §5.4.6.1), cp. above Figure 5.26. The only reasons that can prevent one from re-encrypting all the messages at all MIXes are high computational expenses and performance requirements as was shown in [Pfit\_89, §3.2.2.4]. In such MIX networks reduced length of messages could provide valuable information for attackers. It is easy to see that in the bottom part of Fig. 5.26 messages do not cross within the MIX's which can not be seen in the upper part of the figure.

Therefore it is desirable to use a re-encryption scheme that maintains the length of the messages so that nobody is able to know (except for the original author of the message) while looking at the message how many times its has been re-encrypted or still has to be re-encrypted. Moreover, all messages should have the same message length – as it was shown in the beginning of this section – and should not be distinguishable because of the applied re-encryption scheme. Thus is required a *universal re-encryption scheme that maintains a message's length*. This must inevitably be an indirect re-encryption scheme as it was said in the derivation of a re-encryption scheme for receiver anonymity.

##### **Indirect re-encryption scheme that maintains a message's length:**

There is a new notations we use additionally to the notations introduced in the section about re-encryption schema for receiver anonymity: squared brackets are used for block limits. Starting with the sender (also called  $MIX_0$ ) the MIX  $MIX_1, \dots, MIX_m$  shall be passed through in sequence and finally the receiver (also called  $MIX_{m+1}$ ) shall be reached. The receiver does not need to know that he is the receiver when receiving the first message. He processes it the same way as previous MIXes. Each message  $M_j$  consists of  $b$  blocks of same length which are used by the applied asymmetric encryption system that is created by  $MIX_{j-1}$ . The first  $(m+2-j)$  blocks of  $N_j$  contain the (return-) address part  $R_j$ , the last  $(b-m-1)$  blocks contain the content of a message. Each  $MIX_j$  decrypts the first block of the message  $M_j$  using its secret decryption key  $d_j$ . The result of decryption is a key  $k_j$  of a symmetric encryption system that uses the block

equal MIX order: shrinking message lengths are no problem



different MIX orders: shrinking message lengths are problematic

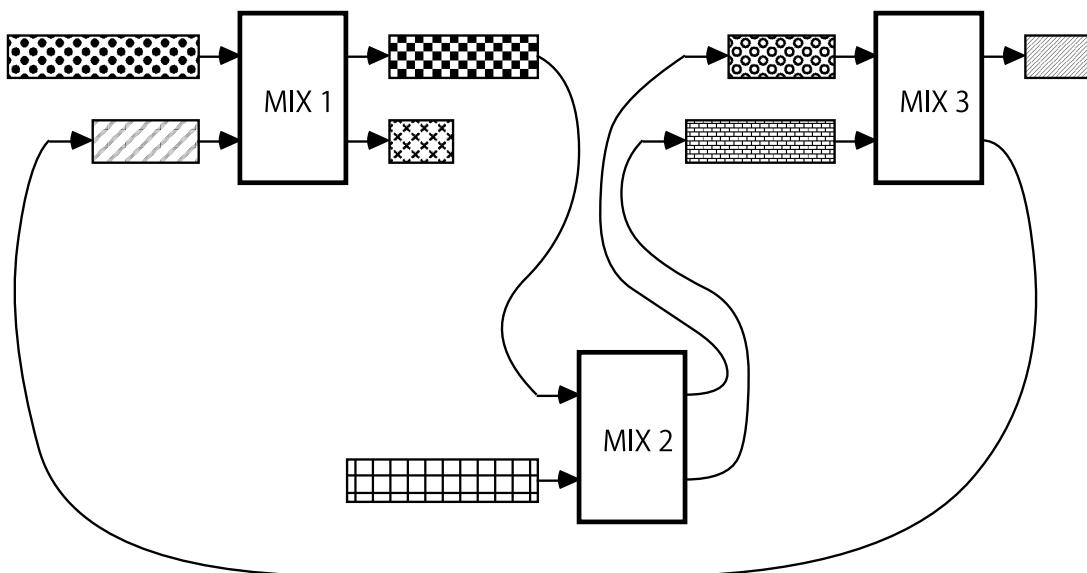


Figure 5.26: Reducing message length during re-encryption

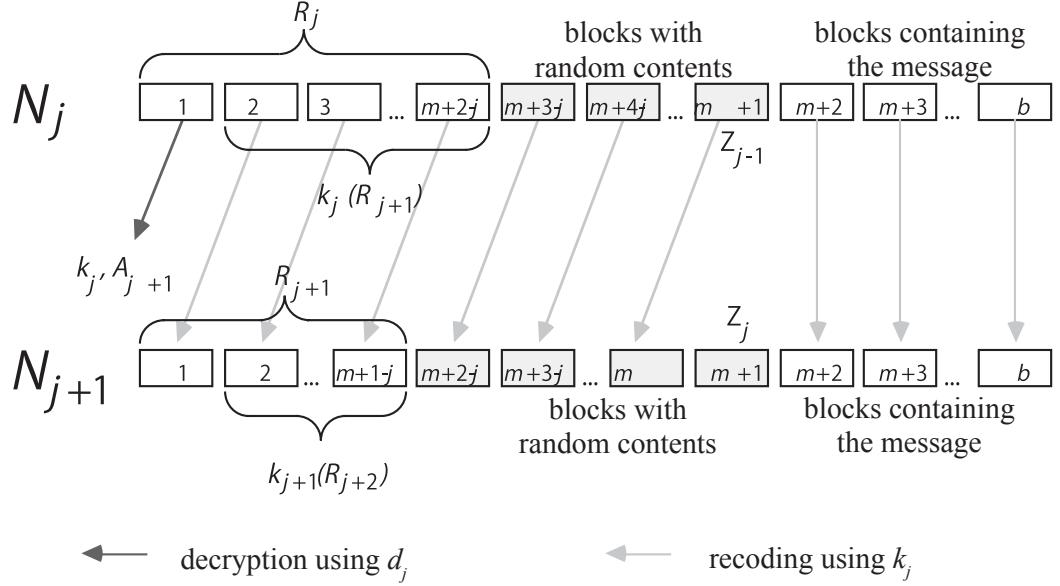


Figure 5.27: Indirect length preserving recoding scheme

length for re-encryption of the remaining message (it is also possible to use asymmetric encryption system but it would be more expensive). It will also find the address  $A_{j+1}$  of the next MIX (or receiver). The receiver finds his own address as the address of the next MIX (or receiver) and realizes that it is the receiver of the message. This first blocks of  $N_j$  are not needed by the following MIX (and also the receiver), that is why they are discarded by  $MIX_j$ .  $MIX_j$  re-encrypts the remaining  $(b - 1)$  blocks using the key  $k_j$  and appends a block with random content  $Z_j$  in front of the first block with message content in order not to change the message length (in [Chau.81, page 87] this block is also re-encrypted by using the key  $k_j$  which does not make much sense since the block has random content anyway).

To make it work, the return address  $R_1$  used by a sender (based on a randomly chosen unique name  $e$ ) is created using the following recursion scheme:

$$R_{m+1} = [e]$$

$$R_j = [c_j(k_j, A_{j+1}), k_j(R_{j+1})] \text{ for } j = m, \dots, 1$$

The creation scheme of the messages  $N_j$  and especially maintaining the length of messages in this recursive re-encryption scheme is shown in Figure 5.27. Since  $MIX_j$  does not need to know its index  $j$ , it does not need to differentiate between blocks of the (return) address part and blocks containing random content. In reference to Figure 5.27, this means that the  $MIX_j$  merely knows that block  $m + 1$  of the message  $N_{j+1}$  is a block containing random content (it is gray on the figure).

All MIXes have to know  $m$  (in order to know where to insert their random content block), so the MIXes know both  $m$  as the upper bound for the number

of MIXes that are passed through and  $b - m - 1$  as the upper bound for the length (given in blocks) of the messages content.

From the creation scheme of the return address part it follows that all blocks (except the first of  $MIX_j$ ) of the (return-) address part are *decrypted* with  $k_j$ . Whether the last  $b - m - 1$  blocks that contain messages' content are supposed to be *en-* or *decrypted* with  $k_j$  depends on the re-encryption executed by  $MIX_j$  and if it provides anonymity for a sender or a receiver. If re-encryption using  $k_j$  provides sender anonymity,  $MIX_j$  has to decrypt because otherwise the receiver has to know the key  $k_j$  – then the re-encryption as protection against him will be useless – or he would not be able to understand the content of the message. Similarly,  $MIX_j$  has to encrypt if the re-encryption should provide receiver anonymity. Instead of adding redundant bits as in the recursive scheme of creating the (return-)address parts mentioned above (which tells MIXes if they should en- or decrypt the blocks with message content) this information is considered as part of the key  $k_j$ .

So that  $MIX_j$  is not required to know its index  $j$ , all of the blocks which have random content should be processed the same way as all blocks (except the first one) of the (return-) address  $R_j$ . They shall be decrypted. This is represented formally by joining the (return-) address part  $R_j$  and the blocks with random content into a message header  $H_j$  (header). In the following  $[H_j]_{x,y}$  stands for the blocks from  $x$  to  $y$  (including both) of  $H_j$ . If  $x > y$  then  $[H_j]_{x,y}$  does not stand for a block.

$$\begin{aligned} H_1 &= R_1 \\ &= [c_1(k_1, A_2)], k_1(R_2) \\ H_j &= K_{j-1}^{-1}([H_{j-1}]_{2,m+1}), [Z_j] \\ &= [c_j(k_j, A_{j+1})], k_j(R_{j+1}), K_{j-1}^{-1}([H_{j-1}]_{m+3-j,m+1}), [Z_j] \text{ for } j = m, \dots, 2 \end{aligned}$$

If the indirect scheme of re-encryption for **sender's anonymity** that maintains the length of a message, then a sender has to generate keys  $k_1, k_2, \dots, k_m$  and has to know the decryption key  $c_{m+1}$  and the address  $A_{m+1}$  of a receiver.

To make a receiver able to understand a message, a sender has to encrypt it with keys generated by him before dispatching the message. Thus the message  $N_1$  is created by the sender according to the following scheme based on the message content  $I = [I_1], \dots, [I_i]$  that is divided in  $i = b - m - 1$  blocks (if the message is too short it should be extended up to the required length) and the message header  $H_1$ :

$$\begin{aligned} N_1 &= H_1, I_1; & I_1 &= k_1(K_2(\dots(c_{m+1}(I_1))\dots)) \\ & & &= k_1(k_2(\dots k_m(c_{m+1}([I_1]))\dots)), \dots, k_1(k_2(\dots k_m(c_{m+1}([I_i]))\dots)) \\ N_j &= H_j, I_j; & I_j &= k_{j-1}^{-1}(I_{j-1}); & (\text{for } j = 2, \dots, m+1) \end{aligned}$$

The message  $N_j$  consists of

1. the (return-) address part  $R_j$  that has  $m + 2 - j$  blocks at a time,

## 5 Security in Communication Networks

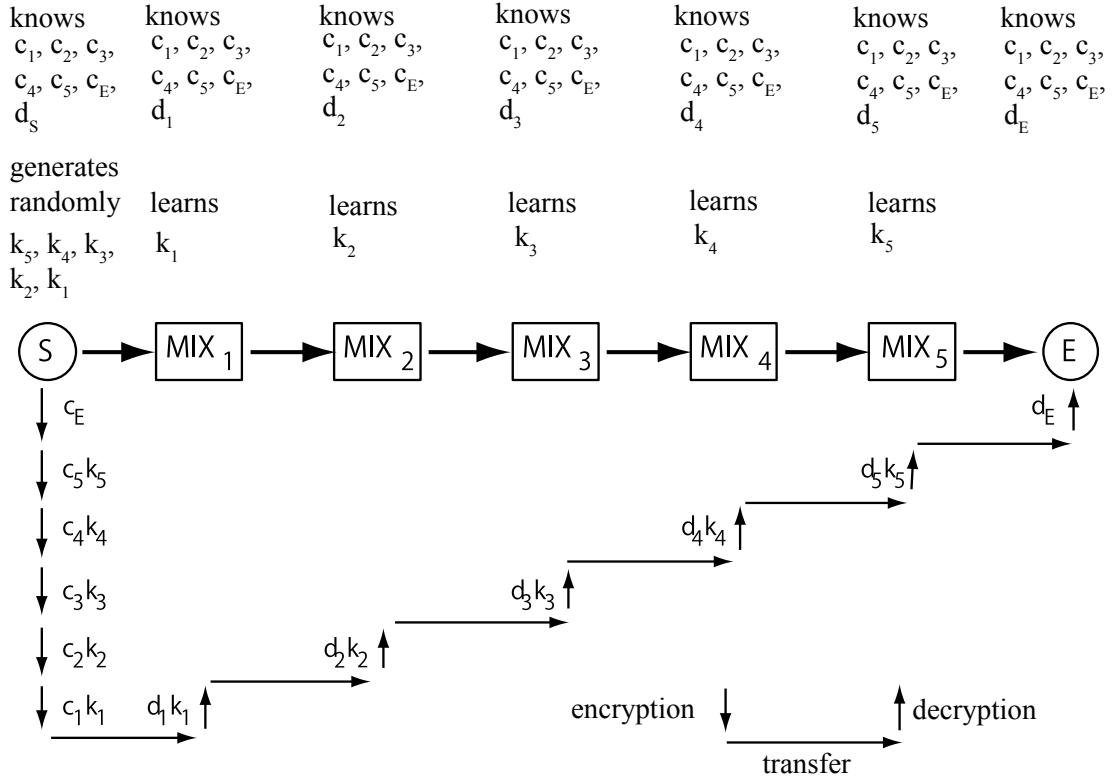


Figure 5.28: Indirect recoding scheme for sender anonymity

2.  $j - 1$  blocks of random content that could be “decrypted” several times if needed,
3. the content part of the message  $I_j$  that could be decrypted several times if needed and that was generated and encrypted with the keys  $c_{m+1}, k_m, \dots, k_1$  by the sender.

Using this message  $MIX_j$  creates a message  $N_{j+1}$  using the recursive scheme explained above. Figure 5.28 shows the use of the re-encryption scheme maintaining length of the message for sender anonymity for  $m = 5$  which is already known from Figure 5.25. If the indirect re-encryption scheme maintaining length of a message should provide **receiver** anonymity then the receiver generates keys  $k_0, k_1, \dots, k_m$  and anonymous return address  $(k_0, A_1, R_1)$  that is made known to the sender. The sender creates a message  $N_1$  using  $(k_0, A_1, R_1)$  as well as each  $MIX_j$  creates a message  $N_{j+1}$  using  $N_j$  according to the following recursive scheme:

$$\begin{aligned}
 N_1 &= H_1, I_1; & I_1 &= k_0(I) \\
 &&&= k_0([I_1]), \dots, k_0([I_i]) \\
 N_j &= H_j, I_j; & I_j &= k_{j-1}(I_{j-1}) & (\text{for } j = 2, \dots, m+1)
 \end{aligned}$$

After receipt of  $N_{m+1}$  the receivers recognize the return address by the unique name

$u$  and can decrypt the message and receive the plaintext using known and appropriate keys  $k_m, k_{m-1}, \dots, k_0$ .

It is shown in Figure 5.29 for the example  $m = 5$  using the graphical representation of Figure 5.25. It is not shown in detail how the receiver  $R$  sends the return address to the sender  $S$ . Numbers above and under the arrows specify the order of the steps.

If **a sender and a receiver** should stay anonymous towards each other in a verifiable way while using this indirect re-encryption scheme maintaining length of a message then the blocks of the (return-) address from  $m$  to a certain index  $g$  ( $1 < g \leq m$ ) have to be created by the sender and the rest  $g - 1$  blocks have to be created by the receiver. That is how the sender receives  $R_g$  and creates from this  $m + 2 - g$  blocks the return address  $R_1$  that consists of  $m + 1$  blocks. He encrypts the message content with the key  $k_s$  (he received it as a part of the return address) and with the  $g - 1$  keys that he had generated. After receiving the message content that was encrypted several times, the receiver decrypts it with the keys  $k_g, \dots, k_m, k_S$ .

Figure 5.30 explains this for  $m = 5$  and  $g = 4$  using the graphical notation of Figure 5.25. The numbers near the arrows specify the sequence of the steps as on Figure 5.29.

In order to save transmission and time effort the functionality of the last MIX that provides sender's anonymity and the functionality of the first MIX that provides receiver anonymity can be provided by one MIX (in the example on Figure 5.30 the functionality of MIXes  $MIX_3$  and  $MIX_4$ ). It makes sense when the sender and the receiver have no special trust preferences for individual MIXes.

If messages between mutually anonymous partners have return addresses by mistake it should be emphasized that in any case all of  $i$  blocks with the message content should not be available to extract actual message content:  $m + 2 - g$  blocks are required for the return address – it is so for each message (even if different  $g$  is chosen for each message), because exactly like in the indirect scheme for re-encryption for receiver anonymity *each MIX (as long as he keeps his key pair) is allowed to work only on different address parts in the indirect re-encryption scheme maintaining the length of a message*.

The same is valid for re-encryption of blocks which belong to the *same* message and re-encrypted with the same key: if the used encryption system is a block cipher it is not allowed for two blocks to be the same. The same is also valid for self synchronizing stream ciphers, cp. §3. If a synchronous stream cipher is used a MIX merely needs to take actions regarding it. What was said before is in turn valid for each indirect re-encryption scheme and it is especially emphasized when we explicitly talk about “blocks”. This **active attack by repetition of indirect re-encrypted message parts** was missed in [Chau\_81, Pf1\_85, page 14, 27].

Furthermore, it should be once again emphasized that re-encryption of the (return-)address part which is specified by the keys from  $c_2$  to  $c_{m-1}$  is only computationally secure. The anonymity that is achieved by MIXes from  $MIX_2$  to  $MIX_{m-1}$  for communication relations is also only computationally secure. Re-encryptions that are specified by  $c_1$  to  $c_{m-1}$  can be information-theoretically secure if sender or receiver exchanged secret keys with  $MIX_1$  resp.  $MIX_m$ . As long as a sender or a receiver can choose the order of the MIXes freely they should exchange the secret key with the MIX they trust

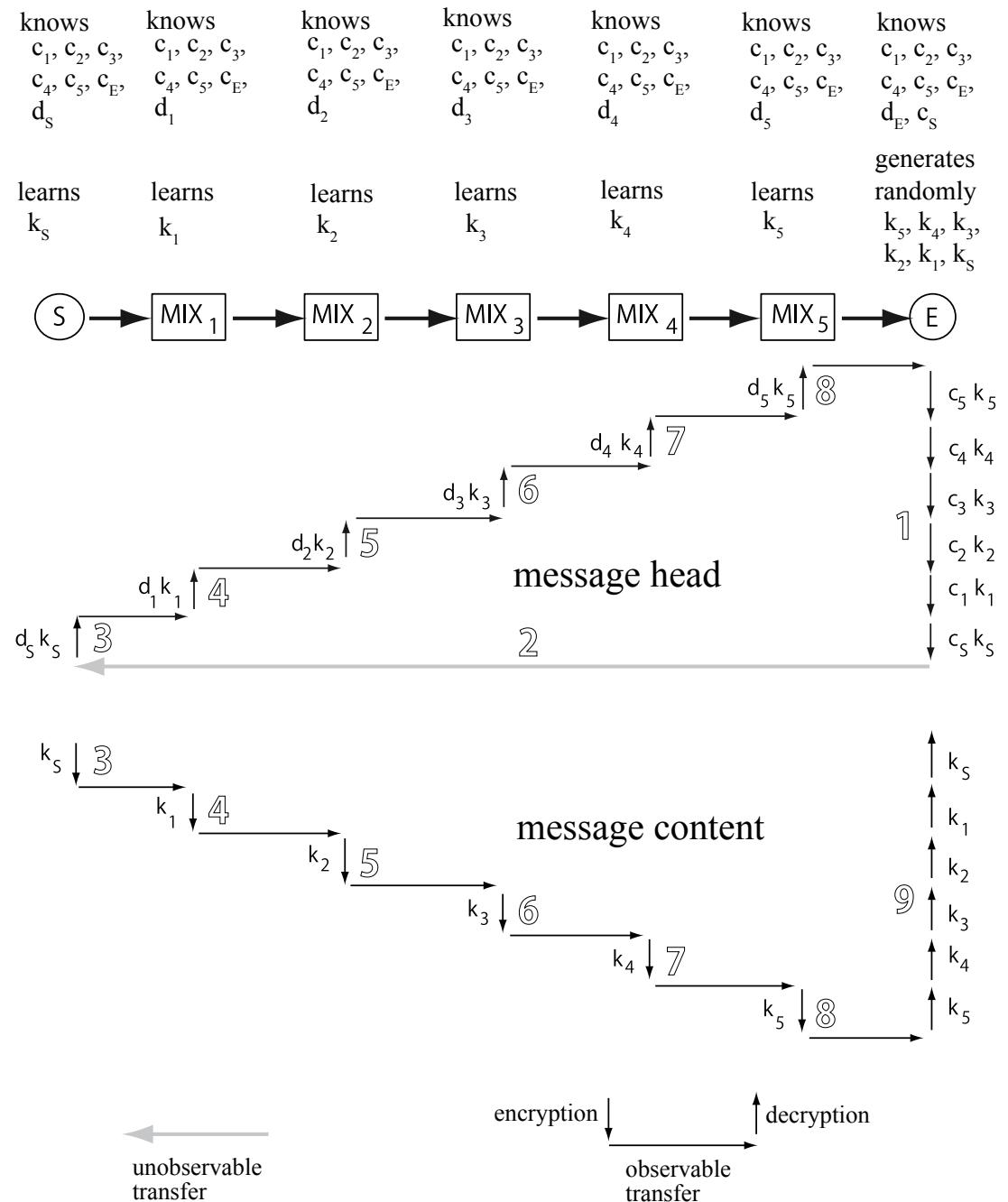


Figure 5.29: Indirect recoding scheme for receiver anonymity

#### 5.4 Basic measures settled inside the communication network

knows $c_1, c_2, c_3,$ $c_4, c_5, c_E,$ $d_s$	knows $c_1, c_2, c_3,$ $c_4, c_5, c_E,$ $d_1$	knows $c_1, c_2, c_3,$ $c_4, c_5, c_E,$ $d_2$	knows $c_1, c_2, c_3,$ $c_4, c_5, c_E,$ $d_3$	knows $c_1, c_2, c_3,$ $c_4, c_5, c_E,$ $d_4$	knows $c_1, c_2, c_3,$ $c_4, c_5, c_E,$ $d_5$	knows $c_1, c_2, c_3,$ $c_4, c_5, c_E,$ $d_E, c_S$
--	--	--	--	--	--	---

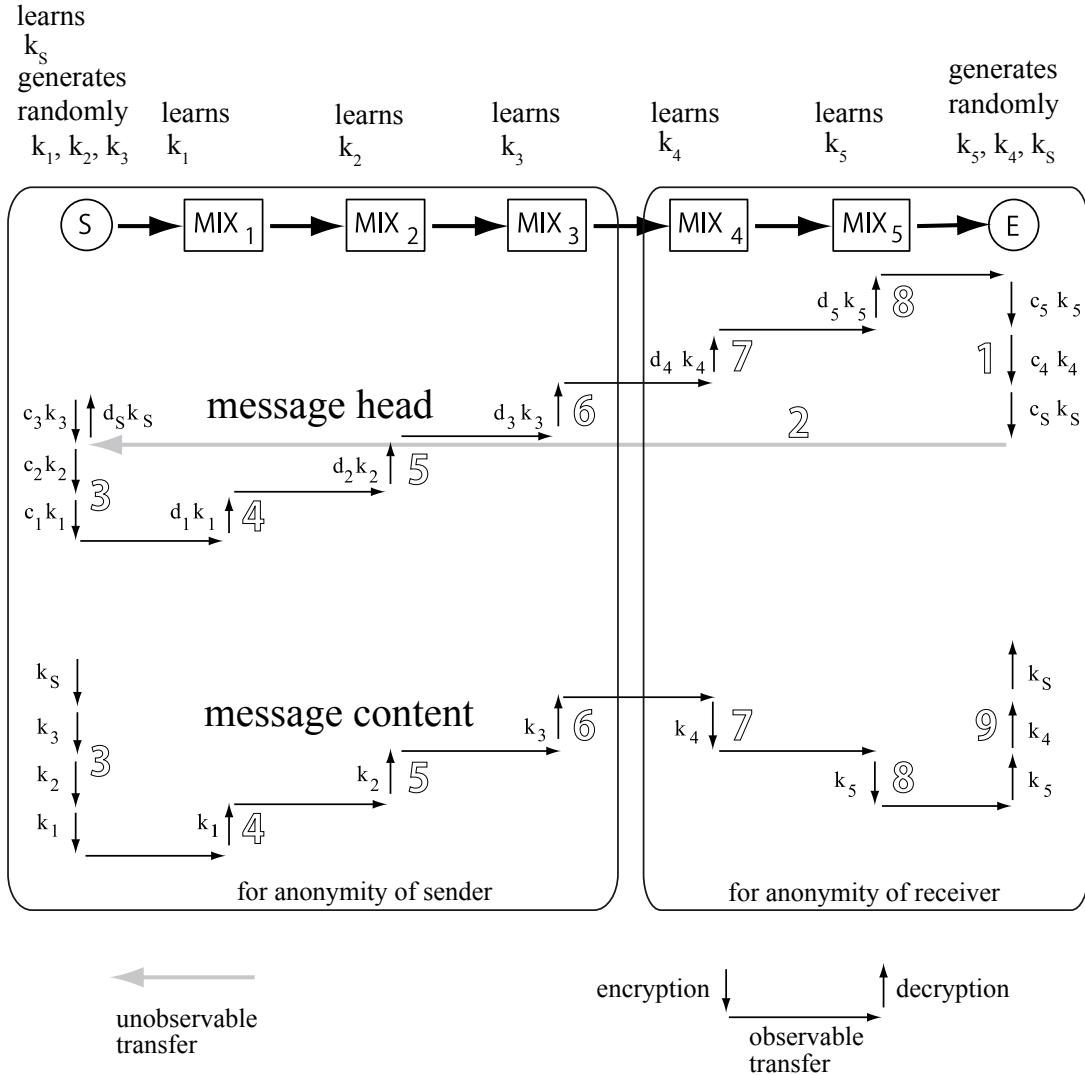


Figure 5.30: Indirect recoding scheme for sender and receiver anonymity

most. As it was explained before, the free choice of the order of MIXes excludes the achievement of the maximal aim of this procedure of the MIX that re-encrypts.

If a symmetric encryption system has a property  $k^{-1}(k(x)) = x$  additionally to what is postulated in §3 for all keys  $k$  and plain texts  $x$  **the property**  $k(k^{-1}(x)) = x$ , i.e.: if

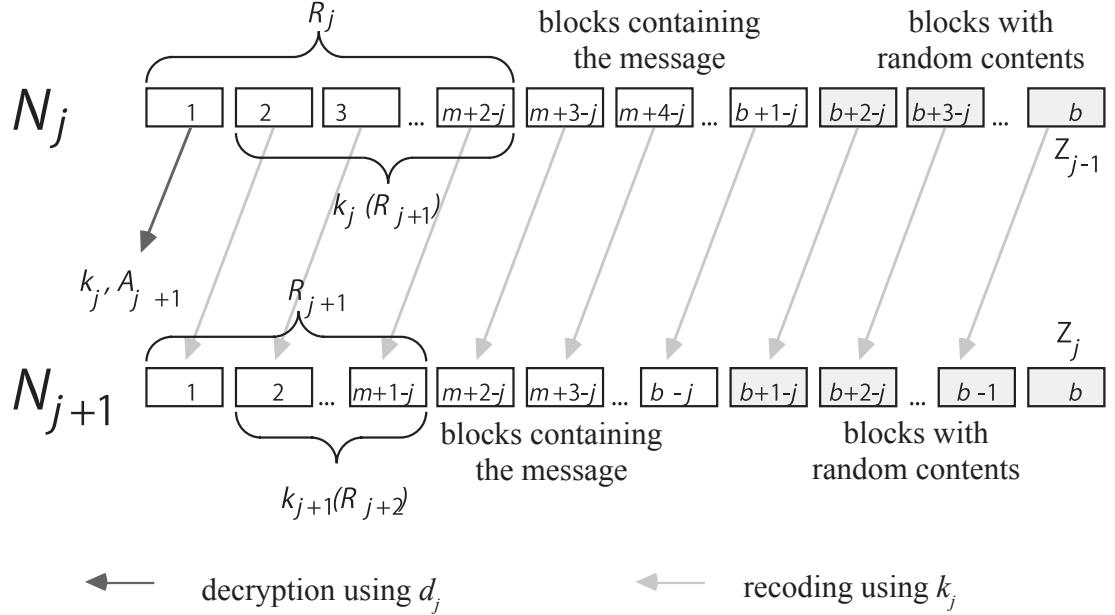


Figure 5.31: Indirect length preserving recoding scheme for special symmetrical encryption systems.

not only en- and decryption but also de- and encryption are reverse to each other, then it is not necessary to decide which re-encryption scheme that maintains message length to use (for sender or receiver anonymity) as will be shown further. Furthermore, it is possible to withhold the knowledge of the following things for the MIX:

- $m$  as the upper bound of the number of MIXes that have to be passed through, as well as
- $b - m - 1$  as the upper bound for the length (in blocks) of a message content.

Instead of the  $m + 1$ th block each MIX attaches his block with random content as last block. The receiver will receive the message content in the blocks beginning at the second block instead of in the last blocks, cp. Figure 5.31.

If it is decided arbitrarily that each MIX encrypts all blocks of the message, that is how one gets the re-encryption scheme maintaining the message length specified in [Chau.81]. As it was mentioned before, a redundant encryption of the attached block with random content is avoided:

$$R_{m+1} = [e] \\ R_j = [c_j(k_j, A_{j+1})], k_j^{-1}(R_{j+1}) \quad (\text{for } j = m, \dots, 1)$$

$$N_1 = R_1, I_1; \quad I_1 = k_0(I) = k_0([I_1], \dots, k_0[I_i]) \\ N_j = R_j, I_j; \quad I_j = k_{j-1}(I_{j-1}), [Z_{j-1}] \quad (\text{for } j = 2, \dots, m + 1)$$

If the message content  $I$  is plaintext the receiver is only able to understand it when he knows the keys  $k_0, k_1, \dots, k_{m+1}$ , thus  $(k_0, A_1, R_1)$  is, in other words, an untraceable return address and therefore the scheme of changing the encoding in order to guarantee *receivers anonymity* is used.

If he is supposed to use the re-encryption scheme for *sender anonymity*, the sender generates the keys  $k_1, k_2, \dots, k_m$  and has to know the cipher key  $c_{m+1}$  and the address  $A_{m+1}$ . In order to make the receiver able to understand the message content, the sender has to decrypt it with the keys generated by the receiver before he sends the message. Only one formula that was shown above has to be modified, namely for  $I_1$ :

$$\begin{aligned} I_1 &= k_1^{-1}(k_2^{-1}(\dots k_m^{-1}(c_{m+1}(I_1))\dots)) \\ &= k_1^{-1}(k_2^{-1}(\dots k_m^{-1}(c_{m+1}([I_1]))\dots)), \dots, k_1^{-1}(k_2^{-1}(\dots k_m^{-1}(c_{m+1}([I_i]))\dots)) \end{aligned}$$

#### 5.4.6.6 Efficient avoidance of re-encryption repeating

As it was shown in the examples before, when MIXes use re-encryption schemes they should take care that no **special re-encryption is executed more than once** (i.e., the re-encryption of same message parts, same encryption system and the same encryption key). Otherwise an attacker could send different messages containing the same message part repeatedly. Then he would observe which output–message part is repeated in the corresponding MIX–outputs with corresponding probability. Using this knowledge the attacker could be able to bridge this MIX according to the message that had contained the message part originally.

There is one unfortunately false idea one can up while trying to solve this issue which can cause a lot of storage and message-comparing effort. The idea is not to allow an attacker to have access to the messages (which also means to parts of messages) between MIXes as well as between the sender and the first MIX and the receiver and the last MIX using virtual link encryption. The mistake in this idea is that this kind of attack is only impossible for an attacker, but not for participants, in the process of a message re-encryption, especially MIXes. The first and the last MIX could disclose the communication relations together by sending a message part of a message which was sent by the first MIX repeatedly through all MIXes that lie between them (and would be completely useless in this case).

So there are only *four possibilities for reducing the effort needed for storage and message comparing*. As it is described in [Chau\_81, page 85]

1. The *publicly known decryption keys* of the MIXes *can be exchanged for new keys more often* or
2. The parts whose re-encryption is specified by the public decryption key of the appropriate MIX can contain a "timestamp", which determines precisely<sup>10</sup> at which juncture their only acceptable re-encryption will take place.

---

<sup>10</sup>It can also be useful to determine when the message was re-encrypted the last time. This makes a consideration between the keeping–small of the already–mixed–list and the flexible use of e.g., return addresses possible.

The last argument would reduce the effort almost to zero but both arguments are inapplicable in the case of anonymous return addresses if the sender is supposed to choose freely when to reply.

Therefore the possibility mentioned in [Pfi1\_85, page 74f] is important:

3. Instead of saving the image of message parts (after the definition of the asymmetric encryption systems they must have a length of at least 100 bits, but should be even longer than a hundred of bits for security reasons ) that should be re-encrypted by MIXes having a variable length, they should be saved with a constant length, e. g., 50 bits, using reducing hash functions.

This hash function should simply map the domain proportionate to its range – it is allowed (with justifiable effort) to be possible to find two similar domains which are mapped to the same image. Before a MIX applies its secret decryption key to a message part, he creates the appropriate image using the hash function and checks if it is already recorded in the already-mixed-list of MIX. If yes, he ignores the message, if no, the image is registered in the already-mixed-list and the decryption is done. The disadvantage of this procedure is a very small probability that occasionally two different message parts can be mapped to the same image and the one that arrives later is wrongly not processed. In this case the sender is supposed to retry this with another message, e. g., by changing the key or the random bit chain. To be able to do so, fault-tolerance measures are required, but they are also required because of other more important reasons (in short, they are needed anyway) and described in detail in [Pfit\_89, §5.3].

Another very similar possibility is chosen for an implementation of MIXes extended for Internet (MIXMaster):

4. There is a certain *part* of the message saved (and always compared). This can be a part of the message that should be or is already re-encrypted. In the implementation of the MIXMaster, for example, the random bit chains are chosen to be such part (cp. §5.4.6.2)

In contrast to the third solution, the fourth saves the effort of applying a hash-function, but assumes for the storage efficiency that a message part with high randomness (in technical terminology: entropy) is chosen. Obviously, random bit chains have this property, that is why the MIXMaster-implementation is done very well.

By means of the procedure of **anonymous polling**, that is described in the following, the 1st and the 2nd possibility for reducing storage and message-comparison efforts of the MIXes can be also applied for the protection of the receiver: instead of an anonymous return address (with long validity) only a *classification number* is disclosed to the partner. The partner sends his answer which is addressed with the classification number encrypted in the end-to-end manner using sender anonymity scheme to a non-anonymous instance  $X$  by an anonymous call of intermediate storage. Once in a while a participant station of the receiver sends an untraceable return address to  $X$  using a sender anonymity scheme and the participant station receives an answer within a determined batch order if it has already arrived. Although more messages have to be sent obviously using the anonymous demands the advantage that

- a) it is exactly known before who will perform which change of encoding in which batch and
- b) therefore both the first and the second possibility are applicable,

is supposed to reduce the memory-and message-comparison-effort so much that nevertheless the effort is reduced at all.

The procedure of the anonymous demands is of relevance where distribution for protection of the receiver is not applicable because of narrow band participant-connection-conductions, cp. §5.5 .

#### 5.4.6.7 Short preview

If everything that was said before is done properly, the worst consequence of a **modifying attack** (or errors) against a MIX is the loss of a message but not the anonymity of the communication relations. A loss of a message can be detected and proved by public access to messages that were sent by a MIX or in case of virtual encryption of the connections that were signed with a MIX-specified signing key, cp. §5.4.7. Specially designed fault-tolerance measures reduce the probability of messages loss due to MIXes failure (a failure of a single MIX in the sequence of MIXes is enough to lose a message) as it will be explained in more detail in §5.4.6.10.

Because of the time delay cause by waiting for messages, checking for repetition and changing the order of messages as well as delay for decoding in an asymmetric encryption system, the already described scheme of re-encryption is inapplicable for applications that require short transmission time.

If this scheme is modified, as introduced in [Pfi1\_85, page 25–29], anonymous channels that satisfy, for example real-time-requirements sufficient for voice communication can be provided. They will be described in more detail in §5.4.6.9. To this end a special message is transmitted using the procedure described above in order to establish the channel. This message delivers the key of a faster symmetric encryption system for each chosen MIX. This MIX will use the key for decryption of the traffic in this channel. Re-encryption in channels is useful if and only if at least two channels of same capacity are *established and destroyed simultaneously* by the same MIX. If not only the relations of communication are supposed to be protected by re-encryption but also **sending** and **receiving** of participant stations, each participant station must be a MIX or must send a lot of meaningless messages. This was described in §5.4.3 and will be also described for the boundary conditions of narrow band participant in §5.5.1.

#### 5.4.6.8 Necessary properties of the asymmetric encryption system and breaking the direct RSA-implementation

For conclusion it is pointed out explicitly that the encryption systems that are used by the (middle) MIXes have to resist not only (as each asymmetric encryption system) an adaptive active chosen-plaintext attack but also a **chosen-ciphertext attack** substantially (i.e., instead of the difficulty that the MIX do not put out the random bit

chains). One can not exchange keys after each batch when anonymous return addresses are used, that is why the asymmetric encryption system that is used has to be resistant to **adaptive chosen-ciphertext attack**.

§3 was states that so far there exists no one-step encryption system that is resistant to such an attack just under the assumption of the difficulty of a pedigreed standard problem in a provable way and would therefore be a canonical candidate for the implementation of the (middle) MIX. Since there are *multiple-step* asymmetric encryption systems that do this in a provable way, one can gain provable secure MIX implementations from them. But these are very expensive because of the multiple-steps-property between sender (the one who specifies the re-encryption) and *each* MIX that is passed through: if (middle) MIXes  $MIX_1$  to  $MIX_n$  are used, the exchange of the key pairs between sender and  $MIX_i, i > 1$  has to take place through each  $MIX_j, 1 \leq j < i$  [Bött\_89, §3.3.1]. This makes the main advantage of the MIXes to contradict procedures for data protection concerning traffic and interests, namely relatively small required bandwidth between participant station and communication network. Therefore only the case of implementations for one-step asymmetric encryption systems is explicitly treated in the rest of this book.

It was shown in §3.6.3 that RSA is not able to resist a non-adaptive chosen ciphertext attack without an appropriate redundancy rating which was proven by the decoder. It will be shown in the following that the use of **RSA** is **absolutely insecure if no additional redundancy rating and if coherent and random bit chains**(i.e., like it is described in [Chau\_81, page 84]) are used. This is shown, in order to keep the notation simple, in Figure 5.24.

The attacker observed a message  $c(z, N)$  in the entrance of the MIX. The attacker chooses an appropriate factor  $f$  and turns in the message  $c(z, N) * c(f)$  to the MIX for mixing it in the same resp. in a later batch. After the perfect (Unverkettbarkeit ???) of  $c(z, N)$  and  $c(z, N) * c(f)$  with arbitrary chosen factors  $f$  was shown in §3.6, “practical” perfect (Unverkettbarkeit ???) applies if only “a lot” of factors are possible, i.e., the MIX has no chance to recognize an active attack. It remains to show that the attacker receives something that makes him able to find out  $N$  from the output messages from the appropriate batch. This can be represented in the general notation intuitively and approximately right as follows: The MIX creates  $d(c(z, N) * c(f)) = (z, N) * f$  internal. If the attacker is able to choose  $f$  so that in the place of the output message accrues  $N * f$ , in general notation this means  $(z, N) * f = (?, N * f)$  his attack was successful: he just has to prove which two message of the resp. of the both batch(es) comply the equation  $X = Y * f$ . It is not relevant to know what accrues at the place of the random bit chains because this will not be proven or output by the MIX.

It cannot be explained why this attack succeeds without using additional details of RSA. The ones that are necessary for comprehension are that en- and decoding take place by exponentiation within a ring of residue classes. The modulus  $n = p * q$  who characterizes the ring of residue classes, whereby  $p$  and  $q$  are prime numbers of at least 500 bit length each (for security reasons), forms with the exponent, which is used for encoding, the publicly known cipher key.  $b$  bit long random bit chains and  $B$  bit long

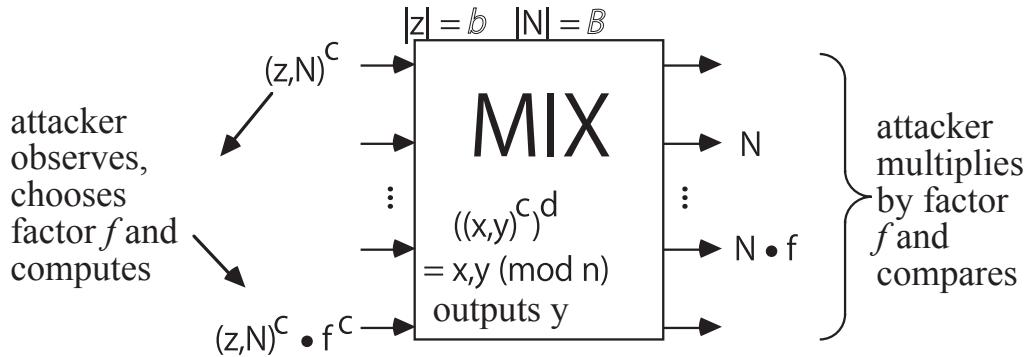


Figure 5.32: Breaking the direct RSA implementation of MIXes

messages can be put in each block of this block cipher, as long as  $2^b * 2^B < n$ . In the description of the implementation by David Chaum  $b$  is much more smaller than  $B$  (as it is in each efficient implementation). If the random bit chains are put in a bit location of higher valence then  $(z, N) = z * 2^B + N$  and  $(z, N) = (z * 2^B + N) = z * 2^B * f + N * f$  applies.

From the congruency  $(z, N) * f \equiv z' * 2^B + N'$ , which defines<sup>11</sup> the values  $z'$  and  $N'$ , follows  $z * 2^B * f + N * f \equiv z' * 2^B + N'$ , from this follows  $2^B * (z * f - z') \equiv N' - N * f$  and from this follows  $z * f - z' \equiv (N' - N * f) * (2^B)^{-1}$ . If the attacker chooses  $f \leq 2^b$  than  $-2^b < z * f - z' < 2^{2b}$ . The attacker now inserts into the formula  $z * f - z' \equiv (N' - N * f) * (2^B)^{-1}$  for  $N$  and  $N'$  all output messages of the appropriate batch(es) and proves that  $z * f < z'$  lies in the adequate interval. This is, with high probability because  $b$  is much more smaller than  $B$ , just for only one pair of the output messages the case. The first message  $N$  of this pair is the message the MIX is supposed to get bridged with an active attack. The second message  $N'$  is the output message which corresponds to the input message which was created by the attacker.

If two or more pairs are fulfilling this condition the attacker repeats his attack with another factor. In his second attempt and in all following attempts, too, he performs the pairwise inserting only with those message (of the first batch) that were not excluded by the previous tests. The attack on the relations of communication of a message  $N$  is to be executed with an effort that is quadratic to the batch-size of  $N$ : The attacker has to encode with RSA only once in each attack (as well as multiplying the result with  $N$  and reducing modulus  $n$ ) as well as performing one for each message one addition, two multiplications, two reductions modulus  $n$ , and two comparisons. It is described in [PfPf.90, page 376] how the effort of this attack can be reduced to  $O(\text{batch-size} * \text{ld(batch-size)})$ .

The assumption that  $b$  is much smaller than  $B$  is not necessary. An appropriate

<sup>11</sup>One assumes that on both sides of the defining congruency are the smallest non-negative representatives of the remainder class. That means that in reality the congruency is an equation which determines in addition with the differences that result from the length of  $z'$  and  $N'$  the values of  $z'$  and  $N'$

attack is already possible when  $B$  is so large that a choice of  $f \leq 2^{B-1}$  results in the unlinkability of  $(z, N)$  and  $(z, N) * f$  for the MIX that must be bridged. The resulting difference  $-2^b < z * f - z' < 2^{b+B-1}$  is already enough for the attacker to be able to eliminate in each step of his attack a part of messages of the original batch that are still possible. The effort for this attack increases just by a logarithmic factor.

If an adaptive attack is not possible, e. g., because the MIX changes his key pair after each batch, the attacker can obtain the same information with his iterative attack by choosing more factors, creating appropriate more messages and passing all these message and the message with which he wants to bridge the MIX in for being mixed in the same batch.

It was not proved before if a descent of the effort for the attack is possible with the attacks on RSA that were published recently.

Everything that was shown in examples using an direct scheme of changing the encoding for senders anonymity is also valid for indirect schemes by using whole message parts instead of (return-) address parts. This is possible in all schemes of changing the encoding specified in [Chau\_81] (cp. [PfPf\_90, §3.2]). But this can be avoided by using appropriate schemes of changing the encoding, i.e., such ones where no direct but indirect changed parts are used in an observable way. At this the used (symmetric) encryption system used for changing the encoding should work completely different from RSA.

If the random bit chains are positioned on the bit places of lower valence an analog and also successful attack exists [PfPf\_90, §3.2].

An analog attack fails if instead of a coherent random bit chain a lot of single random bits are “interspersed” into the message in not too huge distances and are sorted out by the MIX. A more weak attack is possible indeed either if the random bit chains are shorter than the encoded messages or if only a greater block of message bits is not interrupted by random bits [PfPf\_90, §3.2]. At this a block of about 10 bits passes for a greater block.

This weaker attack can be impeded easily by mixing the interspersed random bit chain and the message, that was encoded, by **encoding with a completely different (symmetric) encryption system** (e. g., DES with a publicly known key), before it is encoded with RSA.

An alternative approach is that the plaintexts have to fulfill an appropriate **redundancy attribute**, cp. §3.6.4.1. If the plaintext  $z, N$  that was decoded by a MIX is not fulfilling the redundancy attribute the MIX will not put out any part of the message. To impede extensive trying, if one is not sure of the cryptographic strength of the redundancy attribute the senders of falsely created messages can be identified by collaboration of all MIX's: each  $MIX_i$  specifies for the message that is traced back the explicit encoded random bit chain  $z_i$  in the case that a deterministic asymmetric encryption system is used (resp. when using an indeterministic asymmetric encryption system the bit chain that was used. If  $MIX_i$  is not able to do this the indeterministic asymmetric encryption system is inapplicable). The senders can ask for an explanation as long as all transmissions of messages are public or all messages are authenticated by the actual sender. It is noted with back tracking of messages that only the generator

is held responsible when using untraceable return addresses. The user must be able to stay anonymous, otherwise communication partners can be identified by sending falsely created return addresses.

#### 5.4.6.9 Faster transmission by using MIX-channels

The former described MIX-network is mostly used for unobserved transmission of single messages. But the MIX-net is also applicable for channels in principle.

It is shown in the following how the delay for channels that is caused by MIX-operations can be reduced and the distribution of all messages (i.e., channels) can be avoided. Though the resulting MIX-channels are not applicable regarding the constraints of the narrow band ISDN but they are the basis for the absolutely applicable time-slice-channels (cp. §5.5.1).

The hybrid encoding realizes simplex-channels in a way:

The main part of each message  $N_i$  is just encoded with a simple symmetric *streamcipher*. Since applicable streamciphers have en-and decoding velocities in a range of Mbit/s and allow en- and decoding of each bit immediately; the transmission of this main part is not delayed.

The beginning of each message  $N_i, i > 1$  is delayed distinctly: The first block of the message is encoded asymmetrically. For decoding the receipt of the whole first block must be awaited and the actual decoding is comparatively slow if asymmetric encryption systems are used. The effort for change ordering and for the test of repetition of message-beginnings is to be added.

Therefore the slow channel-building is separated from the actual transmission of a message [Pf1.85, §2.1.2] and is realized by independent signaling messages.

Hereby a partitioning of the available bandwidth of the subscriber's connection in both directions is made: a signaling part for establishing a channel and a data part for the actual transmission of messages. The type of the partition is not relevant for the following procedure, but in this section the static partition in a signaling channel and several data channels that is used with ISDN is taken over.

In order to establish the channel a MIX-input message  $N_1$ , called **channel establishing message**, is sent over the signaling channel, which sends the key  $k_i, i > 1$  needed for the symmetric part of the hybrid encoding to the MIX's. Thus it is not distributed by  $M_m$ . In order to keep it uniform  $k_1$  is the abbreviation for the key  $k_{1A}$  which is agreed between the sender  $A$  and  $M_1$ .

Each MIX memorizes the assignment of the channel establishing message from the input batch to the output batch. Each channel establishing message is now assigned to an input channel of  $M_i$  regarding to its position in the input batch of  $M_i$ . The assignment of the channel establishing messages from the input batch to the output batch defines at the same time the mediation of the input channels of  $M_i$  to the output channels of  $M_i$ , e.g., an assignment of input- to output frequencies when a frequency multiplex is used. Therefore a channel, over which the actual message could be transmitted now, is defined by all MIX's by mixing the channel establishing message.

The distribution of all channels to all power supply lines seems to be less sensible because it is assumed that each network termination is able to receive only a few, using ISDN two channels. The problem of the receiver of being unobservable in front of the sender is therefore solved in another way, similar to the untraceable return addresses in [Chau\_81].

In the first instance the problem is eluded by equating sender and receiver: a channel can be used as **sending channel** or as **receiving channel**.

A **sending channel** is the precise analogue of the hybrid encryption: the sender establishes the channel, encodes continuously the stream of information  $N$  as

$$k_1(k_2(\dots k_m(N)\dots))$$

and transfers it to  $M_1$ .  $M_1$  decodes continuously using  $k_1$  and transfers the stream in the same way as the appropriate channel establishing message to  $M_2$ . All MIX act in the same way, i.e.,  $M_m$  creates the plaintext stream  $N$  at the end.

A **receiving channel** is a sending channel that is used “backwards”: The *receiver* establishes the channel (with the same type of channel establishing message).  $M_m$  receives from the *sender* the information stream  $N$  which is not encoded specifically (but of course end-to-end encoded) for the MIX’s, encodes it using the key  $k_m$  and leads  $k_m(N)$  “back” to  $M_{m-1}$ . The other MIX’s do the same, i.e.,  $M_1$  puts out the encoded stream  $k_1(\dots k_m(N)\dots)$ . Since the receiver knows all keys  $k_i$  he is able to specify  $N$ .

Sending and receiving channels are completely symmetrical in terms of their unobservability: As well as the sender is able to use his sending channel without being identified by the receiver the receiver is able to use his receiving channel without being identified by the sender. Whereas the receiver of a sending channel is observable by the sender and the sender of a receiving channel is observable by the receiver.

In order to combine both advantages and to exclude both disadvantages the actual **MIX–channels** are created as links of sending– and receiving channels (Figure 5.33, 5.34 and 5.35): The sender establishes a sending channel which ends at  $M_m$  and the receiver establishes a receiving channel which starts at  $M_m$ .  $M_m$  diverts the information stream that arrives at the sending channel to the receiving channel.

The channels that are supposed to be linked are specified by a common **channel flag** which is received consistently in both channel establishing messages by  $M_m$ . It must be known by both partners, of course, and because one can hardly expect static and outside of the network linked relations, the flag must be chosen by calling Partner  $R$  and must be sent to the called partner  $G$  within a **channel request message** (for logical reasons in tandem with a key for end-to-end-encoding and with the information if  $R$  wishes to send or to receive). In Figure 5.33  $R$  was arbitrarily assumed as sender and  $G$  as receiver of the MIX channel. It is possible, of course, to resume the channel request message and the channel establishing message of the calling to a single signaling message.

Channel request messages, sending– and receiving channel establishing messages represent the three types of signaling messages of the procedure. The transmission of all

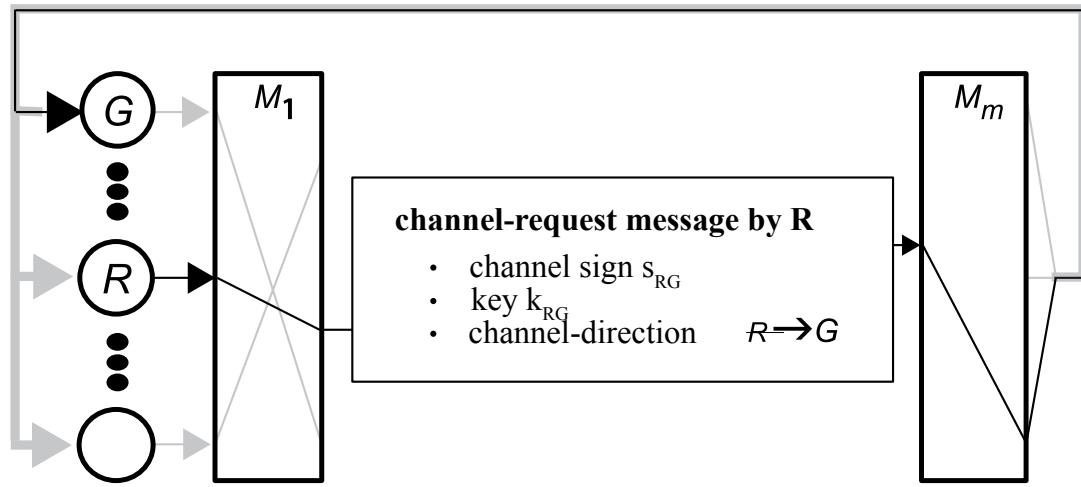


Figure 5.33: Processing of the channel request message sent by R

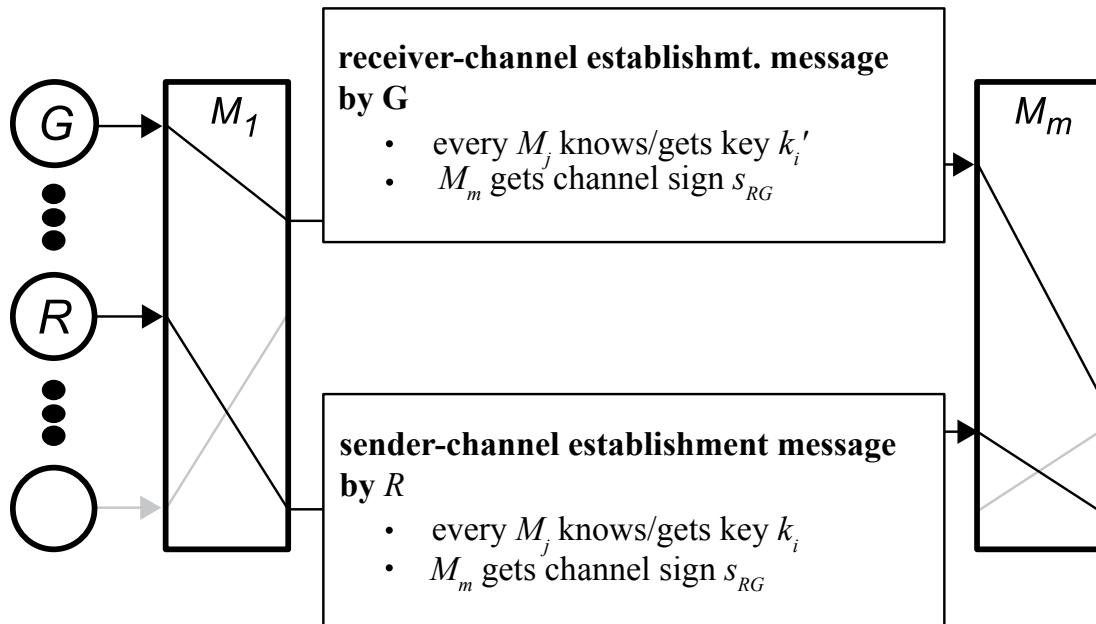


Figure 5.34: Processing of channel establishing messages

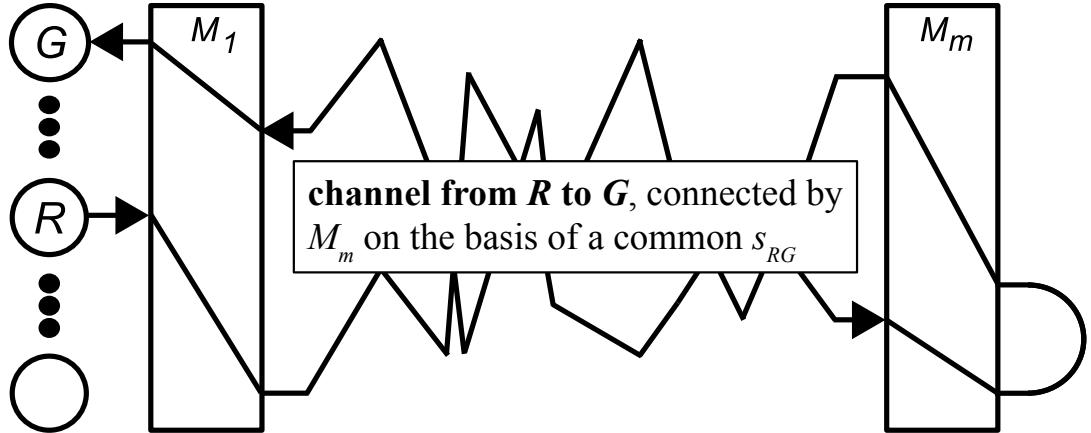


Figure 5.35: Connecting channels

signaling messages is done by using the basic scheme explained in §5.4.6.2 apart from the channel request messages which will be distributed.

Since the size of the input batches influences the effort of the MIXs decisively it is useful to mix the messages of the three different types separately. The unobservability is not limited when used in this way.

To minimize the search in terms of repetitions, either the time stamps of the MIX–input–messages have to characterize the batch as well as the type of message or the MIX have to use different keys for all three different types of messages for changing the encoding. Specification of message types causes less effort for synchronization for the network terminal but on the other hand a strict synchronously service is supposed and the synchronization channel represents the main bottleneck of this procedure. Hence the second possibility is supposed in the following.

As in the basic scheme the problem of observing the sender in a sending channel is solved by having the network terminals maintaining the same number of sending channels (if necessary pseudo sending channels) at anytime.

Without distribution the receiving of a receiving channel is now observable. Therefore each network terminal has to maintain the same number of receiving channels (if necessary pseudo receiving channels) at anytime, too.

The equal length of all channels is guaranteed by the MIXs: if there is nothing transmitted on sending– or receiving channel, each MIX  $M_i$  is bridging this by sending meaningless data.

Thereby it is particularly avoided that the sender identifies the receiver by transmitting no information on the sending channel and therefore the receiver will receive nothing anything at all. That is the reason why the receiver establishes a receiving channel by himself apart from [Pfi1\_85].

It is also prevented that pseudo receiving channels attract too much attention when

data is missed.

A **duplex–channel** is established by two opposite running simplex–channels. Both channels could be established by the calling with the same channel establishing message. This saves transmission capacity, of course, but in the following this will be exposed as obstructive. Therefore from now on two completely separated MIX channels are supposed. Though a common channel request message suffices.

In §5.4.1.3 it was described in general how a changing attack by a sender on the appropriate receiver can be prevented. In [PfPW1\_89, §2.2.3] this happens for this specifically discussed situation. Since the changing attack by a receiver on the appropriate sender cannot be prevented, as it is described in [PfPW1\_89, §2.2.4], and is also possible for MIX–channels, one could argue that this measure is completely senseless for duplex–channels because the attacker is able to attack his partner as sender as well as receiver.

For practical reasons the prevention of an attack on the receiver is useful nevertheless because it would be much easier as an attack on the sender: Instead of disturbing sending actively, as it is described in [PfPW1\_89, §2.2.4.2] the sender has to be able to observe all receiving channels entirely. Moreover this attack on the receiver succeeds immediately while the attack on the sender just represents a test which has to be repeated logarithmically in the number of the senders in order to find the sender.

Since the usual costliest tasks for the usual MIX channel, viz sorting of messages and detecting repetitions, are canceled, i.e., just a simple changing of the encoding and mediation of input and output channels takes place, the effort for the MIXs for managing the MIX channels can be neglected. Changing the encoding of the data by using the symmetric stream cipher hardly carries weight because it can take place with a much higher data rate than the transmission (cp. §5.5.2).

This procedure could not yet be used under the constraints of the narrow band ISDN. Beside the consideration that this procedure could not be executed homogeneously for all network terminals at once but only within certain anonymity sets, there is another fatal problem. As for the messages in §5.4.6.1 it applies for MIX channels also:

- Each network terminal has to contribute the same number of channel request messages and sending– and receiving channel establishing messages for each channel request input batch of  $M_1$ , and maintain an appropriate same number of sending and receiving channels.
- Messages of one batch have to be of same length, what means here especially that channels that were established at the same time have to be removed at the same time.

This would mean above all that each network terminal has to wait with the removal of a channel (even if it is a pseudo channel) until the longest channel that was established at the same time has been finished. Since there are only two 64–kbit/s–channels available for a participant with a basic connection in the narrow band ISDN this would be unreasonable.

This limitation is eliminated by the procedure described in §5.5.1 .

#### 5.4.6.10 Fault-tolerance at MIX-nets

Other error correction methods like end-to-end-time limits and repeated transmission by transgression of the time limits are necessary in MIX-networks without distribution of the “last” (cp. §5.4.6.3) information unit, because this error correction method is not working satisfactorily: There are no useful time limits for electronic mail for example. Furthermore, when using untraceable return addresses, the original generator is supposed to generate a new return address as a replacement for an address that became unusable by loss of a MIX – the user of the return address can not do that (cp. §5.4.6.3). This problem can be avoided by anonymous polling instead of “normal” untraceable return addresses, (cp. §5.4.6.6)

For these motivated error tolerance procedures and for procedures that will be deduced in the following subsections a common and appropriate graphical notation is used, which can be seen in Figure 5.25. The transmission of a message (as an example for a unit which is independent from others) from sender  $S$  over 5 MIX’s to the receiver  $E$ , the applied keys and their awareness, as well as the order of encryption-, transmission-, and decryption are represented. The simplest scheme of encryption (scheme of changing the encoding directly) is mapped as it can be seen in Figure 5.24 and 5.25.

Because there are no information units illustrated, one can understand Figure 5.25 as representative of the (abstract) scheme of encryption-, transmission-, and decryption. Equally Figure 5.25 can be seen as a representation of the scheme of changing the encoding indirectly – as well as for senders and receivers anonymity which shows only the message-independent keys as well as en-and decryption, because there are no encryption structures shown precisely.

In Figure 5.25 the property of series can be seen better than in Figure 5.24. If only one MIX drops out on the way between  $S$  and  $E$  the encryption and the transmission of the message can not be proceeded.

As it is explained in §5.4.9 and apparent from Figure 5.40 the MIX network is based on a communication network that can be realized in the layers 0 (medium), 1 (physical), and 2 (data link) as well as the lower layer of layer 3 (network) regardless of demands of anonymity and unchainability. As previously reasoned, this implies that the layers of the communication network can use conventional error tolerance measures without endangering the anonymity and unobservability of network participants. Thus transient as well as permanent errors within these layers can be tolerated by them.

MIX’s themselves have to be and can be built in a way that errors (or changing attacks) can not impinge on a MIX boso that he puts out his decryption key, mixes units of information more than once, or puts them out in a wrong order. Whereas it is acceptable that the MIX forgets his key and is discontinuing his service completely (fail-stop-service [ScSc\_83]) in case of a fatal error (or a changing physical attack). In order to be supposed to tolerate errors external as rarely as possible it is possible to build up the MIX as internally error-tolerant as long as the conditions mentioned above are kept in case of errors or changing attacks.

Transient errors in the MIX’s that are externally observable are recognized by end-to-end protocols between sender  $S$  and receiver  $E$  and are tolerated by repeatedly sending.

(In order to increase performance an additional intermediate state with MIX-to-MIX protocol is possible.) Therefore only the externally observable permanent errors in MIX's remain. These errors can be recognized resp. localized by using the usual techniques, e.g., the loss of a MIX by using time limits resp. the diagnosis, which MIX has dropped out by staying away of a life signal that must be given cyclic (message with date and signature, I am alive message). Permanent loss of MIX requires appropriate methods to remedy the errors. There are, in principle, 3 possibilities:

The first one, described in [Pf1\_85, §4.4.6.10.1.1], requires no coordination of the MIX's. But it represents an end-to-end protocol, which is—as mentioned—adverse for unchainability and therefore for anonymity and unobservability.

In contrast to that, coordination between MIX's is necessary in the other two possibilities and therefore a non end-to-end working error correction is possible, because in these cases the MIX's are able to substitute other MIX's. How this is supposed to happen and what must be remembered is covered in [Pf1\_85, §5.4.6.10.2].

Characteristics of shifting channels are covered in [Pfit\_89, §5.3.4] for all three possibilities. In [Pfit\_89, §5.3.4] a quantitative rating of all three possibilities is shown.

#### 5.4.6.10.1 Different MIX-sequences

In case of the first possibility the sender tries to repeat a transmission on another way (Figure 5.36), i.e., over a disjoint MIX-sequence (also known as ([EcGM\_83], a bit shorter also in [BeEG\_86]) *dynamically activated redundancy*). This solution is easy, but slow (as in general in case of dynamically activated redundancy), because the end-to-end protocol has to detect the error first in order to start a second transmission, possibly without knowing which MIX has got broken.

In a MIX-network without distribution of the “last” information units and without anonymous calling up untraceable return addresses with a long validity in order to guarantee receivers anonymity are necessary according to §5.4.6.2. In case of untraceable return addresses with long validity the procedure of error tolerance mentioned above is very complex if there is no temporal relation between messages and responses, as it is the case with electronic mail, for example. The receiver  $E$  is not able to try another way for transmission if the received return address falls out (at least one MIX is defect in this sequence). This also applies if always two or more return addresses with long validity (*statically generated redundancy*) are exchanged and one MIX is defect in each sequence. The original sender  $S$  (more exactly: his participant station) would be supposed to send more and more new return addresses to  $E$  (more exactly: to his participant station), as long as he does not receive a response from  $E$ . This would be completely superfluous in most cases because  $E$  just had not had the time to answer. Alternatively  $S$  is able to inform himself about which MIX's have fallen out in the meantime <sup>12</sup>. Then he would send new return addresses to  $E$  only if all sent return addresses to  $E$  with long validity

---

<sup>12</sup>Same behavior of all participant stations and consistent distribution of information about which MIX's have fallen out, is important here, too. Otherwise attacks on the anonymity of the owner of the return address would be possible. Analogous attacks on the anonymity of the receiver can take place, according to the attack described in §5.4.1.1, second item

are fallen out or have fallen out. Because, in the last case, it would also be possible that—after an error attempt of  $E$ —the return addresses are permanently unusable for  $E$ .

This first possibility of error tolerance boils down to a **two-phases-concept** which includes all stations. In one phase information units are transmitted anonymously, in the other phase errors are tolerated, e.g., in case of a fallen out MIX by sending new untraceable return addresses where the fallen out MIX is not needed from sender to receiver (also known as: *dynamically generated*, dynamically activated redundancy.) Since all addresses must be untraceable return addresses (§5.4.6.4) in a MIX network without distribution of the last information units to all stations and without anonymous calling up for reasons of mutual anonymity of sender and receiver, this new distribution in the MIX requires an indirection step cogently in terms of non-anonymous places for new distribution of the addresses (e.g., the already mentioned address directories with untraceable return addresses (§5.4.6.4).): After a MIX has fallen out all stations will be informed. Each station sends a list of anonymous addresses that became unusable, an anonymous displacement address and a second anonymous address contingently to the non-anonymous places for distribution of addresses with a scheme for senders anonymity over the MIX's that are still working.

In a MIX-network with distribution of the “last” information units to all stations, untraceable return addresses are superfluous because the receiver is completely protected by the distribution and the use of normal implicit addresses. Thus in such a MIX-network there is no need to exchange new addresses when a MIX falls out. Of course a list of all fallen out MIX's should be distributed to all stations, in order to guarantee that those stations will not use these MIX's for the scheme of senders anonymity.

This is also valid for the procedure of anonymous calling.

A variation of the procedure of end-to-end-protocols with statically or dynamically activated redundancy which would save time in case of an error consists of executing each anonymous information transfer parallel over several disjoint MIX-sequences (statically or dynamically generated, statically activated redundancy).

A disadvantage of each end-to-end error correction is that statistical attacks using rates of sending information units are possible, as already mentioned at the beginning of this chapter. These attacks could cause a chaining of traffic events if applicable. This is distinctive especially in MIX-networks without distribution of the “last” information units and with individual choice of the end-to-end time limits (more exactly: the intervals of the time limits, in which it is reacted at a random point in time), resp. with individual choice of number of information transfers that are executed parallelly.

If there is a possibility to ignore statistical attacks using sending rates of information units, the following protection criteria applies:

**protection criteria:**

Different MIX-sequences are as secure as the original schemes (cp. §5.4.6), as long as at least one MIX in a used MIX-sequence is not controlled by an attacker.

Here it is unimportant if the not-controlled MIX is mixing correctly or is mixing (e.g., if he is fallen out) nothing at all.

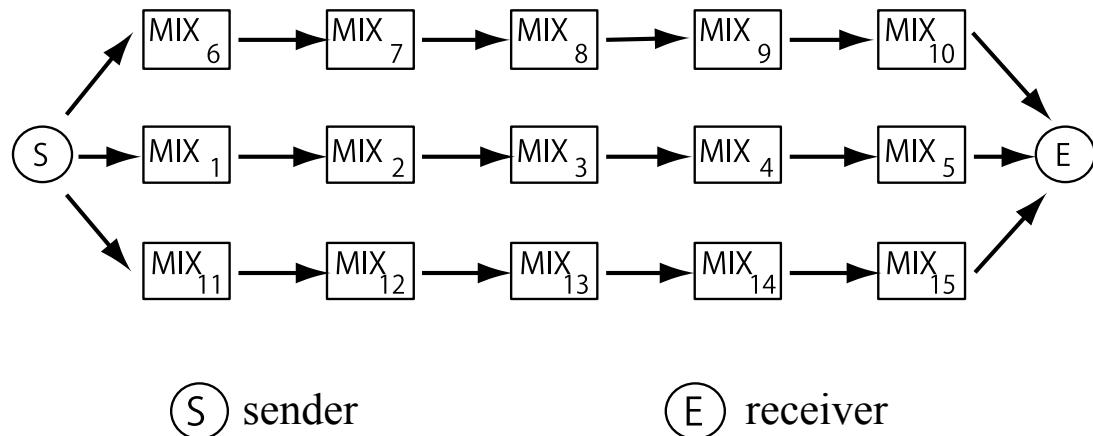


Figure 5.36: two additional alternative ways over disjoint MIX sequences

#### 5.4.6.10.2 Substitution of MIXs

To facilitate a non end-to-end working error correction and avoiding therefore all disadvantage of the first possibility the MIXs are put in position in the second and third possibility to substitute other MIXs. This is advantageous in particular for the availability, but causes the *coordination problem* which will be described in the following subparagraph.

The possibility of substituting MIXs is supposed to be statically (statically produced redundancy) – there would be no need to substitute MIXs if dynamically produced redundancy is used. A completely different MIX sequence could be chosen as described in §5.4.6.10.1 .

Statically produced redundancy can be activated either statically or dynamically in general.

A static activation in regards to the substitution of a MIX is thereby no different from a distributed, internal fault-tolerant implementation of a MIX (cp. §5.4.6.10): The function of sending the input to all distributed parts of a MIX is delegated to the antecedent MIX (resp. a succumbing communication network). The function of collecting outputs of all distributed parts and the creation of the final result is delegated to the subsequent MIX. Coordination of all distributed parts of a MIX can not be delegated (cp. §5.4.6.10). The coordination must ensure that all non-failed parts can be processed on the same input-information-units in each batch. Otherwise a changing attacker could cause differences between input- and output-information-units of single parts of the MIX and use them for bridging the distributed implemented MIX in terms of the protection of the communication relation. This succeeds if averages or differences of the batches have cardinality 1, cp. §5.4.6.

A static activation regarding the substitution of several MIX's in sequence is problematic in terms of the protection of communication relations, because

- the multiply mentioned coordination problem exists for the MIX in sequence regarding to each step,
- an information unit can only be protected in terms of communication relations with such information units that are passing through the MIX in sequence at the same time, i.e., each same MIX in serial as well as equivalent sequences parallel.

A static activation regarding the substitutions of several MIX's in sequence is only possible if static cascades of MIX are given (cp. [Pfit\_89, §4.2.3.1].

Since both possibilities generated static, static activated redundancy seems to be very costly on the one hand and on the other hand uninteresting in terms of the leeway of the concept of generated static, dynamically activated redundancy is treated in §5.4.6.10.2.1 and §5.4.6.10.2.3.

#### 5.4.6.10.2.1 The coordination problem

In the MIX–network, as it was described in §5.4.6, each MIX was responsible for mixing information units at most once with one key pair as long as he owns it. Otherwise an attacker could bridge a MIX, rather an information unit, mixed twice, because this information unit will be with high probability the only information unit that appears twice in the appropriate outputs of a MIX. If MIX's can be put in the position to substitute another MIX this response would be transferred to a *team* that is created by him and all MIX's which could substitute him. Thereby it is important that this response is abided by, if an arbitrary subset of the team falls out and the communication between team participants that are still working is broken. In particular, it is important that information units that have already been mixed by failing or momentarily unreachable team members are not being mixed by a team member again – although an attacker will try to do exactly this.

This task can be solved by a simple protocol between the team members, though the effort of this protocol is considerable because of the additional time delay of messages caused by its execution and the additional communication between and the additional memory cell in the MIXes.

##### Inefficient coordination–protocol [Pfi1\_85, page 74]:

Before a MIX is mixing information units he sends all his input information units to all team members that have not fallen out.

The MIX is only mixing an information unit after having received a confirmation from all members that they have not already mixed this information unit and will not mix it.

A fallen out MIX receives all information units from the other members, that have been mixed or are supposed to be mixed before he will start mixing again.

To be able to execute this protocol each MIX of the team has to save all information units which have been mixed by himself or by the other members. In order to handle

the required memory, the four methods described in §5.4.6.6 can be used for reducing memory – and searching effort (exchanging publicly known decryption keys of the MIX's more often, time stamp, abbreviated hash-function, just saving, and comparing a certain part of a message).

If the method of the abbreviating hash function or the method of saving and comparing only certain parts of messages is already applied by the requester and not by the answerers (in the above described coordination protocol) first, this method would reduce not only the effort for saving and searching but also the effort for communication.

Other methods for reducing the effort for communication and the time delay of the proper message are specifically for the second and the third possibility and will be therefore described in §5.4.6.10.2.2 and in §5.4.6.10.2.3.

The above mentioned inefficient coordination protocol requires that the MIXs know for sure which MIXs of the team have fallen out or not. Otherwise an attacker could try to isolate or to persuade two groups that the MIXs of the other group have fallen out. If he succeeds a successful attack would be easy. To avoid this even if the MIXs do not know for sure which MIXs have fallen out resp. have not fallen out the above mentioned coordination protocol could be extended by the following rule:

It is only mixed if an absolute majority works (and is able to communicate).

This can be guaranteed by the usual techniques of authentication, cp. §3. Here – as in the efficient coordination protocols which will be described in the following – the requesting MIX will be counted as active resp. positive when the absolute majority is ascertained.

#### 5.4.6.10.2.2 MIXs with reserve-MIXs

The second possibility (of the three that were announced in §5.4.6.10) consists of the fact that the secret key of a fallen out MIX is known by one or more other MIXs who e.g., are prosecuted by the same organization (Figure 5.37). Or this key can be reconstructed (by using a scheme of threshold value, cp. §3.9.6) by a lot of MIXs that are prosecuted by organizations that trust each other [Pf1\_85].

Both methods facilitate a MIX-to-MIX protocol without changing the address scheme or the decryption scheme by letting the sender or the MIX transmit the information unit to a member of the appropriate team if one or more MIXs haven fallen out. Therefore information about fallen out MIXs or the team structure is needed. Both can be either known globally or communicated by request – the failure of a MIX, e.g., by a missing answer (receipt). Here it is not expanded on the likewise important information about the reachability of MIXs which can be limited by loss of parts of the underlying communication network.

Expressed with a concept of the world of error, tolerance it is about a *statically generated, dynamically activated parallel-redundancy*.

As announced, it is possible to limit the effort of the coordination protocol for avoidance of multiple mixing of the same information unit:

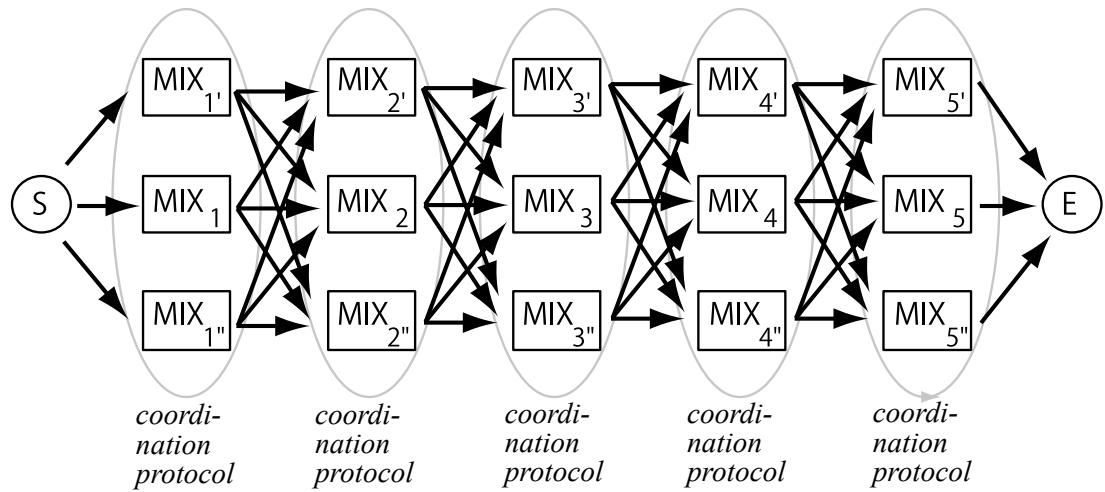


Figure 5.37:  $MIX_i$  can be substituted alternatively by  $MIX_{i'}$  or  $MIX_{i''}$  ( $i=1,2,3,4,5$ )

#### Efficient coordination protocol for MIXs with reserve MIXs

[Pf1\_85, page 75f]:

Any additional delay of mixing can be avoided if each MIX declares with his reserve MIXs that they will mix with only one key when it has fallen out. Then it suffices that he sends at the latest his input-information-units to a distinct greater number than the absolute majority of team members when he puts the output-information-units out. If he stops mixing when he does not receive authenticated receipts by a majority of team members within a certain time interval, an attacker would be at most able to bridge him by isolating information units from which he has not received a receipt yet.

Reserve-MIXs whose failure could be proofed (e.g., by authenticated communication with its communication processor) do not need to be counted when the (absolute) majority is ascertained.

If a MIX has fallen out the reserve MIXs are able to determine a *substitute* from among them (and e.g., qualifying him for reconstruction of the MIX-key if a threshold scheme, cp. §3.9.6, was used). The substitute acts in the same way as the MIX if he would not have fallen out until he himself will fall out (and the remaining reserve MIXs determine a new substitute) or the substituted MIX is repaired and starts mixing in synchronization with the end of the mixing of his substitute.

In my opinion the risk, that an attacker is able to bridge a MIX by isolating information units for which he has not received an receipt yet, is balanced out by far by avoidance of each additional delay of mixing. Because changing attacks which try to use this weakness of the coordination protocol can be detected easily (at least when they occur often) and have to intervene in many places in a synchronized way if information units are passing

through several MIXs each: To trace the path of an information unit from sender to receiver the attacker is supposed to bring the active MIX of each team to failure or to isolate him after he had mixed the information unit.

a MIX should choose as reserve MIXs, such MIXs which are more than 10 kilometers away from each other for ensuring that an error is effecting other MIXs as little as possible [Pfi1\_85, page 76].

If an organization prosecutes several MIXs that are spatially far away from each other it seems to be very useful to let these MIXs work as reserve MIXs for each other.

**protection criteria:** If the MIXs are coordinated in a suitable way then MIXs with reserve MIXs are as secure as the original schemes (cp. §5.4.6) as long as the team with a MIX that is used in the decryption structure is not controlled by an attacker.

A team is not controlled by an attacker, at least not if he does not control a team member or he is not controlling the proper MIX as well as the substitute if a threshold scheme is used, and when he additionally does not control enough MIXs of the team as they are needed for reconstruction of a key (and therefore he will not ever receive the key) and when he is not controlling an absolute majority of a team (otherwise this majority would be able to establish a substitute without a loss of the currently mixing MIX and let both mixing parallelly and uncoordinated.)

#### 5.4.6.10.2.3 Leaving out a MIX

The third possibility is to ensure that each MIX receives sufficient information in each message to be able to bridge and to leave out a MIX (more generally: up to  $b$  MIXs with an arbitrary natural number  $b$ ) [Pfi1\_85, page 77f].

In contrast to the second possibility, the third needs changes of the address schemes or decryption schemes to facilitate a MIX-to-MIX-protocol too. These changed address schemes and decryption schemes will be derived particularly in this section.

Further variations are referred to in [Pfit\_89]. In [Pfit\_89, §5.3.2.3.2] protection criteria for characterization of the achievable anonymity for relations of communication by use of these address and decryption schemes are established. In [Pfit\_89, §5.2.3.3 and §5.3.2.3.4] proper and efficient coordination protocols for two possible modes of operation are developed: Either only fallen out, i.e., as few as possible, MIXs are left out or as many as possible. The latter complicates the coordination problem but allows a profit of efficiency in some cases.

To facilitate understanding we will begin with the first case which describes that each MIX is able to leave out the next MIX in a sequence of chosen MIXs. In this case the procedure of error tolerance can tolerate the loss of one or even more MIXs that do not adjoin each other.

To leave a MIX out his predecessor must not only be able to create the message meant for him but also the message meant for his successor (Figure 5.38).

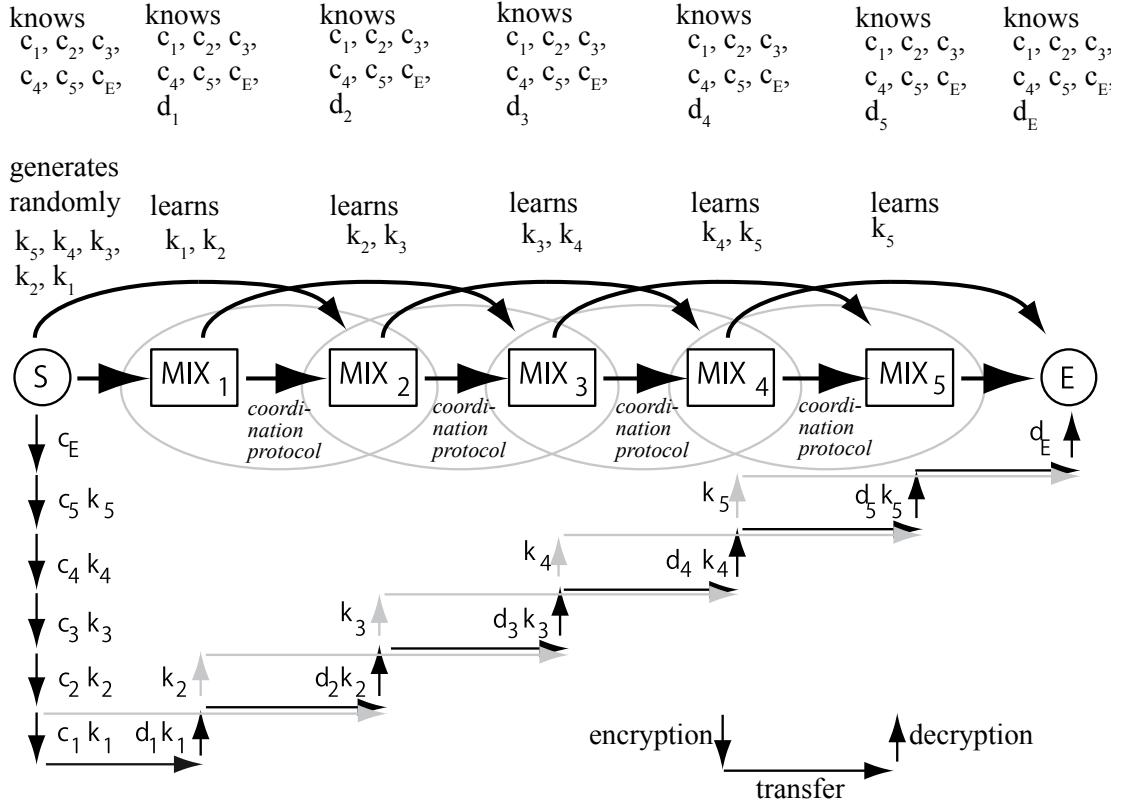


Figure 5.38: One MIX can be left out each; the coordination protocols are transacted in the picture within groups of minimal circumference

If each MIX keeps the messages for his successor and again for its successor separately the length of the message grows exponentially with the number of used MIXs. To avoid this the sender of a message chooses (as in all schemes of changing the encoding, cp. §5.4.6) a different key for each MIX (e.g., of an efficient symmetric encryption system) with which this MIX decodes the message meant for his successor. Each MIX receives this key and the key meant for his successor encrypted with its publicly known encryption key. He will receive the rest of the message separately and encrypted with the non-publicly known keys. The addresses of the following two MIXs can be prefixed unencrypted to each message or can be encrypted with the mentioned two keys together with the publicly known encryption keys of the MIX. This will be described more formal in the following.

As in §5.4.6  $A_1, \dots, A_n$  is the sequence of addresses and  $c_1, \dots, c_n$  the sequence of publicly known cipherkeys of the chosen MIX-sequence  $MIX_1, \dots, MIX_n$ , where  $c_1$  can be also a secret key of a symmetric encryption system.  $A_{n+1}$  may be the address of the receiver which is called  $MIX_{n+1}$  for simplification, and  $c_{n+1}$  may be his cipherkey.  $k_1, \dots, k_n$  may be the chosen sequence of non-publicly known keys (e.g., of a symmetric encryption

#### 5.4 Basic measures settled inside the communication network

system). The sender creates messages  $N_i$  which will be received by  $MIX_i$  according to the following recursive scheme for creation originating in message  $N$  which is supposed to be received by  $MIX_{n+1}$ :

$$\begin{aligned} N_{n+1} &= c_{n+1}(N) && \text{(Scheme 1)} \\ N_n &= c_n(k_n), k_n(A_{n+1}, N_{n+1}) \\ N_i &= c_i(k_i, A_{i+1}, k_{i+1}), k_i(A_{i+1}, N_{i+1}) && \text{(for } i = 1, \dots, n-1\text{)} \end{aligned}$$

This scheme is not the only one that is suitable. For instance the next schemes fulfills the same purpose:

$$\begin{aligned} N_{n+1} &= c_{n+1}(N) && \text{(Scheme 2)} \\ N_n &= c_n(k_n, A_{n+1}), k_n(N_{n+1}) \\ N_i &= c_i(k_i, A_{i+1}, k_{i+1}, A_{i+2}), k_i(N_{i+1}) && \text{(for } i = 1, \dots, n-1\text{)} \end{aligned}$$

The sender sends  $N_1$  to  $MIX_1$ .

In both schemes  $MIX_i$  is able to compute the messages  $N_{i+1}$  and  $N_{i+2}$  as well as the addresses  $A_{i+1}$  and  $A_{i+2}$  from  $N_i$ .

In case of some coordination protocols that will be discussed in the following it is important, too, that  $MIX_i$  is able to proof if his predecessor  $MIX_{i-1}$  has not change a bit of the message  $N_i$  (and correlative no one else, because  $MIX_{i-1}$  discovered the most about the message and was therefore able to act most single-minded)

As long as the encryption systems are not broken <sup>13</sup>, an attacker is only able to change specific parts of a message for which he knows the applied key. Otherwise the message would just get lost or is ignored by a station where the message is recognized as incorrect with the help of an error-recognizing code. In both cases the attacker would not receive some helpful information. Thus  $MIX_{i-1}$  has the possibilities to substitute the whole part which was encrypted with  $c_i$  (without being able to read the content of the original part, because he knows  $c_i$  but not  $d_i$ ) or to manipulate only the part of the text which was encrypted with  $k_i$ .

The danger is here that he could try to change addresses. This is not critical for the protection of communication relations as long as addresses are only used for transmission of messages because it is assumed in any case that the attacker controls all wires. But it can become critical when the messages specify which MIXs have to coordinate with each other to ensure that no message is mixed twice.

In both schemes the attacker ought to change the part encrypted with  $c_i$  in order to change an address. To do so he ought to invent a new key  $k_i$ . But this would be recognized after decryption of the part encrypted with  $k_i$ , viz  $k_{i+1}(A_{i+2}, N_{i+2})$  in scheme 1 or  $k_{i+1}(N_{i+2})$  in scheme 2. Therefore these parts have to contain enough redundancy for ensuring that  $MIX_i$  is able to detect this.

---

<sup>13</sup>The encryption systems are assumed to afford more than they are supposed to in order to keep them easy, cp. §3 and exercise 3-10d). If the encryption systems would not afford this they must be expanded. This can happen, e.g., by using a collision resistant hash function, cp. §3.6.4.1

If a coordination protocol is used where this check is not possible,  $A_{i+1}$  can be left out in scheme 1.

The choice between scheme 1 and scheme 2 (i.e., if  $A_{i+2}$  is contained in the first part, encrypted with a costly asymmetric encryption system, or in the second part, encrypted with a less costly symmetric encryption system) depends on the used asymmetric encryption system: If the length of the information units that are encrypted with the asymmetric encryption system for security reasons is supposed to be so long (as it is the case in RSA, cp. §3.6) that  $A_{i+2}$  fits in in any case, it is more useful to use scheme 2 instead of scheme 1 where it is filled up with random bit chains. Because it is less costly trivially to encrypt  $A_{i+2}$  with the more costly asymmetric system without additional effort than en-and decrypting  $A_{i+2}$  with the less costly symmetric encryption system with additional effort. If  $A_{i+2}$  contains any redundancy it is advised that the entropy (i.e., the random and therefore the unknown information content to the attacker) of  $k_i, A_{i+1}, k_{i+1}, A_{i+2}$  does not get too low. But since the symmetric encryption system must defeat a complete search over the set of keys (cp. §3), thus the length of the key must be too huge for this attempt (which is the case with a length greater than 100 bit),  $k_i$  as well as  $k_{i+1}$  contain enough entropy.

The interested reader can find many more encryption schemes and a quantitative evaluation in [Pfit\_89].

#### 5.4.7 Tolerating modifying attacks on RING-, DC- and MIX-Networks

In a communication network every error can be considered an altering attack. Errors “by mistake” are a subset of all (physical and logical) altering attacks. Despite this, all errors have been handled in a manner to keep anonymity in all circumstances. An attacker may now permanently create errors, therefore recrypting messages with wrong keys, preventing the transmission, sending his own messages although he would not be allowed by the protocol for multiple anonymity access. Or simply by overlaying the keys invalidly to make a denial of service. This problem of altering attacks on availability can be solved in two ways [MaPf\_87, page 403f].

On the one hand, after every error it is checked if there was an altering attack (and if so by whom). Because the methods for that are quite expensive and do recover the sender and receiver of the message it is better to act only if you have become suspicious (like an abnormal error rate). To discover altering attacks all basic techniques require that *all stations can uncover their own activities*:

- In a MIX-Network every MIX needs to disclose all incoming and outgoing information units from a certain point of time onwards - as long as it is not already done by the communication network. Then everybody can check if a message sent from him was incorrectly encrypted or decrypted by the MIX or if it was not sent at all. To prove this to a third party the contexts of the affected information units

needs to be uncovered<sup>14</sup>. But it is not necessary to reveal all relations of all information units so that this measure would cause big effort but will not endanger the anonymity of the information units transmitted correctly [Chau\_81, page 85].

- In a DC-Network every station must reveal all key tokens (but not the initial value of the PRNG if used) and the token sent for those tokens on which an altering attack is suspected [Cha3\_85][Chau\_88].
- In RING- and TREE-Networks every station must reveal its tokens sent for the tokens on which an altering attack is suspected.

On DC-Networks as well as on RING-Networks and TREE-Networks it is the job of a **revealing technique** globally agreed on to take care:

- A) That *all tokens can be revealed*. Otherwise you could hamper certain tokens without notice.
- B) That *no token is revealed that could undermine the sender's anonymity* of directly sensitive messages or message you can be related to those.

The latter one demands the usage of a reserving technique as multiple access technique.

In [Cha3\_85][Chau\_88] (only) the following combined reserving and **revealing technique for the DC-Network** is included:

- All stations reserve exactly one transmission frame in every round. If there are not as many transmission frames reserved as there are stations participating in the DC-Network, all reservation bits are revealed and it is checked whether every station has sent exactly one reservation bit. Because all stations reserve exactly one transmission frame per round the uncovering gives no information about the stations that stuck to the reservation scheme.
- Prior to the reservation every station sends an *uncovering claim* encrypted with a block cipher. That claim consists of the bit position of its reservation and the message to send. If the station is hampered during sending and it wants the interference to be revealed, it publishes the key, with which its possible uncovering claim is encrypted. This makes the possible uncovering claim a real uncovering

---

<sup>14</sup>Who is to specify the decryption of an information unit (that is who encrypted it) can do it that way: He announces the random number used at the encryption of the information unit. Then everybody can verify (if a *deterministic* asymmetric concealment system was used) whether the encrypted information unit combined with the given random number is equal to the MIX's input. In that case and if the information unit does not appear as an output of the MIX, the MIX has performed an altering attack - unless he can prove that he has processed the information unit in a previous batch so that he has just ignore a recurrence. With a *indeterministic* asymmetric concealment system the encrypter has to announce all other random information. Nevertheless the encryption is *deterministic* too, so that without announcing the decryption function you could deduce the right output information unit.

claim. Because this undermines the sender anonymity of the message a station will only do so if it does not have to send anything. Then the message is a meaningless message or a meaningful trap for the interferer.

David Chaum's revealing method does not accomplish B) at all and - depending on the readers presumption about the station's behavior<sup>15</sup> - A) only in a complexity theoretical way:

- The attacker can accomplish the revealing of any desired tokens sent *before* by publishing possible revealing requests that are not related to his messages. Even though if the reservation bit are also revealed the attacker is exposed.
- If the attacker can break the block cipher used for the revealing method, he can
  - a) hamper without becoming exposed
  - b) reveal picked tokens *after* sending

depending on the behavior of the reserving station and the block cipher used.

a) holds true if the stations will not always send possible revealing requests. Or if wrong bit positions are used for valid messages and the block cipher used (like DES) exclude some mappings from plain to ciphertext. The latter is always true for  $(2^{\text{block length}})! > 2^{\text{key length}}$  (cp. §3).

b) holds true even *after* the attacker has sent his revealing request if the block cipher allows all possible mappings from plain to ciphertext. Otherwise the attacker can only pick the token(s) prior to the sending.

With an unfavorable choice of the stations behavior and the block cipher then even a) is true as well as b) with subsequent choice of the attacker.

After this it is clear that a) can be avoided by the execution of the combined reservation and revealing method (as described above) as far as the attacker can not sign meaningful or meaningless messages. In [WaPf.89][WaPf1.89] revealing methods are shown that do not have the weaknesses from b). The claims A) and B) can be provably achieved even in the information-theoretical world. It should be a goal to expand those results to other classes of multiple access systems (or finding the proof that it is not possible).

Revealing methods for the DC-Network can be canonically transferred to RING-Networks and TREE-Networks.

It should also be noted that for making revealing possible (especially in RING- and TREE-Networks) a considerable additional effort for saving the information units is needed (even if no "errors" occur).

In the rest of this section it is shown how to react to disputes by revealing so that the "guilty" one becomes penalized.

If the *transmissions on all connections are known* (i.e., if a broadcast media is used) the modifying attacker can be discovered without doubt in a MIX-Network (also on

---

<sup>15</sup> [Cha3.85][Chau.88, page 72] does not mention if stations will send revealing requests for valid messages too and if those potential but never occurring revealing requests have the right bit positions and messages. As it will soon be seen, at least the first one seems useful. Besides it eases the description.

RING- and TREE-Networks but because it is their main feature that nobody knows all transmissions on all connections!). But on DC-Networks it is different, because the keys used by the stations must be known to make it possible to decide who has sent. If not all transmissions over the connections are publicly known, the same effect can be achieved by every station signing its information units sent. Even though the signing and storing of the signatures is quite a big effort, it makes it possible to determine what exactly was sent (A special case occurs if both stations on one connection are attackers. Than of course it is not possible to trace the communication).

If in a MIX-, RING-, TREE- or DC-Network *not all transmissions over the connections are known and if not every information unit is signed*, it is possible to identify three groups of stations. The first one is neither accused nor accuses others, while in the second or third group at least one of the two is the case. One of them is the altering attacker. The other one is not guilty but is accused by the attacker. It is assumed that there is only one attacker in the network that may have corrupted several stations, and so contradictions only occur in the attacker's environment. The attacker has only one chance to remain covered if he accuses only a few other stations. Otherwise the majority of the accused but innocent stations would be against him. In the undecidable case that some few stations accuse each other you would continue as usual. But you would reconfigure the stations so that they will not be able to accuse each other again. That can be achieved by:

- In a MIX-Network other connections (addresses) are used so that every sender creates addresses in such a way that no message between these two mixes is passed.
- In a DC-Network no keys can be exchanged between the two affected stations (present keys are removed). The alternative would be that both partners would sign their common key before usage, so that every one has each others signature. In case of dispute a third party can decide (after examining the signatures) who is wrong [Cha3\_85]. The drawback of direct signing of the shared key is that every one of the two stations can not just show but also *prove* the key towards a third party (even without knowledge of the other one). This is done by showing the signature of the other one (his own not is useful because he could easily sign another, false key). That is why [Chau\_88, p. 73f] proposes not to sign the shared key but to sign two parts which summed up are the shared key. To prove the key for a third person both parties are needed because the signature of the partner that did not sign is needed by the other one<sup>16</sup>
- In RING- and TREE-Networks the rings and the trees are reconfigured. This is easy if the topology is set up as virtual ring or tree over a real network. If the network topology is fixed to the physical topology reconfiguration is pretty hard.

If one participant is in a group several times, that accuses another group or is accused on its own, he is considered as a modifying attacker and is therefore excluded from the network. This technique is only practical if relatively few participants are altering

---

<sup>16</sup>Since the invention of undeniable signatures this splitting is not necessary any more (cp. §3.1.1.2).

attackers. Otherwise the altering attackers could cooperate and exclude the correctly communicating stations from the network (of course the amount of stations that can be observed is reduced).

### 5.4.8 Active chaining attacks over limited resources

In this section a universal active chaining attack using the limitation of resources is shown as well as its detection and defense.

The goal of the attack is to determine if two distinct pseudonyms (i.e., implicit addresses) belong to the same station or not. The attack is made up of requesting a service from both pseudonyms with such a load that you can conclude from the speed of the service which pseudonyms belong to the same station. Resources are computation power, I/O bandwidth, memory.

If the

- local *computation power* is limited or
- the bandwidth for *sending* and *receiving* anonymously and unrelatable per station is limited to a proper fraction

the station of course can not handle service requests that exceed these limits in realtime or even at all. This results in the three attacks using one resource each. To conduct such an attack the attacker needs to

- estimate the computation power of the attacked station or
- find out the bandwidth in the communication network (which is easy in a MIX-Network without the distribution of “last” messages (cp. §5.4.6.3) because it is known to every participant; otherwise he has to estimate). For estimating he can violate a fair access protocol (cp. §5.4.5.4.2 for collision resolving algorithms with average values and overlaying) to shorten the bandwidth.
- If an attacker knows the bandwidth a certain amount of attacking stations have to cooperate to exceed it. Or a few station violate the fair access protocol.

If a service is provided on time the result is (assuming knowledge of the bandwidth) that both pseudonyms do not belong to one station. If the service is not on time it could mean that there are other requests producing load. If the attacker knows a pseudonym of each station (i.e., a publicly assigned address) and if each station provides a service under this address, the attacker picks the non anonymous pseudonym out of the two. So, the attacker can allocate every pseudonym to a station. This operation is of linear complexity in terms of the number of stations.

This *active* chaining attack via limited resources can be detected by the attacked station on the one hand: As long as a station does not reach its limits due to external service requests it is not considered under attacked. This situation is much better for verification of the protection than for *passive* attacks.

On the other hand, the active attack can easily be made more difficult:

- Every participant is free to provide as much computation power from his station as he likes for external requests. It can even be (pseudo)randomly distributed across time.
- With DC-, RING- and TREE-Networks all stations are configured to make every station able to send with the whole bandwidth. Although it disables some optimizations for saving resources [Pfit\_89].
- With distribution all stations are configured to make every station able to receive with the whole bandwidth. Although it disables some optimizations for saving resources [Pfit\_89].

With a great effort and tremendous limitations of the usable power, the active attack can be completely prevented by a *statical splitting of the resources*: Prior to the actual service request all pseudonyms from which a service will be requested are published anonymously and unrelatable. Let  $n$  be the amount of those pseudonyms. Every station then provides only the  $n - th$  part of the minimal computation power (as well as the minimal bandwidth) available in the communication network.

With a practical deployment you will have to find a trade-off between dynamical and statical resource splitting.

It should be noted that attacks on resource limitations and their avoidance by statical splitting is already well known: If two processes use the same resource (like two OS tasks the same CPU) with a fair and dynamical resource splitting, every process may slow down the other one by holding back the resource. If one of them has access to the realtime he may notice the delaying. This modulatable delaying is a hidden channel from the other process to this one (cp. §1.2.2).

### 5.4.9 Classification within a layer model

To simplify the development, the implementation and the connecting of communication networks, they are designed as **layered systems**. Every layer uses the services of the layer below as well as it provides convenient service to the layer above. The *International Standards Organization* (short ISO) standardized a 7-layer model, the *Basic reference model for Open Systems Interconnection* (short OSI). Because all international standardizations refer to ISO/OSI the following figures will show the layers for end-to-end and link encryption.

To disable switching centers to access any unnecessary protocol information there has to be an end-to-end connection on the deepest layer that directly connects between the stations. This is why the transport layer (layer 4) is used for that<sup>17</sup>

---

<sup>17</sup>This may not apply to *non canonical* communication networks that consist of protocols above layer 4. There you only have an end-to-gateway and gateway-to-end transport security.

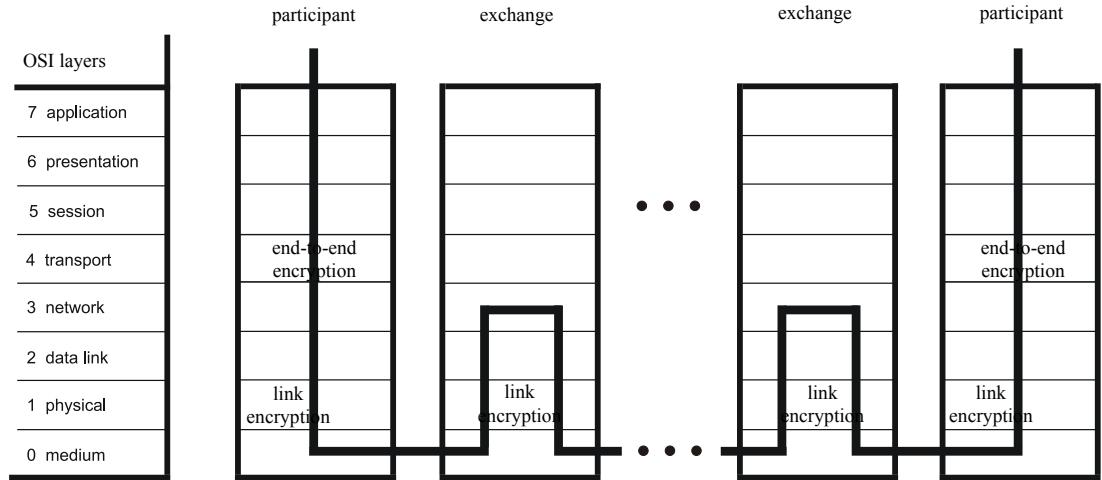


Figure 5.39: Allocation of end-to-end and link encryption in the ISO OSI reference model

Because attackers that can eavesdrop the connections should not gain any unnecessary protocol information the link encryption should take place in deepest layer where digital data is processed. This is why layer 1 is suitable for that. On communication channels that are exclusively assigned to two instances each, you can (cp. §3) conceal if and what for information was transmitted without any additional costs by using a stream cipher.

Those layers are named as possible places for implementing end-to-end and link encryption. Hopefully these arguments will persuade to pick these layers (and not the layers above or none) as place for the implementation, because some implementors hope for a faster and easier implementation. As well as the intelligence services hope for easier eavesdropping.

If you mistrust the implementation on a certain layer regarding confidentiality and integrity, the data transmitted there should be encrypted for guaranteeing concealment and integrity by the layer above (cp. §3). With that additional end-to-end or link encryption connections may arise. The latter seems not that necessary because the deeper layers are of less design complexity than the layers above. Besides, remote maintenance for correcting errors is not necessary in the deeper layers.

The basic technique for protecting the receiver uses broadcasting.

Broadcasting can be realized by an appropriate routing in layer 3 (network layer), so that layer 4 only has to generate and evaluate implicit addresses. One possible realization of broadcast is flooding [Tane\_81] which can be achieved with a very simple path-finding strategy: Every station transmits every message to all neighboring stations from which it has not received this message.

In communication networks especially designed for broadcasting, broadcasting is done on layer 1. At least for some services, layer 1 can pick the information which is assigned for the station. In fig. 5.40 it is called channel selection.

#### 5.4 Basic measures settled inside the communication network

OSI layers	distribution	MIX-Net	DC-Net	RING-Net	TREE-Net
7 application					
6 presentation					
5 session	has to keep anonymity and unlinkability				
4 transport	implicit addressing				
3 network	distribution	buffering and recryption			
2 data link		realizable without consideration	anonym. multiple access	anonym. multiple access	
1 physical	channel selection		key and mess. superpos.		digital signal regeneration
0 medium	for anonymity and unlinkability			ring	tree

Figure 5.40: Allocation of basic techniques for securing traffic data and data on interest in the ISO OSI reference model

MIX- and DC-Networks employ special cryptographic mechanisms to create anonymity and unrelatability (from a global point of view the term “creating anonymity and unrelatability” is a contradiction in terms, because you can not withdraw any of the attackers information so that all measures for keeping anonymity and unrelatability can be kept (cp. §5.1.2)). I will classify these mechanisms into the highest sublayers of layer 3 resp. 1 because

- the MIX-Network uses the routing of (the deeper sub-layers of) layer 3, but which is no end-to-end measure;
- the DC-Network uses the transmission of tokens (bits) on layer 1, but it is not involved in discovering transmission errors, multiple access or other services of layer 2 (data link layer)<sup>18</sup>

RING- and TREE-Network - like all communication networks working with the principle of “unobservability of bordering connections and stations as well as signal regeneration” - create anonymity within the network by using a ring- or tree like media in layer 0 and digital signal regeneration in layer 1.

From this follows that broadcasting as a basic measure for protecting the receiver

- can use any desired implementations of layer 0, 1, 2 without efficiency drawbacks, as long as the implementation does not take place in a communication network not designed for broadcasting, or
- can use only an implementation of layer 0 if the implementation took place in a communication network designed for.

---

<sup>18</sup>If a DC-Network is implemented on a media with multiple access (like the todays typical LAN) it has to have to be implemented on layer 2 or above.

## 5 Security in Communication Networks

The MIX-Network may use any desired implementations of layer 0, 1, 2, and deeper sub-layers of layer 3 without efficiency drawbacks, the DC-Network any desired implementations of layer 0, and deeper sublayer of layer 1.

All these (partial) layers can be implemented without any quality losses for anonymity, so that they all may be used as measures for improving performance and reliability.

All layers (within the communication network) above the layers that do create anonymity and unrelatability have to **preserve anonymity and unrelatability**. This means that upper layers do not exclude or relate any (or at least only a few) alternatives which make the anonymity and unrelatability enabling layers possible - seen from the attackers perspective<sup>19</sup>. Otherwise an attacker could utilize the commonly accessible knowledge about the upper layers to identify sender and receiver or to relate them to each other. For example, not only do the protocols for anonymous multiple access have to use the bandwidth of the shared channel of the DC-, RING, TREE-Network efficiently, but also keep the anonymity of the sender (as well as the unrelatability of sending events).

This has consequences for the standardization of the protocols of the layer, which may have not been recognized (or just ignored) by the responsible standardization board. If a standard does not contain a subset of functionality that keeps up anonymity (or unrelatability) it is useless for a communication network with verifiable privacy. Consequently such international standards may be rendered illegal in certain countries where the constitution or certain laws oblige privacy, and national standards may be illegal. Especially the first option and the fact that modifying standards may take very long, may seriously hamper the future of international communication. Besides the question remains whether people from other countries want to use communication networks that are easily observed.

It is necessary for the reasons explained above that protocols of the layers above the layers that create anonymity and unrelatability keep anonymity (or even unrelatability of the sending events) for the participants. There should be certain limitations for the service. Consider two people communicating by telephone. You can not hide from each other that sentences (or information units that are implemented as switched continuous bit streams or as packets that occur if there is something to listen to (TASI=time assigned speech interpolation [Amst\_83][LiFl\_83][Rei1\_86])) are related to each other. In other words: It is not possible to create more anonymity or unrelatability towards the participants than the service used provides. The demand that all events on a higher layer completely correspond to events to the layers below and that all events can be connected refers exclusively to unobservability of non participants. If the attacker is either a regular participant on its own or if he has the cooperation from certain participants, you may save some efforts. You do not need to try to deny a linkage of certain events or objects that are already connected to others by the service used.

It should be mentioned that the techniques for protecting the receiver, the MIX-, DC-, RING-, or TREE-Network can be implemented in *higher* (sub)layers as well, like on a

---

<sup>19</sup>In other words: This means, that no (at least not that many) alternatives, which are possible (and distinguishable by the attacker) in the layer creating anonymity and unlinkability, are excluded or linked by higher layers.

layer 3 (network layer). Distribution and MIXes would be placed on layer 4 (transport layer) which would require the inclusion of some additional functions like pathfinding. If overlaying sending would be implemented on layer 4, the anonymous multiple access would be needed to be implemented on a higher layer once more. The same holds true for RING- and TREE-Networks where an additional encryption between the units on layer 4 would be necessary to simulate physical unobservability.

Therefore, distribution, MIX-, DC-, RING- or TREE-Network can be considered as *virtual* concepts, that can be implemented on almost any layer desired. But an efficient implementation needs to avoid unnecessarily complex layering and double implementation of functions on different layers (cp. fig. 5.40).

Figure 5.40 also shows how the extension of the concept of “Unobservability of bordering connections as well as digital signal regeneration” with “overlaying sending” can be implemented: On a media with suitable topology (layer 0) and a (sub)layer 1 with digital signal regeneration, a further layer for overlaying of keys and messages is put on. Above that runs the anonymous multiple access.

As explained in [DiHe\_79, page 420] the implementation of the (sub)layers that create or keep anonymity and unrelatability (within the communication network) may not introduce features that can be used by an attacker to differ between alternatives that are undecidable in the abstract design. Examples of such useless implementations are:

- Modulation of the output of a MIX according to the random bit strings within the messages
- Direct output of the result of a modulo 2 adder where  $1 + 1 \neq 0 + 0$  and  $1 + 0 \neq 0 + 1$  if the analogue signal characteristics are regarded
- Pattern sensitive time jitter in a RING-Network

These and other errors as well as implementations avoiding them can be found in [Pfit\_89, §3.1 and §3.2].

#### 5.4.10 Comparison of strength and effort of RING-, DC, and MIX-Network

In comparing the strength and necessary effort of the shown techniques, the following chart is helpful. You will notice that a DC-Network can resist stronger attacks than MIX-Networks but the effort to deploy is significantly less.

The three different sections in the chart, which describe the DC network, correspond in regards to the attacker model and complexity. At the token  $\geq$ , it is exactly = with duplex communication. At the MIX-Network the growth value that is practically relevant arises by ignoring the very slow length growth of the messages through the necessary random numbers.

	<p>Unobservability of bordering connections and stations as well as digital signal regeneration</p> <p>RING-Network</p>	<p>DC-Network</p>	<p>MIX-Network</p>
attacker model	<p>physically limited</p> <p>information-theoretical</p> <hr/> <p>towards confidentiality</p> <p>information-theoretical, towards service yielding</p> <p>complexity theoretically limited</p> <hr/> <p>complexity theoretically limited</p> <ul style="list-style-type: none"> <li>• cryptographically strong</li> <li>• well examined</li> </ul>	<p>complexity theoretically limited;</p> <p>There are not any secure practical asymmetric encryption systems known against adaptive active attacks</p>	
Effort per station and bit	<p>Transmission</p> <p><math>O(n)</math> (precise: <math>\geq \frac{n}{2}</math>)</p> <p>Key exchange:</p> <p><math>O(k \cdot n)</math></p> <hr/> <p>Transmission:</p> <p><math>O(n)</math> (precise: <math>\geq \frac{n}{2}</math>)</p> <p>Key exchange:</p> <p><math>O(k \cdot n)</math></p>	<p>polynomial in <math>n</math> but impractical</p> <hr/> <p>Transmission:</p> <p><math>O(n)</math> (precise: <math>\geq \frac{n}{2}</math>)</p> <p>Key exchange:</p> <p><math>O(k \cdot n)</math></p> <hr/> <p>Transmission:</p> <p><math>O(n)</math> (precise: <math>\geq \frac{n}{2}</math>)</p> <p>Key exchange:</p> <p><math>O(k \cdot n)</math></p>	<p>Transmission inside user's interface area:</p> <p><math>O(k)</math> practically <math>\approx 1</math></p> <p>Transmission inside the whole network:</p> <p><math>O(k^2)</math> practically <math>\approx k</math></p>

$n = \text{participants}$      $k = \text{coherence of keygraph and DC-Network, amount of MIXes}$

## 5.5 Upgrading towards an integrated broadband network

The upgrade of this narrow band towards a **broadband Network** should coincide with the level of population in the area (cp. fig. 5.41).

If there are already cable funnels in the area or if it is quite a rural area and if PRNGs with appropriate speed and security properties are available it should be less expensive to keep the cable structure. The distribution networks would be executed with overlaying sending even if PRNGs are too expensive. If there are no cable funnels in the area and if there are no PRNGs with appropriate properties, then RING-Networks should be implemented. If none of the conditions above are true and if there is an immediate necessity to act, a RING-Network should be installed for the time being despite the higher expenses. Overlaying sending can be installed later on.

As times goes by the amount of distribution networks executed with overlaying sending can be increased. Until Germany has a full featured distribution network, MIXing cascades should be used for highly sensitive narrow band applications not only in the local replay stations but also in the main (supra-regional) replay stations.

Especially in large cities where the local switching center has a high density of ports (mainly used by businesses) the technology will switch from copper cable to fiber optic cable soon. This and the higher density of local switching centers, as well as the fact that there are also some regional offices in the city, makes it possible to cascades even over these regional offices.

Parallel to the expansion there also has to be an increase in reliability. With that, fault tolerance measure will come increasingly into play.

Finally, it shall be mentioned that it is mainly unclear how to relate *costs* and *efficiency* between the network expansion shown here and the planned one.

### 5.5.1 Telephone MIXes

The following technique, called **Telephone MIXes**, allows the participants to create **unobservable connections** (unobservable end-to-end channels). Unlike MIXing cascades, the unobservable connections may exist forever and can be shutdown by both partners.

We will mainly focus on the connections between the station interfaces. The connections on top of this are only regarded if they differs from ISDN (144 kBit/s full duplex).

Telephone MIXes do not require alteration of the long distance network and allow a step-by-step extension (so you may use both, analogue and digital local relays).

Narrow band applications that can be run over ISDN's 16 kBit/s  $D_0$  channel are only covered in passing.

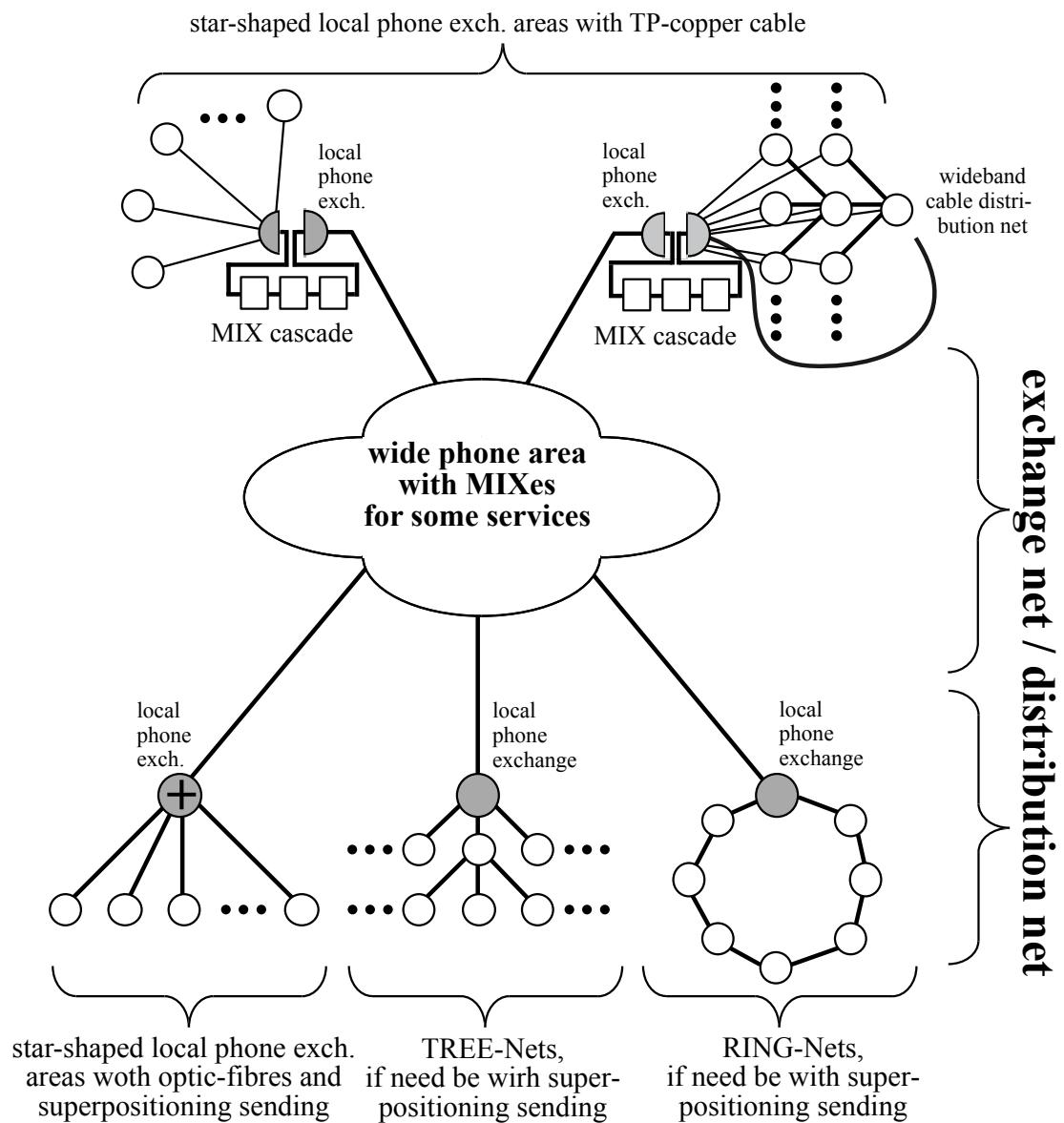


Figure 5.41: Integration of different security measures into one network

In contrast to earlier sections, here we will work with a hierarchically ordered network with local and long distance networks.

In §5.5.1.1 and §5.5.1.2 the basics of telephone mixes and of time slice channels are covered. §5.5.1.3 and §5.5.1.4 look at the establishing and down shutting of connections. In §5.5.1.5 we will take care of network accounting, in §5.5.1.6 of connections between local switching centers with MIX cascades and those without. Finally in §5.5.1.7 we will talk about reliability.

### 5.5.1.1 Basics of time slice channels

As seen, we could solve the problems in §5.4.6.9 about MIX channels by splitting long enduring connections into some short ( $\approx 1$  s) consecutive MIX channels, so called **time slice channels**. With every new time slice every participant has the opportunity to create new connections or to shut down existing ones.

Let every time slice channel have the duration  $d$ . The splitting into time slices is done synchronously for all participants of the whole network. As in §5.4.6.9, a time slice duplex channel consists of a sending and a receiving channel, short **TS channel**, and **TR channel**. For every time slice the network interface establishes a constant number of TS and TR channels (therefore a pair of 64 kB/s channels each).

A real connection is introduced by a **connection request message** that is sent from interface  $Q$  (reQuest) to interface  $S$  (reSpond). In the ideal case where  $S$  immediately accepts the request and where both stick to the protocol,  $Q$  and  $S$  connect each others TS and TR channels starting from the time slice previously agreed upon. The channels are disconnected synchronously when one of both wants to relieve the connection (further channels request messages for the single time slice channels are not needed if a proper request connection message is used; cp. §5.5.1.3).

As long as an interface has no real connection, it sends random data to itself by connecting its TS channel to its TR channel. Since meaningful data is end-to-end encrypted, it looks like random data too for the outside observer. Alternatively, you could leave the time slice channel open (unconnected; as in §5.4.6.9). But then  $M_m$  could distinguish it from a real connection. This is also a reason for establishing each part of the duplex channel on its own.

The necessary signalizing is almost done as in §5.4.6.9, over a separate signal channel. This channel needs to be accommodated in the part of the bandwidth not used for data channels (with ISDN it is about 16 kbit/s). This signal channel will be the main bottle neck for this technique because the connection request as well as the TS and TR channel establishing messages have to be transmitted through it (additionally TS has to be distributed).

So in contrast to ISDN, parts of the signalizing that is necessary after initializing take place within the data channel (but only that much that the data channel stays transparent). The signalizing for establishing is kept exclusively to the signal channel.

Since the user's narrow band connections and the long distance network should remain untouched for telephone MIXes, this technique can only be used in *local areas*. Therefore

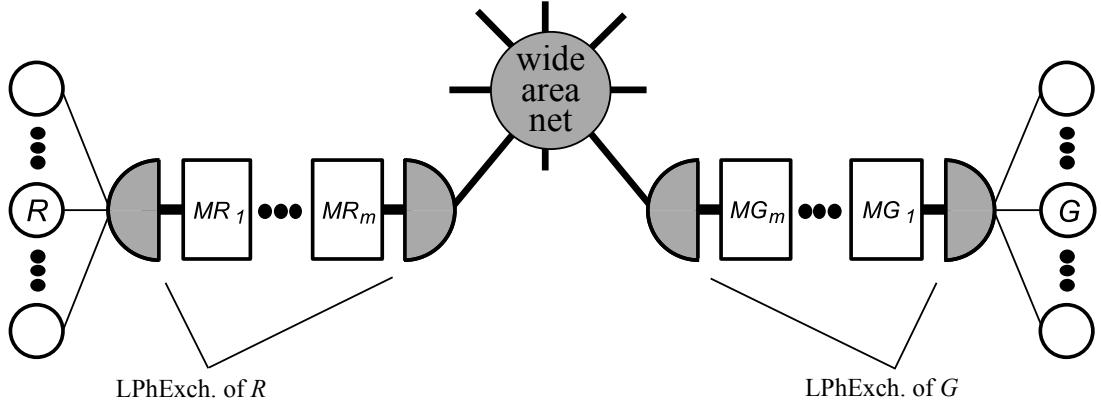


Figure 5.42: Interfaces of caller and callee with respective local switching centers (divided in two functional parts) and attached MIX cascade

every local switching center (LSC) is assigned one MIX cascade with  $m$  MIXes.

The assignment of tasks between the LSCs and the mixes is pretty easy: The MIXes are exclusively responsible for the MIXing functions, all other tasks are done by the LSCs. These tasks can be roughly split into two parts: The first one deals with the communication between the participants and the first MIX of the cascade (as well as the task of distributing of §5.4.1 (cp. [PfPW1.89, §2.2.3])). The other deals with the communication between the last MIX and the long distance net (as well as some transformations, cp. §5.5.1.6).

With larger LSCs ( $> 5000$  participants) and the limited bandwidth of the end user interfaces it will become necessary to separate the participants into logical anonymity sets once for all, so that mixing and distribution is only done within.

The traffic generated by dummy connections (TS and TR are connected to each other, but on the same device) does not hinder matters that much, because it only affects the owners interface. Besides, the dummy connections are only used if no other TS or TR connections are up.

Next we will have a look at if  $Q$  and  $S$  belong to different LSCs. Let  $MQ_1, \dots, MQ_m$  be the cascade of  $Q$  and  $MS_1, \dots, MS_m$   $S$ 's (fig. 5.42).

The special case that one interface is directly connected (without mixing) to the local switching center is examined in §5.5.1.6.

### 5.5.1.2 Structure of the time slice channels

Time slice channels only differ from MIX channels (cp. §5.4.6.9) in their predefined duration as well as the predefined starting point of time. But you need to take care of the influence of the long distance net on the structure and the beginning of the time slices.

## 5.5 Upgrading towards an integrated broadband network

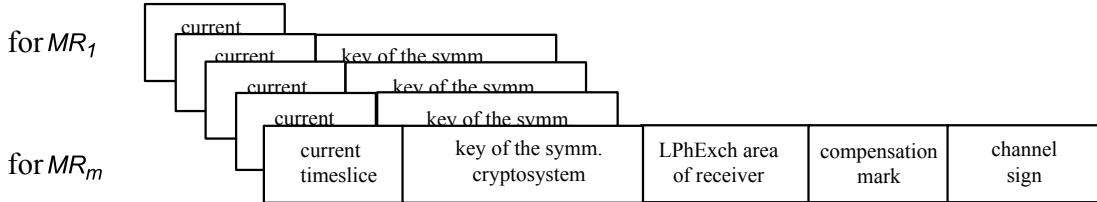


Figure 5.43: Data needed by MIXes  $MR_i$  for TS channel establishment

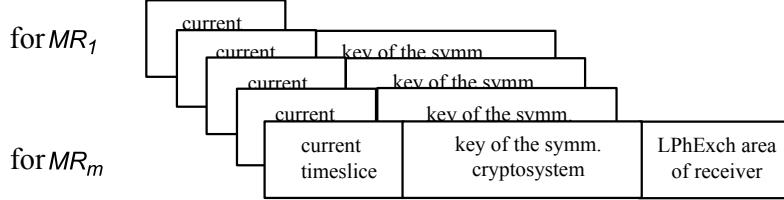


Figure 5.44: Data needed by MIXes  $MR_i$  for TR channel establishment

If  $Q$  and  $R$  are connected to the same LSC,  $MQ_m$ , and  $MS_m$  have to work together mutually as last MIX  $M_m$  in §5.4.6.9:

A TS channel from  $Q$  to  $S$  is passed from  $MQ_m$  over the LSC of  $Q$ , the long distance net and the LSC of  $S$  to  $MS_m$ .  $MS_m$  then connects it to the corresponding TR channel of  $S$ . To accomplish this  $Q$  needs to propagate the LSC of  $S$  in the TS channel establishing message  $MR_m$ .

Since the connection between  $Q$  and  $S$  is made up of many time slice channels it is useful to use the necessary channel in the long distance net for all connections between  $Q$  and  $S$ . The relation between different time slice channels of the connection does not need to be created explicitly for  $MQ_m$  and  $MS_m$  and the LSC; it is sufficient that the channel in the long distance net is shut down if there is *no* time slice channel to be transmitted between the LSC of  $Q$  and  $S$ .

As usual a MIX cascade has to wait with its TR channel of the current time slice until all the beginnings of the corresponding TS channel have arrived.

Therefore  $MQ_m$  and  $MS_m$  have to buffer the quickly incoming TS channels from the same LSC until the TS channels from the long distance net arrive (with a maximum latency of 0.2 seconds in the long distance net, this would be 12.8 kbit/s per local 64 kbit/s simplex channel).

Figure 5.43 and 5.44 show the data needed by the MIXes for establishing a channel. The keys of the symmetric concealment system are passed by the hybrid encryption system. And the remaining data in the  $D_i$  fields of the channel establishing message. The field "compensation mark" is explained in §5.5.1.5.

### 5.5.1.3 Establishing an unobservable connection

To establish an unobservable connection, interface  $Q$  sends a **connection request message** over its signal channel. The message is passed with the scheme from §5.4.6.2, therefore it is mixed with the cascade  $MQ_1, \dots, MQ_m$  and, if necessary, sent over the long distance net to the LSC of the receiver and distributed there to all interfaces.

Because with every new time slice the participant should be able to establish a new connection, the connection request messages have to be mixed once per time slice too. Let us assume that every interface contributes exactly one, if necessary senseless, message per time slice and channel.

The interface  $S$  needs to know the number  $t_0$  of the first time slice indicating when the connection becomes created.

The channel request messages of the single time slice channels (here two per time slice) are combined and integrated into the connection request message:

- $S$  gets the fixed local network identifier of  $Q$
- The channel identifiers of the time slice channels are created with a PRNG through which  $Q$  sends the initial value  $s_{QS}$  to  $S$ .  
If  $s_{QS}(x)$  marks the  $x$ -th identifier derived from  $s_{QS}$ ,  $s_{QS}(2 \cdot t + 1)$  is used for the time slice  $(t_0 + t), t = 0, 1, \dots$  from  $Q$  to  $S$  and  $s_{QS}(2 \cdot t + 2)$  for  $S$  to  $Q$  communication.
- $Q$  and  $S$  are using one single key for all time slice channels for conducting end-to-end encryption (which nobody except for  $Q$  and  $S$  can take notice of).  $k_{QS}$  can be derived from  $s_{QS}$  too.

Alternatively, you could send a  $k_{QS}$  independent from  $s_{QS}$ , or you could use different initial values for each direction. But this would only unnecessarily increase the length of the connection request message to be *distributed* by the LSCs.

If all is working properly,  $Q$  establishes the TS and TR channel as announced, beginning with time slice  $t_0$  with the identifiers derived from  $s_{QS}$ .

In the ideal case the channel between the interfaces  $Q$  and  $S$  are connected with  $S$  establishing its own TS and TR channel beginning with  $t_0$ .

In contrast to ISDN, where the whole signal traffic takes place outside the 64 kbit/s data channel, here it seems useful to utilize the yet unused data channel for the further signalizing of the connection buildup (as it is done on analogue telephone systems):

First, the establishing of a connection from  $S$  is signalized by a **starting message** to  $Q$  (who may also send such a message to  $S$  if a connection this way is needed by  $Q$ ; to prevent the MIX from sending valid a starting message into an unconnected TR channel they must have an appropriate length).

Next, the connection between the terminals can be established. This means that you now can use the usual telephone services: make the peer's telephone ring, the peer's telephone responds and so on. The necessary signalizing for the transition from "waiting for partner" and "partner is online" is done via the yet unused data channel.

## 5.5 Upgrading towards an integrated broadband network

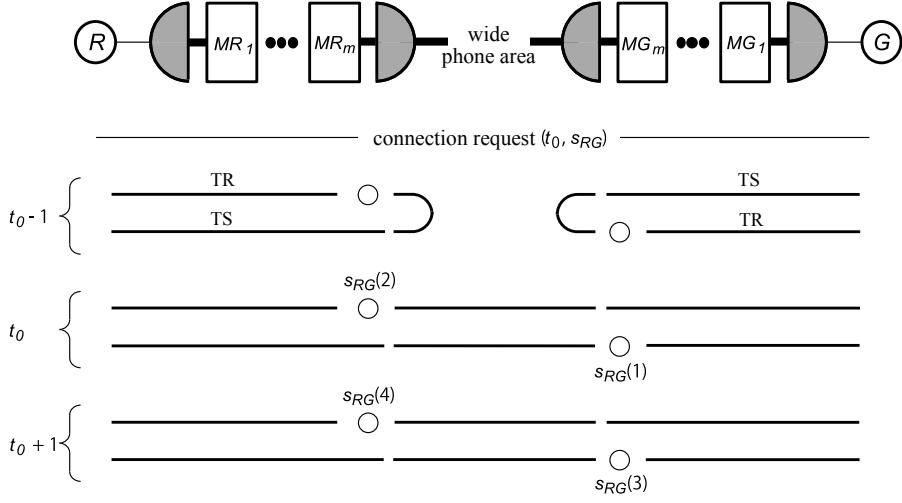


Figure 5.45: Connection establishment in the simplest case. The sending of start messages, i.e., the original start of the connection, is not shown

If  $Q$  and  $S$  do not synchronize their time slices (if  $S$  does not have an unused data channel) it does not affect the unobservability because the gap resulting from a missing TS channel is filled by at least one of the MIXes. TS channels without a corresponding TR channel are ignored by the receiver's last MIX.

So establishing connections can be completely asynchronously. For this  $S$  must save all incoming connection requests at least for some time.

If  $S$  is blocked (all bandwidth is used up by current connections that the user will not want to terminate) or not ready to receive,  $Q$  may have to wait on  $S$  for some length of time. To avoid that  $S$  must create a channel to  $Q$ 's LSC over the long distance net some time after  $Q$  gives up calling, the maximum waiting time of  $Q$  is limited. If the caller can pick any desired limit and send it with the connection request messages to  $S$ , then we can leave out the technique of the telephone MIXes to constantly repeat sending connection request messages over the signal channel. This will save the load of permanent redialing. With longer time-out limits the user may not necessarily wait at the terminal, but wait for a signal from the terminal if the connection is established. If  $Q$  does not wait (the connection establishing is canceled) and  $S$  tries to establish its part of the channel without the starting message from  $Q$ ,  $S$  will cancel the establishing process as well.

If  $S$  waits very long until satisfying the connection request by  $Q$  it needs to create all identifiers prior to the current one. But instead of a linear arrangement you could use a dendriform (cp. [GoGM\_86]) so that you can create every identifier  $s_{QS}(x)$  in  $\log(|x|)$  steps instead of  $|x|$  steps. If there is enough memory to completely save one branch of the tree each, you will need an average of less than two steps for sequential creation. This means that in the general case the effort is doubled.

If one of both interfaces waits for the other and if both are not connected to the same

LSC, in the scheme above the long distance net would be unnecessarily stressed by unused data channels. This can be avoided with a small reduction of the unobservability by enabling the last MIXes each (resp. the LSCs) to recognize the relation of all time slice channels of one connection within the long distance net.

For this  $Q$  and  $S$  are using a *common* identifier for all of their TE and TR channels of their connection in case of a distance call.

The (not that big) disadvantage is that the exact duration of the single distances calls is observable. If there have been only a few connections between two LSCs for a longer period this is also true using the technique mentioned before: An attacker can assume with a high probability that all consecutive time slices are exactly one connection.

But the advantage is that the last MIX in the sender's cascade can suppress meaningless data in the long distance net. For this the starting messages of  $Q$  and  $S$  are transmitted so that  $MQ_m$  and  $MS_m$  can identify them. Only once the starting message from  $S$  has arrived does  $MS_m$  establish the channel to the long distance net. The TE channels are filled by  $MQ_m$  and  $MS_m$  with meaningless data until the arrival of the proper TS channel.

If the connection is not immediately used for data transmissions, perhaps because  $S$ 's response for readiness takes some time, here also meaningless messages can be suppressed. For this the signal "connection to *participant* ready" has to be transmitted in plaintext (as with the starting message) to the last MIXes  $MQ_m$  and  $MS_m$ .

Connection request messages have to be *distributed* inside the local network. But this makes it possible to flood a local network with connection requests with a coordinated attack by many participants. Eventually connection requests of other participants can not be sent any more (*denial of service attack*). This kind of attack is possible in every multi-layer network like ISDN. Because the attacker(s) will not do anything illegal the attack can not be prevented but only detected and the outcomes minimized.

The latter can be achieved by adding priority flags to the connection requests which only allow a limited number of requests of higher priority for every participant. The distribution of connection requests is done by sorting the priorities. The assignment of priorities to connection requests is done adaptively by every network interface: At the beginning every connection request gets the lowest priority. If the connection request is not answered the request will get the next higher priority available.

The correctness of the amount of connection requests with higher priority per participant is checked by the LSCs through a one-time "authorization mark". Such a flag is a digitally signed (by the LSC) message which indicates the priority. Every participant gets a certain amount of flags for every higher priority for a certain amount of time (week, month). They can be distributed in periods with low network load. If a connection request gets no admission flag it will be assigned the the lowest priority.

But now the LSCs could easily identify the sender by the admission flag used. To avoid this the signing of the flags is done with **blind signatures** [Cha8.85][Chau.89]. They enable the participant to convert the admission flag exactly once so that it is still signed by the LSCs but not obvious to whom (cp. §3.9.5 and §6.4.1).

Finally a connection request message for  $MQ_i$  has a field  $D_i$  with:

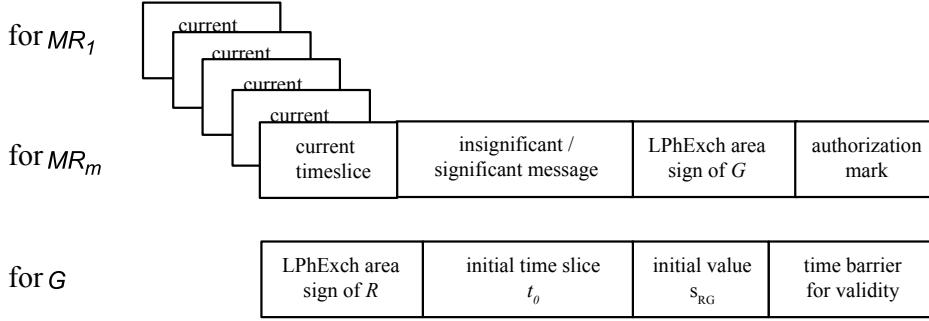


Figure 5.46: Connection request message of  $Q$  for  $S$

- A current time stamp (the number of the current time slice)

and additionally for  $MQ_m$ :

- A hint if the connection request message is meaningless and can be ignored
- The local network identifier of the receiver
- The admission flag

The components of a connection request message are shown in Figure 5.46.

#### 5.5.1.4 Shutting down an unobservable connection

A connection can be shutdown by both the partners. Shutting down is done by one partner disconnecting the TS and TR channels from a certain time slice on and having them “short circuited” or connected to other partners.

There are two main options for how a partner signals a shutdown request to the other by keeping transparency.

On the one hand, the requesting interface can send a shutdown request across the signal channel. But since the signal channel is quite stressed this “natural” solution does not seem so wise.

On the other hand, you could assign parts, precisely one 1 bit per time slice, of the signal channel to the data channel. So this one bit per time slice would be transmitted over the TS and TR channels. The shutdown request could be signaled by sending a “1” on this channel.

#### 5.5.1.5 Accounting the network usage

The techniques mentioned in [PfPW\_89] for accounting network and service usage without identification can be applied here as well.

For sessions within the same local network, accounting for single sessions seems technically senseless because they do not produce any additional load in the network. Besides, those sessions are completely unobservable.

However if there has to be per-usage accounting this can be done by local accounting devices (which are read by the carrier out in fixed intervals) for every participant's network interface by keeping total unobservability.

If unobservability is limited (by the carrier; naturally there is total unobservability) you may use the same accounting techniques as on long distance nets.

To account for sessions using the long distance net (and producing real costs only here) you could add compensation marks into the channel establishing messages for each of the last MIX (just like the authorization marks in §5.5.1.3). If an LSC can recognize the relation of all time slices of a long distance call (cp. §5.5.1.3) so it is possible as before, that a call has can only be made when both sides answer the phone. So it is sufficient to use the TS channels for payment. Besides you do not have to pay for every single time slice.

In Figure 5.43 the data necessary for accounting was assigned arbitrarily to the TS channel establishing messages. For TS channels that do not have to be routed over the long distance net the field for the hire ticket is empty (but of course it exists).

The hire tickets can be bought by the participant with a digital payment system keeping unobservability (cp. §6 and [PWP\_90][BüPf\_87][BüPf\_89]). The necessary message exchange runs over a normal unobservable connection.

If you assume that all time slices of a connection over a long distance net have share a common identifier either the caller ( $Q$ ) or the called one ( $S$ ) may pay the hire: It can be denoted in the connection request message, sent by the caller to the callee and  $MQ_m$ , who is to pay for the connection.  $MQ_m$  tells  $MS_m$  who is to pay. After exchanging the starting messages that do signal the establishing of the connection between the two interfaces, the last MIX of the one to pay will now only accept TS channel establishing messages with valid hire tickets. Otherwise the connection is shut down.

Instead of the last MIXes the LSCs of  $Q$  and  $S$  could take over the accounting if they get the necessary data from the MIXes (connection identifiers, who is to pay, hire tickets).

### 5.5.1.6 Connections between LSCs with MIX cascades and conventional LSCs

For connections of LSCs with MIX cascades and conventional LSCs, the last MIX plays the role of the translator.

If only  $Q$ 's LSC has a MIX cascade,  $Q$  tells the MIX  $MQ_m$  the number of  $S$  (along with the data of the normal scheme:  $s_{QS}$  and the number of the first time slice) with a special connection request message. Then  $MQ_m$  establishes a conventional connection to  $S$  over his LSC and the long distance net and connects it with the time slice channels of  $Q$ .

The calling participant ( $Q$ ) is as unobservable as if  $S$  had a MIX cascade. The participant called ( $S$ ) is as observable as when using today's ISDN.

If only  $S$ 's LSC has a MIX cascade then the main question is about addressing.

If the calling participant uses a conventional call number, the unobservability of the participant called is lost. That is why you should provide implicit addresses for those connections as well.

Soon you will see that an anonymity set will consist of about 5,000 participants. If you assume that every participant uses about 50 different open implicit addresses (cp. §5.4.1.2) and that the amount of possible addresses grows quadratically with the amount of addresses that are actually needed (to prevent collisions), the address space is about  $6.25 \cdot 10^{10}$ . So every implicit address would consist of 11 (decimal) digits. Additionally you will need to select the anonymity set of  $S$ . The size of the local network would never exceed 50,000,000 interfaces. If the anonymity set has 5,000 participants, 4 digits would do it.

Therefore the following scheme seems adequate:  $Q$  chooses the pre-selection number of the local network of  $S$  and establishes a conventional connection to  $MS_m$ . With a further 15 decimal digits it tells  $MS_m$  the implicit address of  $S$ .  $MS_m$  acts as a proxy and sends the corresponding connection request message to  $S$ . If  $S$  accepts the connection  $MS_m$  concatenates the conventional and the protected connection. If  $S$  refuses  $MS_m$  offers to keep the connection request until  $S$  calls back (by accepting the connection). This will prevent permanent redialing.

In contrast, a hidden implicit address will be about 200 digits long and requires some cryptographic abilities by the caller: The caller will need a computer to save keys and for calculating addresses. Furthermore, he must be able to send data from this computer to  $MS_m$  either with ISDN or with modem over analogue connections. The transmission is done within the connections of  $Q$  and  $MS_m$ , therefore not in the signal channel.

It shall be mentioned that an LSC having a MIX cascade can keep conventional participant interfaces.

### 5.5.1.7 Reliability

Obviously the cascade is a *sequential system* in terms of reliability. But here it would be sufficient to make every MIX's internals more fault tolerant. Even if a MIX failed, the interfaces connected to the LSC could leave out the MIX in their messages.

There are also fault tolerance techniques for more than one MIX, cp. §5.4.6.10 and [Pfi1.85][MaPf.87][Pfit.89, §5.3]. But they do increase the effort in general and the problem with untraceable return addresses which makes these techniques necessary does not occur here.

In regards to the protection from transmission errors there is no difference from ISDN. Here too a synchronous but unreliable transmission is assumed.

By using a proper stream cipher (Pseudo one-time pad using OFB) an *additional* propagation of bit errors on the 64 kbit/s channels is prevented. On the signal channel protocols like HDLC are being used to secure the transmission against errors.

## 5.5.2 Effort

Next, we will estimate the practical effort and capabilities for the telephone MIXes shown in the previous section.

It will turn out that if limited to  $2 \cdot 64 + 16$  kbit/s duplex channels for  $m = 10$  MIXes you may connect about 5000 interfaces to MIXing cascade.

As main bottlenecks two things will show up:

- The transmission of TS and TR channels establishing messages, connection request messages, and replies from the interfaces to the LSCs as well as
- the distribution of the connection request messages to the interfaces.

Both must be done using the 16 kbit/s signal channel. Since a certain fraction of the bandwidth has to be used for error correction and sometimes even for narrow band services we will assume an effective duplex bandwidth of  $b = 12$  kbit/s.

The preparation and processing of the messages by the interfaces, the MIXing, and the data channels itself are only minor problems.

In §5.5.2.1 some parameters for describing the requires cryptographic systems as well as some typical values are introduced. In §5.5.2.2 the duration  $z$  of a time slice is estimated, and in §5.5.2.3 the amount  $n$  of interfaces that may be connected to the MIXing cascade. In §5.5.2.4 we will have a look at the complexity of the tasks the interfaces and the MIXes have to perform per time slice. Finally in §5.5.2.5 we will estimate the duration of the connection establishing.

As a reminder here is the message scheme (slightly transformed) again:

Suppose a participant wants to send a message  $N$  (already end-to-end encrypted) from his interface  $A$  to the interface  $B$  of another participant. Additionally he wants every MIX  $M_i$  to receive some certain data  $D_i$ . Then he consecutively creates  $m + 1$  messages

$$\begin{aligned} N_{m+1} &:= N \\ N_i &:= c_i^*(D_i, N_{i+1}) \text{ for } i = m, m-1, \dots, 2 \\ N_1 &:= k_{1A}(D_1, N_2) \end{aligned} \tag{5.1}$$

and sends  $N_1$  to  $M_1$ .  $N_1$  is named **MIX Input message**.

Every  $M_i$  receives the message  $N_i$  from  $M_{i-1}$ ; the transcoding of  $N_i$  consists of the decryption with  $k_{1A}$  resp.  $d_i$  as well as the removal of  $D_i$ .

For  $M_1$  you could use a hybrid cryptosystem instead of symmetrical. But this would only result in bigger efforts needed to create  $N_1$  and it would reduce security in certain cases.

### 5.5.2.1 Parameters for describing the cryptographic systems needed

The parameters mentioned here are oriented on the most known systems, both symmetric and asymmetric systems like DES and RSA (cp. §3.7 and §3.6).

Parameter	symmetric stream cipher	asymmetric block cipher
Key length	$s_{sym} = 128\text{bit}$	irrelevant
Block length	$b_{sym} = \infty$	$b_{asym} = 660\text{bit}$
Length of encryption of $N$	$ N $	$b_{asym}$ (if $ N  \leq b_{asym}$ )

Figure 5.47: Parameters of the cryptographic systems used

It is pointed out (cp. §5.4.6.8) that the security of these cryptographic systems and all MIX implementations basing on is not even *proven* towards relatively hard problems like the factorization assumption.

This disadvantage is ignored here only because, on the one hand, this applies to all practical concealment systems and on the other hand, because DES and RSA are well examined and have not been broken publicly.

The parameters and their assumed values are shown in Figure 5.47.

### 5.5.2.1.1 Symmetrical Concealment Systems

For the time being we will assume a secure block cipher with 128 bit key length like the generalization of DES with alternative key generation (cp. §3.7.7).

For the techniques shown here we will need a synchronous stream cipher, therefore a block cipher that runs in OFB mode (cp. §3.8.2.4). So there will be no expansion due to block padding or something.

### 5.5.2.1.2 Asymmetrical Concealment Systems

For the asymmetrical cryptographic system RSA a modulus of 660 bits seems sufficient (cp. §3.6).

### 5.5.2.1.3 Hybrid Encryption

To save bandwidth the hybrid encryption always tries to take as much as possible of the plaintext  $N$  into the first, asymmetrically encrypted block (denoted  $N'$ ). Therefore if  $N''$  is the rest of  $N$

$$c_E^*(N) := \begin{cases} c_E(N) & \text{if } |N| \leq b_{asym} \\ c_E(k_{SE}, N'), k_{SE}(N'') & \text{else} \end{cases} \quad (5.2)$$

To determine the length of  $c_E^*(N)$  you need to make the encryption *indeterministic* (cp. §5.4.6.2) to keep unobservability.

With using RSA first of all you need to remove the determinism by indeterministic encoding the message. Basically this is done by adding a certain amount  $a$  of randomly picked bits.

The resulting *length expansion* can be reused by making those  $a$  bits the symmetric key  $k_{SE}$ . So  $a = s_{sym}$ .

Unfortunately the exact implementation of the “expansion” is not that trivial. It was shown that if RSA is used, the encoding by pre-pending random bits like in [Chau\_81] results in an insecure MIX-Network (cp. §5.4.6.8).

But still secure is the following expansion: Prior to the encryption of a message the encoder picks a random,  $s_{sym}$  bits long string and concatenates it with the message. The resulting message is applied to a permutation  $\pi$  that “removes” the message’s structure as well as possible. The permutation  $\pi$  and its inverse  $\pi^{-1}$  are commonly known and be defined by a fixed key of a symmetric concealment system. Encoding goes the other way round by applying  $\pi^{-1}$  first and removing the  $s_{sym}$  bit long string next.

If you consider this expansion then:

$$|c_E^*(N)| = \max\{b_{asym}, |N| + s_{sym}\} \quad (5.3)$$

### 5.5.2.2 Minimal duration of a time slice

Every interface has to send three MIX messages for every time slice and every available 64 kbit/s data channel: a TS and a TR channel establishing messages as well as a connection request message. The lengths are denoted with  $L_{TS}, L_{TR}, L_C$ .

The length of the reply (cp. [PfPW1\_89, §2.2.3]) is denoted with  $L_R$ . If RSA is used as signature system in general  $L_R = b_{asym}$ . Since the replies do not carry weight due to their short length it is assumed that every interface sent a reply for every time slice.

Therefore an interface needs to be able to send  $2 \cdot (L_{TS} + L_{TR} + L_C) + L_R$  bit over its signal channel while every time slice so:

$$z \geq \frac{2 \cdot (L_{TS} + L_{TR} + L_C) + L_R}{b} \quad (5.4)$$

The lengths  $L_{TS}, L_{TR}$  and  $L_C$  are to be estimated next. In §5.5.2.2.1 the length of a MIX message is estimated in general, in §5.5.2.2.2 the specific values of 5.4 are used.

#### 5.5.2.2.1 Length of a MIX message

Due to §5.5.2 (5.1) is:

$$|N_1| \leq |D_1| + |N_2|$$

resp. for  $i = 2, \dots, m$  due to §5.5.2.1.3 (5.3)

$$|N_i| = \max\{b_{asym}, |D_i| + |N_{i+1}| + s_{sym}\}$$

Since for  $i < m$  the message  $N_{i+1}$  consists of at least one asymmetrically encrypted block it holds for  $i < m$  that  $|N_{i+1}| \geq b_{asym}$ . If you resolve the upper recursive equation

$$|N_1| = \sum_{j=1}^{m-1} |D_j| + (m-2) \cdot s_{sym} + \max\{b_{asym}, |D_i| + |N_{i+1}| + s_{sym}\} \quad (5.5)$$

In the following section we presume equation (5.5).

### 5.5.2.2.2 Estimation

The structures of the three message types are shown in Figure 5.43, 5.44 and 5.45. To the fields the following lengths will be assigned to:

Figure 5.43, 5.44, 5.46

- Time slice number: 30 bit
- Local network identifier: 22 bit

Figure 5.43, 5.44

- compensation mark for paying the long distance net usage: 670 bit
- Channel identifier: 28 bit

Figure 5.46

- Discrimination between meaningless/meaningful messages: 1 bit
- Initial value  $s_{RG}$  of the PRNG: 128 bit
- Time out limitation for validity (therefore the waiting of  $Q$  to  $S$ ): 20 bit
- Admission flags to denote the priority of a connection request: 670 bit

For both marks we assume that the priority of the authorization mark resp. the value of the compensation mark are not only specified implicitly by the signature (RSA: 660 bits) but also explicitly (with at least 10 further bits). This will spare the controlling LSC from trying all possible keys for validating the signatures.

For a TS or TR channel establishing messages holds for  $i = 1, \dots, m-1$  that each  $|D_i| = 30$  bits because the key of the symmetric concealment system was already counted for the hybrid encryption and  $|N_{m+1}| = 0$ .

For a TS channel establishing messages is  $|D_m|=750$  bits and after (5.5)  $|N_m|=750$  bits  $+s_{sym}$ . For a TR channel establishing messages  $|D_m| = 58$  bit and after (5.5)  $|N_m| = b_{asym}$ . From (5.5) follows:

$$\begin{aligned} L_{TS} &= (m-1) \cdot (30 \text{ bit} + s_{sym}) + 750 \text{ bit} \\ L_{TR} &= (m-1) \cdot (30 \text{ bit} + s_{sym}) - s_{sym} + b_{asym} \end{aligned} \quad (5.6)$$

For a connection request messages hold for  $i = 1, \dots, m-1$  that each  $|D_i| = 30$ , and  $|D_m|=723$  bits. After (5.5) is  $|N_m| = 723 \text{ bit} + |N_{m+1}| + s_{sym}$ . The other way round  $|N_{m+1}| = b_{asym}$  is sufficient since the receiver has to keep only 200 bits.

Then

$$L_v = (m-1) \cdot (30 \text{ bit} + s_{sym}) + b_{asym} + 723 \text{ bit} \quad (5.7)$$

After (5.6) and (5.7) is

$$L_{TS} + L_{TR} + L_v = (m-1) \cdot 90 \text{ bit} + (3 \cdot m - 4) \cdot s_{sym} + 2 \cdot b_{asym} + 1473 \text{ bit} \quad (5.8)$$

If you now insert the values (cp. fig. 5.47) for  $s_{sym}$  and  $b_{asym}$  and  $b = 12$  kbit/s into (5.4) then is

$$z \geq \frac{474 \text{ bit} \cdot m + 2191 \text{ bit}}{6000 \text{ bits/s}} + \frac{660 \text{ bit}}{12000 \text{ bit/s}} = 0.0079 s \cdot m + 0.4201\bar{6} s \quad (5.9)$$

wherefore  $m = 10$  a value of  $z \geq 1.22 s$  is sufficient.

### 5.5.2.3 Maximum amount of interfaces to be served by one MIXing cascade

The amount  $n$  of interfaces that can be served by one MIXing cascade is limited by the distribution of the connection request messages: for every interface there is only one receiving channel of capacity  $b/n$  available.

How strong this limitation is depends on the traffic situation. We simplify this by assuming that the network is a M/D/1-System where  $\lambda$  is the maximum average rate with which every participant receives connection requests. We will assume  $\lambda = \frac{1}{300} \frac{1}{s}$  so at peak loads there are twelve connection requests per participant and hour on average (This value for  $\lambda$  is surely sufficient if not even overrated: In accordance to long distance call statistics [SIEM\_87] in Germany from 1985, there are less than 3.1 calls per interface and day on average. If you trust the expectation of Siemens  $\lambda = \frac{1}{300} \frac{1}{s}$  is too high even for peak loads: According to [SIEM\_88] the most powerful connection switch from Siemens can perform 4.8 connection attempts per participant and hour. Besides if using Telephone-MIXes only the incoming calls are interesting towards  $\lambda$ ).

Every distributed connection request message consists of exactly one asymmetrically encrypted block (cp. §5.5.2.2) and therefore has the length  $b_{asym}$  so that there is a maximum of  $\mu := \frac{b}{b_{asym}}$  connection request messages per second to be distributed.

If  $T_q$  is the average system time, and therefore the time a connection request needs on average to get to an interface, then holds [Klei\_75, §5.5 especially (5.74)] in peak hours on average

$$T_q = \frac{2 \cdot \mu - n\lambda}{2 \cdot \mu \cdot (\mu - n \cdot \lambda)} \quad (5.10)$$

This results in

$$n = \frac{\mu}{\lambda} \cdot \frac{\mu \cdot T_q - 1}{\mu \cdot T_q - 0.5} \quad (5.11)$$

For  $\lambda = \frac{1}{300} \frac{1}{s}$ ,  $T_q \leq 0.5 s$ ,  $b = 12$  kbit/s and  $b_{asym} = 660$  bit is makes  $n \leq 5137$ .

Especially for connection request messages the usage of open implicit addresses (cp. §5.4.1.2) would be worthwhile: Since the called interface  $S$  identifies the calling interface  $Q$  by its open address  $a_{QS}$  the connection request only has to contain the number of the desired time slice. All other information can be reused by  $S$ . If the caller does not use its own interface  $Q$  but is calling from another local network the connection can be routed over  $Q$  to the current interface. If an address is 100 bit long the connection request will not be bigger than 130 bit. If only those types of addresses are used the limit is  $n \leq 27389$  due to  $\mu = b/130$  bit (cp. (5.11)).

The M/D/1 assumption is, as explained, somewhat simplifying. Although this does not matter: The arrivals are almost “state less” since a redialing by repeating the connection request message is not necessary. The service duration is indeed deterministic apart of some repeats due to errors. The available bandwidth does not need to be statistically limited. Besides the assumption that  $T_\nu \gg 1/\mu$  has very little influence on  $n$  so (5.11) is mainly the trivial claim  $n \cdot \lambda \leq \mu$ .

#### 5.5.2.4 Computation complexity for the interfaces and MIXes

The computation complexity for the interfaces and MIXes mainly consists of the cryptographic operations: Preparing and recognizing implicit addresses, Preparing and recognizing of MIX messages, signing and testing replies, encryption and decryption of data channels.

But since the encryption rates of both the concealment systems are much higher than the transmission rates those operations are not problematic.

The remaining effort for the interfaces like managing keys, implicit addresses etc. can be neglected.

The remaining effort for the MIXes is mainly about buffering and resorting of messages as well as the testing of time stamps. This affects 6 input batches per time slice with  $n$  messages each and therefore less than 27389 messages (cp. §5.5.2.3). This should be possible, with special hardware, within 0.01 seconds per MIX. To save some additional time you may configure the connection request messages so that it first establishes a sending channel to  $MQ_m$  resp. the LSC over which the additional data for  $MQ_m$  and  $S$  is sent. Through this the participant can decide, if and with whom he desires a connection, at some later time.

In general you need to take care that messages of the different time slices are processed pipeline-wise by the MIXes. So the time slices may begin on different MIXes at different moments.

#### 5.5.2.5 Connection startup duration

The startup duration mainly consists of

- waiting for the transmission of the connection request message by the MIX cascade of the calling interface ( $\leq z$ )
- the latency of the MIX cascade of the calling interface ( $m \cdot 0.01$  s)
- the latency of the long distance net ( $\leq 0.2$  s)
- waiting for the distribution to the receiver (during peak loads  $T_\nu$  on average)
- waiting for the establishment of the TS channels by the called interface ( $\leq z$ , as long as the called interface responds immediately)
- the latency of the MIX cascade of the calling interface ( $m \cdot 0.01$  s).

Summed up, this makes  $2 \cdot (z + m \cdot 0.01 \text{ s}) + T_\nu + 0.2 \text{ s}$ , with the value from above about 3.34 s. On average this duration will be halved.

## 5.6 Network management

First of all in section §5.6.1 we will discuss who (sets up and) runs which parts of the communication network (and then consequently is responsible for the service quality).

After that in §5.6.2 it is shown how accounting for usage of the communication network can be done securely and conveniently without undermining anonymity, unobservability, and unrelatability.

The payment for (higher) services provided by the communication network can be done with any digital payment system desired (cp. §6).

### 5.6.1 Operating a Network: Responsibility for service quality vs. threats of Trojan horses

In regards to operating a network and responsibility for service quality there is the following dilemma [Pf1\_85, pg. 129]:

If everybody uses an arbitrary user station (which you probably already guessed since nothing else was said explicitly) then no organization is willing (and able) to take responsibility for the service quality. This is true at least if mistakes or insufficient performance may influence the service quality of other user stations that do not even communicated with the faulty station (that is why we talk about network participants but not users beginning with §5.4). Such a “anarchistic-liberal” network management may be optimal for the validation (by the users) of anonymity, unobservability and unrelatability but is almost totally unusable for service integrated communication networks. But of course it can be run on any communication network desired for the price of less costly efficiency, less performance, less reliability, and more distrust through the rest of the society always asking what those participant may hide.

On the other hand if one organization designs, produces, installs, and runs all network components, especially the participant stations, the organization may be willing to be responsible for the service quality. But nobody, especially no participant, can rule out that parts of the organization may install Trojan Horses (cp. §1.2.2). Such an organization would be uncontrollable in terms of security and confidentiality.

Since every protection measure that is implemented with the communication network is realized by the participant stations, it is useful to split the stations regarding design, production, installation, and operation into two parts: The terminals and the interfaces.

The **participant terminals** are those parts that have nothing to do with communication with other stations. Consequently the terminal can only be under control by the participant.

The **network interfaces** are all parts of the participant station that come in touch with the communication with other stations. Consequently the carrier is responsible for

the network interface.

For example, the CCITT calls it Data Terminal Equipment (X.25 related) for our participant terminal and Data Circuit-Terminating Equipment for our interface (cp. [Tane\_81][Tane\_88]). With ISDN it is called Terminal Equipment and Network Termination [ITG\_87][Tane\_88].

As long as there is to be a carrier that is responsible for service quality the complexity of the interface should be kept as small as possible for better validation of security and confidentiality. Additionally, design, production, installation, and operation can be publicly checked. This would make it acceptable for anonymity, unobservability, and unrelatability to have only a few or only one vendor of interfaces whose quality is accepted by the carrier. The vendors have to publish the designs inclusive of all production aids and design criteria. Consumer organizations like “Stiftung Warentest” should continuously look for Trojans inside the interfaces as well as inside the participants stations. This is quite complex at the beginning if design and production aids and all designs have to be checked. But later on the control of the unmodified production is much easier.

Interfaces should consist of only a few integrated digital circuits and analogue sender and receiver circuits. Consequently maintenance or repair are quite seldom. And most importantly there should be necessity for *remote maintenance*, therefore no ability for the vendor or carrier to modify the software aboard the interface over the communication network. Today many telephone switches and other devices do have this ability since software failures have to be corrected immediately. As seen in §5 a vendor can install or remove Trojans remotely at will. But this does not hold true if remote maintenance is limited to *remote inquiry*: With this kind of access no software can be altered over the communication network but only existing diagnosis tools can be executed. The results of this is that a service technician can order the needed spare part and bring it along. But guaranteeing that remote maintenance is “read-only” is the common problem of excluding a Trojan Horse (cp. §1.2, §5).

In the next subsections we will describe which functions the interfaces have to offer for both the basic techniques for protecting transmission data and data on interest. Figure 5.48 gives an overview.

### 5.6.1.1 End-to-End encryption and distribution

Since faulty end-to-end encryption or decryption, generation, or recognition of implicit addresses or faulty receiving of distributed information units only affect the participant’s station and its communication partners this can be put into the participant’s terminal.

### 5.6.1.2 RING- and TREE-Networks

The interface of RING- and TREE-Networks have to have not only the physical interface with digital signal regeneration but also the access techniques and all the fault tolerance techniques needed. Because if these functions generate errors (reduction of performance or reliability) all other participants of RING- and TREE-Network are affected.

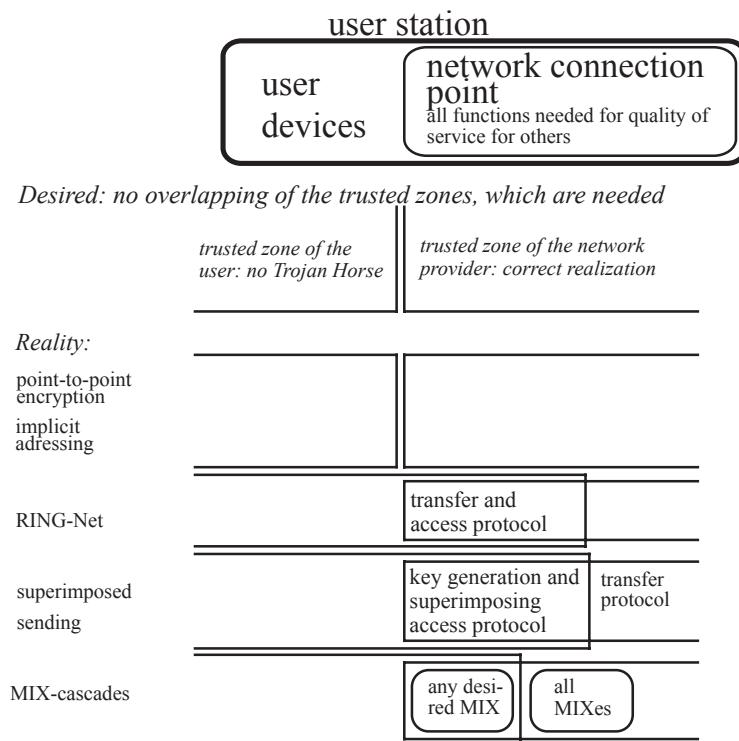


Figure 5.48: Functions of the interfaces

If there is a Trojan Horse in the interface of the carrier or the vendor so it may register the sending of the affected station which overrides the security measures of the transmission topology and the digital signal regeneration for the sender completely. With that the problem of the Trojan Horses was transferred from the LSC to the interfaces. Nevertheless, the situation is much better (cp. §5.6.1) since interfaces are much easier than LSC. This make a validation for Trojan Horses much easier. Additionally, such a validation must be performed much less often, because maintenance is less often needed.

### 5.6.1.3 DC-Network

Next to the physical network interface the interface of the DC-Network has to have the PRNG, the access techniques and all required fault tolerance measures. If those functions cause problems on performance and reliability on one participant all other participants of the network are affected too. If there are Trojan Horses in the carriers or vendors interface so it may register the sending of this station. And this would repeal the protection of overlaying sending. The problem of Trojan Horses was therefore only transferred from the switches to the interfaces. But as in §5.6.1.2, the situation has improved.

With the modular structure of the interface corresponding to figure 5.40 the modules that represent the physical interface (media and the lower sublayer of layer 1) are uncritical in terms of anonymity, unobservability, and unrelatability according to the attacker model of DC-Networks. Here anybody may undertake an observing attack. Even modifying attacks will not harm the anonymity with an appropriate action but only prevent the service. Appropriate protocols for error and attack diagnosis are shown in §5.4.5.6 and §5.4.7 as well as in [WaPf.89][WaPf1\_89].

For a large public DC-Network with a carrier it is even more necessary to prove the cryptographic strength (or to validate; cp. §3) of the PRNG than for a local or special network. Otherwise the PRNG may contain a hidden trapdoor which would enable the network carrier or designer to observe the sending of the participant stations. Or the pseudo random number generation, that has been a defacto standard on this DC-Network for years and is therefore highly expensive to replace, will finally be abandoned.

### 5.6.1.4 query and superpose as well as MIX-Networks

If servers or MIXes are run by normal network participants the interface involves a big database with adders or (small) switches and a larger number of very powerful decryption and encryption devices as well as all measures for fault tolerance. As explained in §5.4.6.7 you may try not only to secure the communication relation but also the sending and receiving of participant stations by MIXes. Next to the difficulties shown in [Pfit.89, §3.2.2.4] that either many meaningless information units or long latency have to be accepted, those voluminous interfaces can canonically observe the participant stations (but not their communication relations). This attempt is not only very expensive and inconvenient but its outcomes are also highly doubtful.

Though the switching, functions of the MIXes can be eliminated by forcing a fixed order for going through the MIXes (cp. §5.4.6.1). But also for everything else maintenance and repair could arise more often than is acceptable for devices stored in private rooms.

The maintenance and repair situation is completely unproblematic as long as (cp. §5.5.1) normal participant stations will not operate as servers or MIXes.

Since requesting and overlaying will not hamper the service quality of uninvolved participants through the “errors” of other participants, that do not run server functionality or whose servers are not affected by errors, the measure for protecting each single participant can be done exclusively on the participant terminal. Those measures are the creation of requesting vectors, the encryption and sending to the server, the receiving and decryption of messages from servers, as well as the overlaying.

Since the service quality of uninvolved ones in a MIX-Network will not be hampered by errors of others that do not run MIX functions or whose MIX functions are not affected, the measures for protecting each single participant can be done exclusively on the participant’s terminal. Those measures are the multiple encryption and decryption. With the technique shown in §5.5.1 of meaningless time slice channels and distribution of the connection requests sending and receiving can be protected much better than with the technique of §5.4.6.7 where every participant station is a MIX.

In the case of a large public MIX-Network, it is much more important than for local or special networks that the cryptographic strength of the concealment system used is proven in public or at least validated (cp. §3 and §5.4.6.8).

#### 5.6.1.5 Combination as well as heterogeneous networks

If - as seen in §5.4.4 - several techniques for protecting network and data on interest are combined it is useful (regarding the validation of the protection) not to realize an interface that has *all* the necessary functions but a cascade of all interfaces necessary for the different techniques, corresponding to the layering from §5.4.9. Then an interface corresponding to a lower layer of a remote terminal with a Trojan Horse can not observe the communication.

#### 5.6.1.6 Connected, hierarchical Networks

If the transitions between networks (resp. network levels and the upper layers needed by every network part from time to time) are realized in a manner that a group of operators can take over the responsibility for the service quality the remaining network parts can be designed at will. Carriership and responsibility for service quality can be therefore handled almost independently for each different network part.

## 5.6.2 Accounting

Within an open communication network a convenient and secure accounting of the carrier must be possible. While organizing the accounting you need to take care that anonymity, unobservability, and unrelatability in the communication network are maintained throughout the accounting data.

Basically there are two opportunities: **Individual** accounting per usage (or subscriber) where the paying participant is anonymous. Or the **general** accounting with a lump sum payment for every participant. This does not need to be anonymous since no interesting accounting data is created.

For individual accounting

- either non-manipulatable counters [Pfi1\_83, pg. 36f]
- or anonymous digital payment systems (cp. §6)

can be used.

The **non-manipulatable counters** are accommodated into the interfaces. They correspond to today's electricity counters with the difference that they may be accessed (read only) remotely. If the counters are configured so that they can be accessed in big intervals (once per month) they give only a few persons specific data. So they may be accessed without anonymity.

The main advantage of this solution is that there is almost no effort for accounting. The big disadvantages are that a circumvention of the counter or the interface as well as the manipulation have to be detected. But both disadvantages are not that fatal because (in contrast to traditional payment, cp. §6) you can only pay one service with these counters - and for private usage the amounts are quite small. Even today stamps can be much more easily faked than bank notes.

If using anonymous digital payment systems the accounting protocols have to be developed in a way that nobody can cheat. This is important since anonymity, unobservability, and unrelatability prevent subsequent penal processes [WaPf\_85][PWP\_90][BüPf\_90]. This can be easily achieved with the accounting problem: The first of all relatable information units get a prepended digital currency item. The value of this “**digital stamp**” is credited to the carrier before he continues to transport the relatable information units.

Even with local communication in a DC- or TREE-Network the carrier can suppress unprepated messages if he has global control resp. controls the root of the tree. With a RING-Network this is not always possible. Even though with two ring participant *A* and *B* either *A* can send to *B* without payment or the other way round. With transmission frames, executing a duplex channel is equalt expensive to executing a simplex channel, a service that requires a duplex channel can not be acquired without payment.

The main advantage of the “digital stamp” is that there are no un-circumventable and non manipulatable counters. And when using a “good” anonymous payment system no person specific data is created. The disadvantage is the increased communication effort.

To keep this small (especially the latency) the carrier should take on the function of the bank in the digital payment system (cp. §6).

If a participant demands an accounting verification either (depends on the implementation) the terminal or the network interface device creates it for him (and only for him).

By using general lump sum payments you can avoid all problems regarding cheating and almost all effort of the accounting system.

As soon as enough bandwidth is available a blanket payment for narrow band sending and receiving TV will be possible.

## 5.7 Public Mobile Radio

As a supplement to the discussion about open communication networks on fixed connections we will take a look at the communication between mobile participant stations [Alke\_88]. The main focus will be on how to solve the occurring data protection issues [Pfit\_93][ThFe\_95][FJKP\_95][Fede\_99].

The differences to the previous chapters are

- that the bandwidth is and will stay very limited since the electromagnetic spectrum can be occupied only once
- and that network data, data on interest, content are person specific but also the *current location* of the mobile station.

As seen in §5.4.4 it is unrealistic in terms of technically advanced attackers to demand that signals sent from different stations not be distinguished: Due to the analogue characteristics of the sender this is impossible ...

This is why it is assumed that a mobile station can always be tracked and identified if it sends<sup>20</sup> (even if engineers designed it to not be able to do so).

In contrast to this let us suppose that participant stations can be identified or traceable if they are receiving only.

Because radio transmission is easy to listen to you will need a link encryption between the mobile stations and the first fixed station parallel to the end-to-end encryption. But only if layers 1 to 3 of the ISO/OSI reference model create person specific data.

Due to the limitation of the bandwidth and a permanent ability to identify and track the mobile station, the techniques from §5.4.3 and §5.4.5 for protecting the sender are not applicable.

To force all measures for protecting the network and data on interest to take place in the location-fixed part of the communication network the following technique seems useful (cp. fig. 5.49):

---

<sup>20</sup>This is a matter of effort. In [BML\_89, pg. 297] you can read: "Computer controlled receiving and processing units enable the signal intelligence services to pick a certain device from a plurality of devices by its special characteristics".

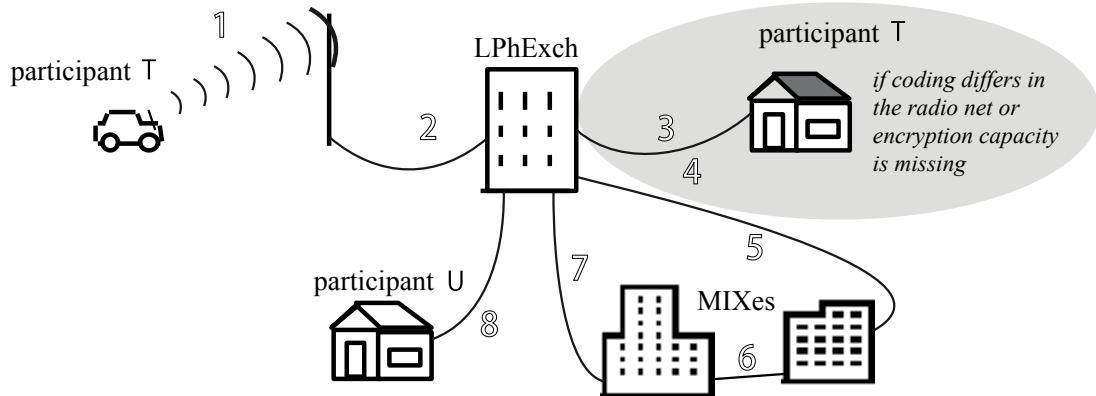


Figure 5.49: Connection of wireless users to a MIX network

As long as the encryption of the payload data is done by mobile as well as in the location-fixed participant station the technique of transcoding MIXes (cp. §5.4.6) can be employed as long as the mobile station has enough capacity to perform the cryptographic functions.

If the encoding of the payload data in the mobile stations is different from the fixed location stations, to save bandwidth for example (13 kbit/s vs. 64 kbit/s), the receiver and the communication network could use this for relating. In this case, at least as long as mobile stations are only a small fraction of all participant stations, there should be link encrypted channel to the next location fixed station. From there on you will have the usual techniques for protecting network and data on interest. This is also true if the encoding of payload data is equal to mobile and fixed stations but the mobile stations do not have enough encryption capacity.

A station that is only receiving should be able to be tracked by the communication network (even on cellbased systems). If there is a connection request or a long message for this station a corresponding implicit address should be distributed within the network. Then the station will reply actively and therefore become locatable. Since addresses only need to consist of a few bytes the effort for this protection measure is small. This may even lead to a effort reduction because the administration of the cellular systems is simplified.

A measure that is independent from the previous ones is to keep the user anonymous towards the sending station. This makes sense if sending stations are not firmly assigned to users like in the GSM Network. There every participant gets a chipcard (SIM = Subscriber Identity Module) with which he can use every sending station. An example would be a rental car where neither the network nor the sending station of the car know who is actually using it. This technique corresponds to the discussion from §5.3.1 about using public interfaces.

Finally it shall be mentioned that mobile participant stations should be designed to operate in case of catastrophe without the location fixed station (cp. [Pfitz\_89], §5 and

## *5 Security in Communication Networks*

§6.1]. They also should be able to use other frequencies (AM, FM) for emergency calls.

The things said about mobile radio communication are to be considered when creating a **traffic guidance system**: It is uncritical regarding anonymity, unobservability, and unrelatability to distribute information to vehicles; it is very critical if vehicles have to send on a permanent basis like with project PROMETHEUS [FO\_87][Walk\_87].

With the arguments from §5.1 the traffic guidance system should be designed to recognize vehicles but not the type nor specific instance - otherwise this kind of traffic data is as unprotected as the traffic data cover in the rest of §5.

Reading recommendations

- Encryption: [SFJK\_97],[PSWW\_98]
- data of interest [Pfit\_89]
- MIXes: [FrJP\_97],[FJMP\_97][FGJP\_98],[JMPP\_98]
- Classification into a layer model: [SFJK\_97]
- Accounting: [Pede\_96],[FrJW\_98]
- Public mobile radio: [FeJP\_96],[FJKPS\_96],[Fede\_99]
- Up to date literature:
  - Computer & Security, Elsevier
  - ACM Conference on Computer and Communication Security, IEEE Symposium on Security and Privacy, ESORICS, IFIP/Sec, VIS
- Notes on current research papers and comments about current developments:
  - <http://www.itd.nrl.navy.mil/ITD/5540/ieee/cipher;>  
cipher-request@itd.nrl.navy.mil
  - Datenschutz und Datensicherheit DuD, Vieweg-Verlag



# 6 Value Exchange and Payment Systems

After a short discussion about degrees of anonymity towards participants, it will be shown how legal security can be achieved for business processes under the condition of anonymity. Two examples will be discussed in detail: First, how value exchange, i. e. exchange of goods against money, can be transactions be secured from fraud. Following that, the execution of digital payment systems will be covered.

This is a lightly revised chapter taken from [PWP\_90].

## 6.1 Anonymity of participants

Obviously it is pointless to claim that business should be unobservable to participants. In fact, the aim, in order to prevent the recording of unnecessary personal data, is to conceal the identity of a participant from his business partners, so to keep him **anonymous**.

There are three criteria for the grades of anonymity:

The first criteria is based on the chosen **model of the attacker** (see §1.2.5), describing the need for anonymity towards specific business partners and the chance of achieving it, even if several partners and non-participants gather their information together. In particular, this includes asking if certain predicted instances exist that can, in the case of business disagreements, lift anonymity through the will of one business partner. This should be prevented, because if some instances can already lift anonymity, it will give them a too powerful position, while leaving them too unreliable when many instances must be used to lift anonymity (the borderline case would require the entirety of users).

Secondly, a specific attacker can be requested from among **all possible acting participants** the true acting participant is hidden. In open systems, all users are potential acting participants for all actions. In reality this can be limited by the capabilities of the communication network, see §5.4. In this way, the declaring user can only hide between some users, but not between all, if the business partner and the network provider work together. Bear in mind, the number of potentially acting users must be big enough that the damage caused by the loss of anonymity for one user can not be inflicted to all acting users without becoming a bigger disadvantage than advantage for the damaging users [Mt2, 16]. For example, if only one out of 10 people is known to be a client of a publisher of unconstitutional material, the constitutional loyalty of all 10 persons can be questioned and making it difficult for them obtain employment in some areas.

Thirdly, it is not enough to pay attention to anonymity for isolated actions, but it must also be taken into account how the examined attacker is able to link businesses or branches of businesses to each other, see §5.1.2. In particular, linkage concretely

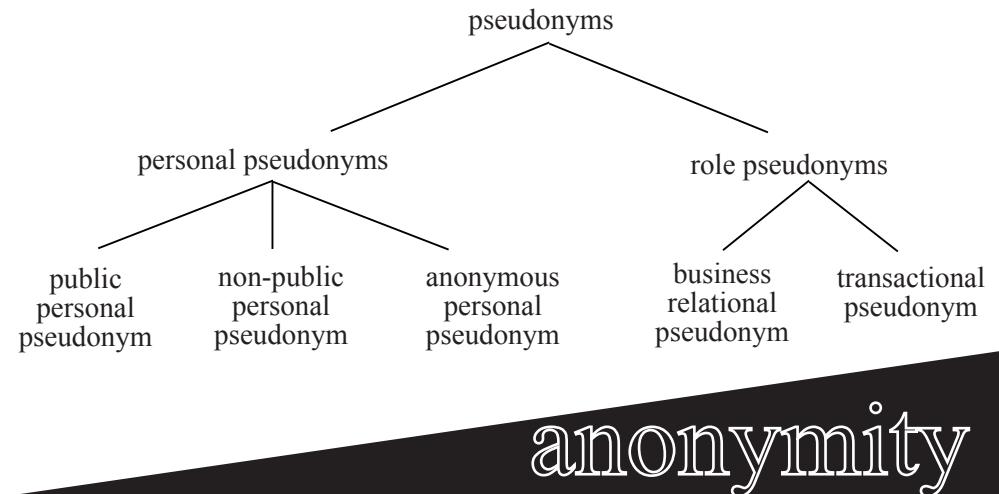


Figure 6.1: Classification of pseudonyms according to personal relation

appears for a business partner, who knows that two actions are being carried out by the same person. Catenation has to be kept at a low level for confidentiality or sometimes for legal security, especially between transactions of the same business or for multiple communications with one credit institute in for authentication purposes.

The user's level of anonymity is not just measured by the symbols which the partner automatically learns from the user, such as type and date of the transacted business, but above all by independently-chosen symbols like identification numbers or testing keys for digital signatures (§3.1.1.2), so called pseudonyms.

From a practical point of view, a useful classification according to the strength of realized anonymity is given in Figure 6.1.

A pseudonym is called a **personal pseudonym** if its owner uses it for several business relations throughout a long period of time. Therefore it is a name substitute. With respect to the linkage opportunities, the person using a personal pseudonym has, we distinguish between three types of pseudonyms.

If considering the first time a "person to pseudonym" is used, then with **public personal pseudonyms**, a person's assignment is, at least in principle, generally known (e.g. telephone numbers). For **non-public personal pseudonyms**, this assignment is only known to a small number of participants (e.g. secret telephone numbers, anonymous account numbers) and for **anonymous personal pseudonyms** this assignment is only known to the owner of the pseudonym (biometric attributes or even his DNA). For personal pseudonyms, an observer continuously receives data about the owner of the pseudonym and after some time it will be possible to identify him. Therefore each personal-related pseudonym is a potential personal identification.

This disadvantage can be overcome through the use of **role-related pseudonyms**,

which, in contrast to personal pseudonyms, are not assigned to a person, but only to that person's role during a transaction. **Business-related pseudonyms**, are role-related pseudonyms used for several transactions (e. g. account numbers used for many recordings on an account, or a stage name). **Transaction pseudonyms** are only used for one transaction, e. g. passphrases for anonymously-posted ciphered advertisements. Using role-related pseudonyms, the different parties are not able to link the collected information for the user's pseudonym simply based on pseudonym equality, but at all cases it is possible to detect correlations of time or the amount of money for a transaction. Nevertheless, there is a threat of becoming identified if business-related pseudonyms are frequently used and enough information is collected. Therefore, from a confidentiality point of view, transaction pseudonyms should be used whenever possible.

For a detailed view on pseudonyms, the following needs to be defined: A person  $X$  acts during transaction  $t$ , in the role  $R$  (e. g. the customer) towards another person in role  $S$  (e. g. service provider) is written as  $p_R^S(X, t)$ . Attributes which are not needed, e. g. transactions for business-related pseudonyms, can be left behind.

Communication between partners kept anonymous in that way, can easily be achieved through a communication network protecting traffic data. This kind of network allows the sending of anonymous messages to recipients that are only specified by arbitrary pseudonyms, without unfolding the participant's location or identity. There is no restriction on the application of cryptographic systems because the keys of asymmetric systems, for encryption or key exchange, can refer to a pseudonym instead of an identifiable user.

After this chapter called attention to the fact that anonymity is necessary and technically executable to obtain provable data security in open digital systems, the following will cover how the desired legal certainty can be achieved without giving up anonymity, i. e. during authentication.

## 6.2 Legal security of business processes while preserving anonymity

This section abstractly examines what must be considered in the design of business processes in order to guarantee legal certainty despite anonymity, without a closer examination of current legislation (see therefore [Rede\_84, Clem\_85, Köhl\_86, Rede\_86]).

This is the following order, with the process of legal transactions also followed by potential claim settlements. With that, the order with the transaction order of the act in the law together with possible damage regulation. Much of the following also applies to non-anonymous business processes in open digital systems, because we have to assume that business partners do not know each other from the beginning nor have the possibility to identify themselves, so anonymity (even when unobservable) applies initially.

## 6.2.1 Declarations of intention

### 6.2.1.1 Anonymous declaring or receiving of declarations

The possibility to declare or receive declarations anonymously is provided by the communication network protecting traffic data.

When making corresponding statements with the condition that either all participants sign or none (which is not the case for most open systems), this can be handled as a complete business process for the exchange of signed statements and can be finalized like an exchange of "goods against money", which is described later in this section. If so desired, special contract closing procedures [BGMR\_85, EvGL\_85] can be applied, allowing a nearly simultaneous exchange of declarations without using third party services. However, the cost of communication rises sharply through the use of contract closing procedures.

### 6.2.1.2 Authentication of declarations

For making a declaration it is often necessary to proof the authorization. **Digital signatures** are an essential tool for doing this in communication networks.

#### 6.2.1.2.1 Digital signatures

Instead of hand-signed signatures, so-called digital signatures §3.1.1.2 will secure in legal transactions through open digital systems that declarations are made by specific persons (or even groups of persons, etc.) associated with a certain pseudonym.

Basic demands towards a digital signature are:

1. Nobody except the owner of the pseudonym should be able to give a pseudonym-specific signature for a document.
2. Everybody is able to test if a declaration provides a pseudonym-specific signature.

The first claim correlates to the *will* of the user: Of course no one can hinder a user in a pure digital system from allowing other users to act under his pseudonym. Even to this day, this corresponds to the possibility of randomly offering somebody many blank autograph signatures, which at most does damage to the user himself, as it must be assumed that the given signatures belong to the owner of the pseudonym.

The simple approach, to trust on the uniqueness of pseudonyms and to submit them together with the declaration like usual signatures, is not sufficient according to the requirements mentioned above, because everybody who received a signed declaration from a user is able to copy the submitted pseudonym to many random declarations. In particular, added to this insufficient approach is the occasionally discussed possibility of using a digital copy of the autograph signature for authentication of declarations. For digital documents, it is easy to separate an autograph signature from the document, even if it is written across the paper-based version of the declaration. (The ideas from [Köhl\_86] cannot be applied, because facsimile signatures, as with bank notes, become combined

## 6.2 Legal security of business processes while preserving anonymity

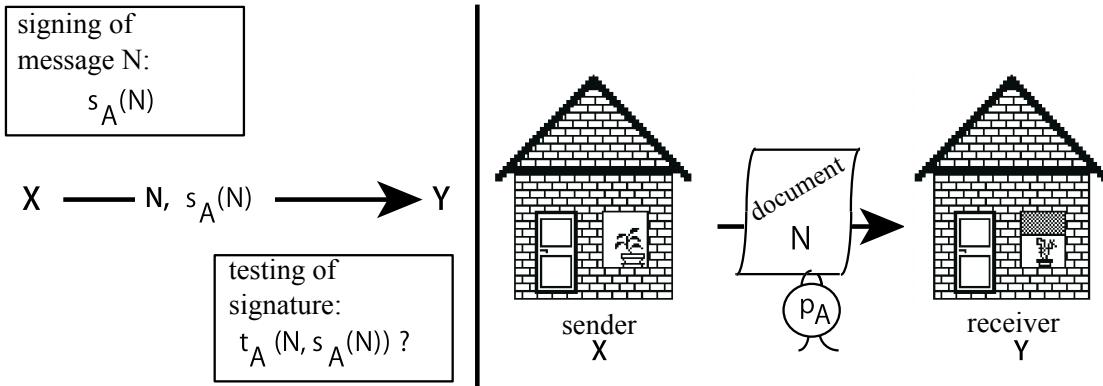


Figure 6.2: Transfer of a signed message from X to Y, functional view (left), graphical view (right)

with special paper forms, which cannot be sent in digital systems. Furthermore, printing on such forms is not more secure than autograph signatures if you consider the copying technology available these days, only the structure of the paper itself or something similar might make a difference).

Therefore, even for the non-anonymous case, a digital pseudonym different from the name is required (public personal pseudonym), which is one reason to understand it as a special case of the anonymous version.

A solution for the above problem is a **digital signature system**.

Figure 6.2 shows the transfer of a signed message from user X to user Y. X uses the pseudonym  $p_S$  (for sender); at the left side a functional view and at the right a graphical view, which will be used in the following, one representing a message as a document, the other a signature as signet.

For some applications it is useful to demand additional properties for digital signature systems:

The recipient e. g. of a GMR-signed message can show it around to convince other participants of the authenticity of the message. The signatory has no control over the circulation of his signature. This insufficiency is remedied by **undeniable signatures** (§3.1.1.2 and §3.9.4): So that another member believes the authenticity of the signature, the alleged signatory must be questioned; this signatory can not deny signatures he actually made.

Every signature system (as well as each asymmetric encryption system) can be broken with enough effort. In common signature system the risk is on the signatory's side. In a **fail-stop signature system** (§3.9.2) the risk can be shifted to the signatures' recipient: If a signature is counterfeit, the alleged signatory can prove this to any third party.

Another variation that genuinely needs to be explained later, is a **blind signature**, which will be introduced in §6.2.1.2.2.

Digital signature systems should not be confused with so-called **digital identification systems**. While the recipient of a signed message is able to prove the authenticity of

a signature to a third party in digital signature systems, these systems only provide a way for the recipient to identify the declaring party as owner of a specific pseudonym at the moment the declaration takes place [WeCa\_79, Bras\_83, FiSh\_87, Simm\_88]. An identification system is therefore a symmetric authentication system for just one message.

For simplicity's sake, the declaring partner submits a pseudonym together with the message (possibly one with a special structure) or encrypts the message using a symmetric cryptographic system where only the two communication partners know the key. After receiving the message it is not possible for a third party to check the signature for authenticity. Therefore identification systems appear inappropriate for the authentication of a declaration for reasons of legal certainty.

#### 6.2.1.2.2 Types of authentication

Depending on where the authority to submit the declaration is obtained, we can distinguish between **self authentication** and **third party authentication**.

Self authentication is given when operator uses a declaration he made earlier, e. g. ordering goods and referring to a previously requested binding offer.

Here the declaring partner wants to show that both declarations stem from the same person. This is a deliberate linkage of different declarations and can be achieved by using the same digital pseudonym and corresponding signature for both declarations. In any case, classic systems would have used the same name and autograph signature, and eventually an additional identification number would have been made for the linkage of different declaration to ease business processes.

Third party authentication is when the declaring participant receives his declaration authority from a previous declaration made by a third party authority.

In this case the declaring partner needs a document such as credentials or a credit certificate to show that the owner of the pseudonym is authorized to give a certain declaration. This document is submitted with the declaration itself.

The issuer of the document, on the other hand, can be authorized by other documents.

Without further measures, third party authenticators e. g. credit institutes (in case they have to give a guarantee for each purchase) would be able to derive unwanted linkages in collaboration with the recipient of the declaration. This can be prevented by using **credentials** ([Chau\_84]), which allow access to authentication issued for a pseudonym different from the one used for the declaration.

Therefore it has to be possible to convert the authentication for the used pseudonym without involving the issuer. On the one hand, no one besides the declarant may recognize the connection between the two pseudonyms without his permission and on the other hand, the declarant should only be able to convert the authentication for his own pseudonyms, not, for example, a friend's pseudonyms.

The first system based on RSA was suggested in 1985 [Chau\_85, ChEv\_87]. A specific signing function is assigned to all possible authentications. To issue a authentication for a pseudonym, it is signed with the corresponding signing function. This implies that the possible authentications must be strongly generalized.

## 6.2 Legal security of business processes while preserving anonymity

If it is sufficient to transform an authentication just once, then the procedure allows the pseudonym for which the authentication ought to become converted, to choose in such a way that it is suitable for signing [Chau\_89]. This is possible if an authorization needs to be used and converted multiple times, but the resulting pseudonyms are then not suitable for signing and are therefore only usable for identification systems.

While the anonymity of the procedure is completely secure (informational theoretical secure), the security itself is at most as secure as RSA.

The special, though also unproved, signature system from [Chau\_89] can be used similarly.

In [Damg\_88] a provable secure system for converting authentications was suggested, based on an arbitrary cryptographic secure signature system. For expense reasons, its use is not practical.

Should the occasion arise that RSA is broken, but other asymmetric cryptographical and signature systems are still available, a variation of the application suggested in [Chau\_81, p.86] of a communication network protecting traffic data can be used according to the following scheme: If  $n$  users are coming together that want to obtain identical authentications (signatures) for pseudonyms  $p_1, \dots, p_n$ , then the organization asked to provide these, checks that all  $n$  users are authorized for the desired signature (i. e. if the organization has already issued an authorization for another pseudonym to this user). Only then does the organization allow every user to send exactly one message containing their pseudonym to the communication network protecting traffic data, which, logically, has to be established using an already existing physical network. This takes place in a random order. The communication network guarantees that nobody can obtain information about the assignment "pseudonym to user". Now the organization signs all published pseudonyms.

Any signature system can be used. While the system for convertible authentications provides informational theoretical security to hide a user  $X$  between all others who are receiving the same confirmation as  $X$  as long as he did not use it, the last system only provides him the same security as the logically realized transfer status information securing communication network can provide the  $n$  users receiving the confirmation simultaneously.

A different approach to prevent users from unwanted linkage through foreign authentication is based on the assumption that **secure and unanalyzable equipment** exists. It would be possible to add and delete permissions of users through communication with other secure systems. The secure system would be able to confirm registered permissions using system-wide provable signatures.

In contrast to normal computer expectations, this application requires the system to work based on secret data, e. g. cryptographic keys, which need to be concealed from the user. Otherwise the user would be able to change permissions inside the system or even clone a system. It would appear unchanged to other users, but could hold additional or different permissions.

Still, it has to be controllable for the user it serves, so it should be in his physical domain. This is widely assumed in literature [Riha\_84, Cha9\_85, Davi\_85, Riha\_85].

The existence of devices that fulfill both requirements is questionable. After all,

the user is able to manipulate and observe the system, especially while it is working, whereby costly measures could be worthwhile, since the analysis of one device enables the building of many copies. Any processing of information inevitably causes an energy transport within the device. To prevent the measurement of the energy transports (e. g. through electromagnetic radiation), the device has to be shielded sufficiently. Because a user could try to analyze his device through destructive measurement, the system has to detect when its safeguard becomes affected from the outside. In this case (and for affects on a function due to internal errors), the system has to destroy itself, i. e. it has to delete its secret information, cp. §2.1.2.

This leads to a competition between constructors of secure devices and the developers of measurement technologies, which probably neither will win in the long run.

Nevertheless, chip cards are used in everyday life as secure devices. Possible applications are identification cards for access to databases or public phones (where the permission is set to n-times and will be decreased for each usage) or for digital payment systems (see §6.4).

In our opinion, devices which have to be secured against their owners should be used as little as possible.

### 6.2.2 Other actions

It is only worth anonymously sending and receiving statements about legal transactions if all other actions in that context also guarantee anonymity. The anonymity is already supplied by the transfer status information securing communication network if the above mentioned actions are realized through sending information via the communication network e. g. merchandising database query results. In an open system it should also be possible to transmit money digitally, e. g. in the form of several consecutive statements, see §6.4. These already cover the most important actions for legal transactions.

If parts of the actions are completed via the communication network (e. g. delivery of material goods chosen and ordered via the open communication network), anonymity cannot completely be kept; and sometimes this is not desirable. Therefore, the following explanations will be limited to statements and actions in the communication network.

### 6.2.3 Securing of evidence

Based on the possibilities for authentication, which were explained in chapter §6.2.1, it must now be explored in which way enough evidence of the transmission and reception of a professed intention can be found.

Just as in chapter §6.2.1, no major new problems occur in comparison to the public case.

The investigation can be divided according to these two criteria:

First the aim of the argument can be considered. The aim could be the transmission or reception of a statement. But it could also be the opposite, i. e. to prove that the transmission or reception has not taken place.

## 6.2 Legal security of business processes while preserving anonymity

The second aim can be reached by simply achieving the first one completely. If it can not be shown that a statement was sent or received, then this statement can be regarded as not sent or received. The second objective is therefore not pursued explicitly in the following.

As a second criterion it can be considered who is interested in the argument: the sender or the receiver. Often the fact of a statement having been transmitted will only be advantageous for one of the two parties, so that only this person is interested in the proof at all and has to collect evidence. Finding a proof is easy for the receiver of the statement, if the statement is a document, i. e it carries a digital signature. Then the presentation of the statement is enough proof that the owner of the digital pseudonym belonging to signature made that statement.

In many cases it is not necessary for the sender to collect evidence for the transmission or even the reception of the statement by the recipient, even if the fact that the statement was given is an advantage for him.

If the exact time of the transmission is not important, it is sufficient to repeat this statement as soon as its transmission is doubted if necessary in court. For example, this is true for transactions in everyday life which take place in open communication networks. Likewise, the sender does not have to collect evidence that he has transmitted information as goods or messages connected with the transmission of digital money. In contrast to the delivery of material goods or paper based documents, the supplier still stores the provided information. The receiver does not obtain additional advantages from repeated deliveries, because the follow-up deliveries only represent copies of the original delivery which the receiver is quite capable of generating himself.

If reception of the input must be proved, then there are possibilities similar to public statements available.

One possibility is to demand a signed receipt; the reception of that can easily be proven (see above). If the supplier does not get the receipt within a defined time limit, it can be enforced in court or provided by a court. In such a case the following alternative is also necessary. The second possibility consists of so-called blackboards in an open communication network at which potential receivers of statements with a receipt have to watch for such statements regularly. As the computer can do this job, it means less of a burden than the obligation to empty the mailbox daily.

The fact that a statement for a user who uses this pseudonym  $p_E$  could be found on a blackboard can be proven by witnesses. The statement has to be encrypted with a key only known to the receiver of the statement, so that the witnesses do not learn the contents of the statement. Assuming there exists a key  $c_E$  for an asymmetric encryption system and a pseudonym  $p_E$  assigned to it in a verifiable way (e. g. through a public trust center or  $c_E$  being a pseudonym itself), the declaring partner is able to prove that the statement arrived at the user of the digital pseudonym  $p_E$  by simply presenting pseudonym  $p_E$  and the unsigned statement (and for indeterministic encryption, if applicable, a random number, see §3.1.1.1 annotation 2). It would be most convenient for a public institution to be a witness, and publish and sign regularly (e. g. daily) a list of all incoming encrypted statements, since in this case, the witness is controllable and the

recipient does not lose his anonymity.

#### **6.2.4 Legal investigations**

If the two engaged parties find themselves in a situation in which one of them doubts the legality of the state of affairs, then the substance of these doubts has to be further investigated.

The user who initiates the investigation does not have to reveal his identity. For the moment it is sufficient to establish whether, for example, the owner of a certain pseudonym was actually deceived. It should be noted that not everybody can be forced to take part in the proceedings, as in any case the identity of all possibly involved persons should not be made public in advance (it could lead to abuse).

Accordingly, all necessary evidence should be owned by users whose participation is guaranteed. This includes the party who initiated the proceedings and all public persons involved, such as notary publics, but also other anonymous persons in case they could suffer damage from non-participation. For example, an anonymous database is sued for receiving money but is not providing satisfactory information. This database is forced to provide the transmitted response, as otherwise it would be assumed that nothing had been sent and the database would be condemned. (Compare §6.2.5 to see that this is really a threat. Additionally, it has to be considered that it is possible within the transfer status information securing communication networks, to securely send to the database the summons under its pseudonym, compare §6.2.3).

#### **6.2.5 Settlement of damages**

If it has been established that an illegal state has occurred, a legal state similar to today's legal enforcement must be reestablished.

In general, the sentence is a fine for the user. The user's property should be big enough, the connection between this property and the determined demands should be clear and access to the property or the special part of it on which the demands are made, should be possible.

Whether it can be assured from the outset that the user has sufficient assets does not depend on anonymity, but on principle considerations referring to the respective legal transactions, and can currently not always be guaranteed.

The fulfillment of remaining demands is the main difference between anonymous and public systems:

If anonymity is undesired, the connection can be established through the name (and other data for clear identification).

This means that the asset management and preservation of evidence can be organized independently from each other, because all the property and also all the evidence, refer to persons known by name.

## *6.2 Legal security of business processes while preserving anonymity*

This can not be done if anonymity must be protected. That does not mean that all possibilities to settle damages are gone, but only that business transactions become more complicated. Already when collecting evidence, one must watch for connections to the property, which should eventually be accessed.

For this reason, the examination carried out in this chapter, which looked at the components of a business process, does not automatically lead to standardized protocols for whole transactions, but only provides strategies that still need to be skillfully combined.

The detailed regulations necessary depend primarily on who is required to do what and for how long, based on the legal transaction.

If the settlement of damages becomes necessary because of i. e. a large credit, then the precondition for the guarantee that the property can be accessed will always be the transfer of a loan security in the form of material goods, e. g. real estate. This can also be done anonymously by using an authorization mechanism (cp. §[6.2.1.2.2](#)). For example, a land registry office could certify an appropriate confirmation of the value of the property and note down the property as mortgaged. However, business transactions of this size are rather unusual for open systems.

A lot more typical are business transactions which are purchases of goods, especially information, with a low value. Here, somebody is merely committed to paying a certain small amount of money within a short period of time after the delivery of the goods. The purchasing of goods for a small amount of money will be examined in a separate chapter ([§6.3](#)), because of its importance for open digital communication networks and in order to have room for the presentation of complete business transactions.

The performance of services like the administration of an electronic mailbox, can – with only two exceptions – be regarded as a succession of many small purchases. For example, the administration of a mailbox could always be accounted for when the user wants to empty it. The transfer of the content would then correspond to the delivery of goods.

The two exceptions mentioned below are services that have to be used to make purchases via the open system possible: the transfer of messages through the transfer status information securing communication network and the provision and the administration of money through an anonymous digital payment system.

The communication network could be paid at a flat rate or through the addition of money (“digital stamps”) to each message, cp. §[5.6.2](#) to mention two examples. In the first case users who do not pay could be excluded from the communication network. In the second case unpaid messages would not be delivered.

As the provider of the communication network is not anonymous, the deliberate damaging of the user would have to be pursued by law outside the system like it is done today.

The services of the anonymous digital payment system can be treated likewise: the banks involved can retain the fees directly, and the customers can take the banks to court if necessary (this can be done through the communication network to protect the customer’s anonymity).

## 6.3 Fault secure value exchange

From the point of legal security, cash purchases in the classical sense could be performed providing full anonymity. The business partners being at the same location ensures that either goods and money or neither can be exchanged, and therefore (apart from additional reclamations) damage regulation will never be needed.

Such simultaneity in the exchange of goods and money is not given in communication networks. There will always be times when business partners have the advantage, so that if they break connection at a certain point in time, it could result in fees that will be enforced if necessary.

In the following two sections, two concepts describe how the enforcement of such fees can be secured. We prefer the second one.

### 6.3.1 A third party guarantees de-anonymization

The first concept guarantees the de-anonymization of debtors in the case that they do not pay their debts. This leads to the non-anonymous case and allows access to all the debtor's possessions.

Technically this is achieved using a special kind of foreign authentication: They show their identity to a third party they all regard as trustworthy, which is not anonymous. This instance certifies to be able to identify the owner of a special pseudonym should the occasion arise (non-public personal pseudonym, cp. §6.1).

The authentication can be based on a third party [Herd\_85] or a sequence of third parties [Chau\_81]. The principle is shown in picture 6.3 for one third party participant (but possibly different ones for different users). Let  $X$  and  $Y$  be the users and  $A$  and  $B$  the non-anonymous instances, who are capable to identify  $X$  and  $Y$ , and let

- $p_G(X, g)$  represent the pseudonym used by  $X$  during business  $g$  with  $Y$
- $p_{G'}(Y, g)$  represent the pseudonym used by  $Y$  during the business (but in a different role than  $X$ )
- $p_A$  and  $p_B$  represent the public personal pseudonyms for  $A$  or  $B$

The identifiability of  $X$  by  $A$  is guaranteed when  $X$  gets his pseudonym  $p_G(X, g)$  signed from  $A$  before being able to use it. Similarly,  $Y$  needs to get his pseudonym  $p_{G'}(Y, g)$  signed by  $B$ . To guarantee that  $A$  always identifies the right person,  $X$  has to send the message "The pseudonym  $p_G(X, g)$  belongs to  $X$ ", signed by himself, to  $A$  before getting his certificate from  $A$  (this could carry his own autograph signatures or, even more secure, a digital signature belonging to a special personal pseudonym).

If you trust the third party not only in regards to authentication, but also legal security, it is sufficient to use an identification system (cp. §6.2.1.2.1); e. g.  $X$  could identify himself with his finger print. This works similarly for  $B$ .

If  $X$  and  $Y$  exchange their signed pseudonyms, they know that if there is fraud,  $A$  is able to identify  $X$  and  $B$  is able to identify  $Y$ .

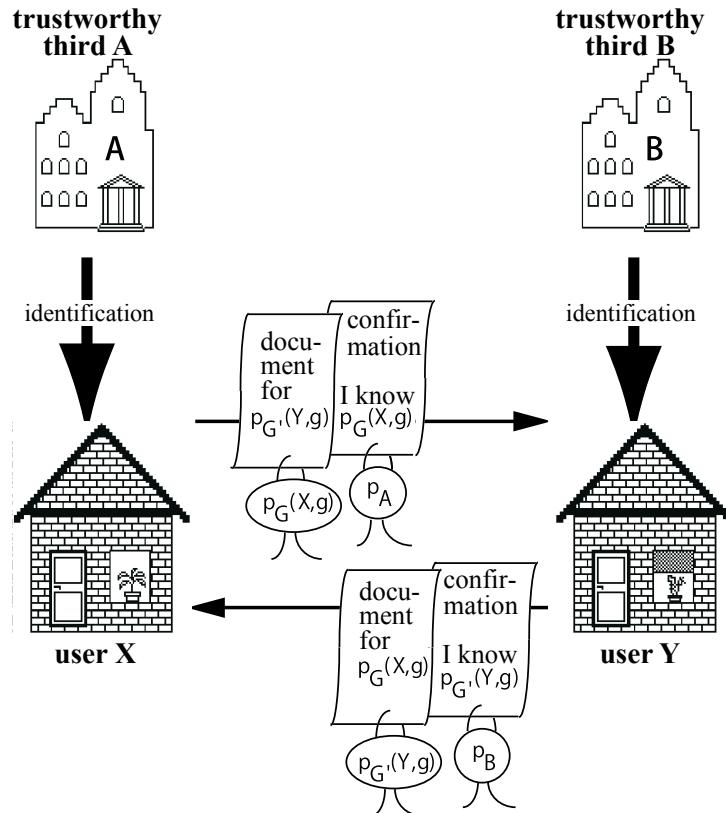


Figure 6.3: Authenticated anonymous declarations between business partners who could be made public

In the following it could happen, that  $X$  complains to  $B$  about  $Y$  broke off communication even though he was obliged to continue it. Therefore  $X$  hands over the messages from  $Y$  which are signed with  $p_{G'}(Y, g)$  to  $B$ , proving a previous existence of the communication agreement. Subsequently,  $B$  asks  $Y$  to continue communications, but now all messages from  $Y$  to  $X$  must be sent through  $B$ . In the case that  $Y$  refuses to proceed,  $B$  publishes  $Y$ 's identity and the investigation can continue like in the non-anonymous case.

If  $X$  accuses the other one by mistake in not sending all messages he received from  $Y$ , the worst that can happen is that  $Y$  has to send the remaining messages for a second time. This time they are forwarded by  $B$ , hence  $X$  cannot pretend that he did not receive the messages.

The same applies if  $Y$  complains about  $X$  in front of  $A$ .

Advantages (+) and disadvantages (-) for this solution:

- Who is able to control  $A$  and  $B$  so that they do not publish the identities of  $X$  and  $Y$  without authorization?  $A$  and  $B$  have to be completely trustworthy in terms of the confidentiality security, but not in terms of fraud security, because they can not change the assignment of "persons to pseudonyms".
- $X$  and  $Y$  are able to cause damage they cannot compensate. E. g.  $X$  could order and use a service provided by  $Y$  even so he does not possess enough funds to cover his debts. Even by publishing the identity of  $X$  the damage to  $Y$  cannot be compensated.
- Considering the cost of communication, this solution suggests personal pseudonyms for  $p_G(X, g)$  and  $p_{G'}(Y, g)$ . As shown in §6.1 this can lead to a gradual revealing of the partner's identities.
- + If personal pseudonyms are used, it is not necessary for  $A$  and  $B$  to participate in every single business between  $X$  and  $Y$ .

[Chau\_81, p.86] describes how the ability to publish someone's identity could be shared between several third parties. The given procedure only allows the revealing of someone's identity if all third party participants cooperate. Some further developments are given in [Pfi1\_85] for tolerating the loss of some third party participants.

Even with the increasing anonymity using several third party authorities, we still have the disadvantage that it is not possible to guarantee the existence of sufficient funds.

### 6.3.2 Trustees guarantee fraud protection for anonymous partners

To prevent damage from uncovered debts, an easy procedure can be applied, which even works without making someone public: money gets deposited with a non-anonymous trustee so that claims can only arise against him [Pfi1\_83, p.29-33], [Waid\_85, WaPf\_85].

The actual business partners can stay anonymous for each other and the trustee. In case the trustee exceeds his authority, since he is not anonymous, he can be sued without revealing the identity of the actual business partners.

Instead of a direct exchange of "goods against money", all participants inform the trustee about the amount of money and the type of goods that they want to receive and hand over money and goods. This has to be performed in exactly this order. Should the trustee receive the goods, but not the money, he would not be able to pay for the supplier's costs. The trustee checks if what he received meets the expectations of his clients. According to the result of this check, he either forwards what he receives or aborts the transaction. The customer who ordered cannot cancel the business anymore, at the latest, when the trustee receives the goods.

To finalize the business as fast as possible, the trustee must be within the open digital communication network. It could, for example, be the network provider, who already provides similar services these days.

Having  $X$  and  $Y$  as users again where  $X$  is randomly chosen as the customer and  $Y$  as the supplier of a good (in the form of information), Figure 6.4 shows the whole procedure where the concrete numbers of the documents define the order of the declarations being made. Therefore:

- $p_K(X, g)$  is the pseudonym of  $X$  as a customer in business  $g$
- $p_L(Y, g)$  is the pseudonym of  $Y$  as a supplier in business  $g$  and
- $p_T$  is the public personal pseudonym of the non-anonymous trustee  $T$

The illustration of the transfer of money is heavily simplified. Of course, money cannot be pictured as a single document or a declaration – otherwise you could increase it in number by producing copies of it (cp. §6.4).

As described in §6.2.3, it is not necessary to secure evidence for all actions performed by the active partner (depositing and forwarding, or returning the money, and delivering and forwarding the merchandise). The proof for the reception of the order can be given by simply presenting the document (first through the trustee and then the supplier).

This solution should be the preferred because of data security, but has one disadvantage: the trustee is asked to perform certain checks of the merchandise, which he can not always and sometimes should not be able to perform.

To come up with a solution for this disadvantage,  $X$  and  $Y$  could agree on a complaint period. During this time the trustee  $T$  keeps the money, but delivers the merchandise (Figure 6.5). This way the trustee only has to prove the authenticity of the money and does not have to check the merchandise anymore, so he is not obtaining information about it.

If the customer  $X$  is not satisfied with the merchandise, e. g. the query was answered falsely, he can, within this time period, stop the trustee from transferring the money and then has to prove that the merchandise was indeed faulty.

If things can be arranged fast enough for a claim to be put before a court, then  $X$  can not cause great damage to  $Y$ , e. g. through the loss of money due to unpaid interest.

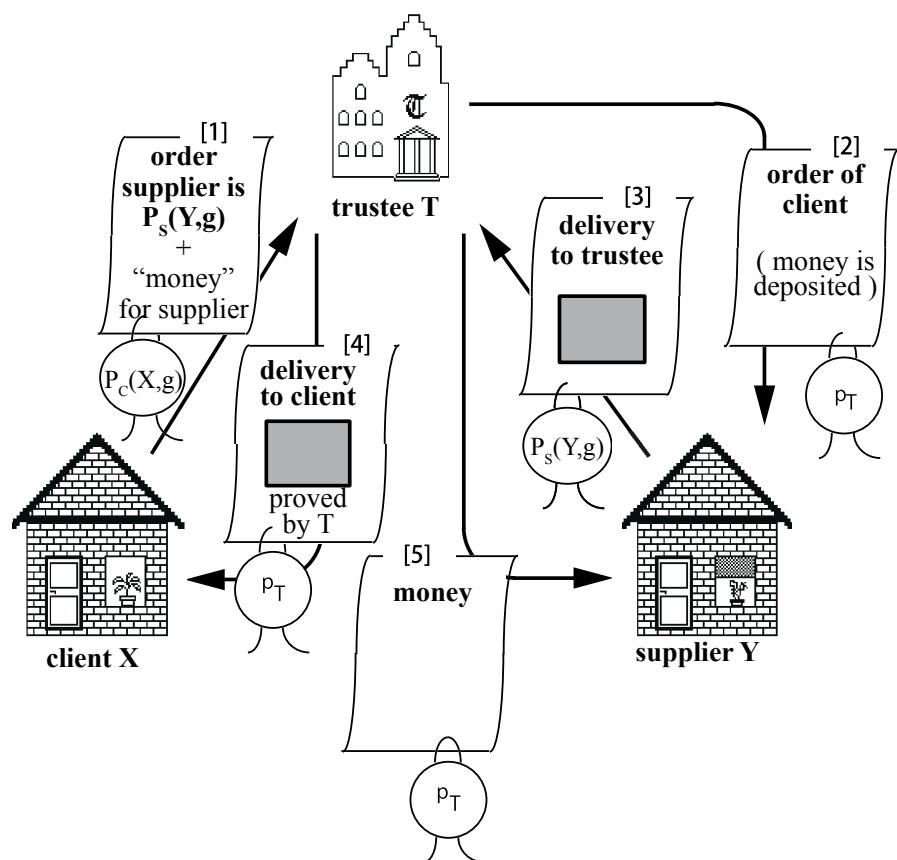


Figure 6.4: Fraud protection for completely anonymous business partners through an active trustee, who is able to check the merchandise

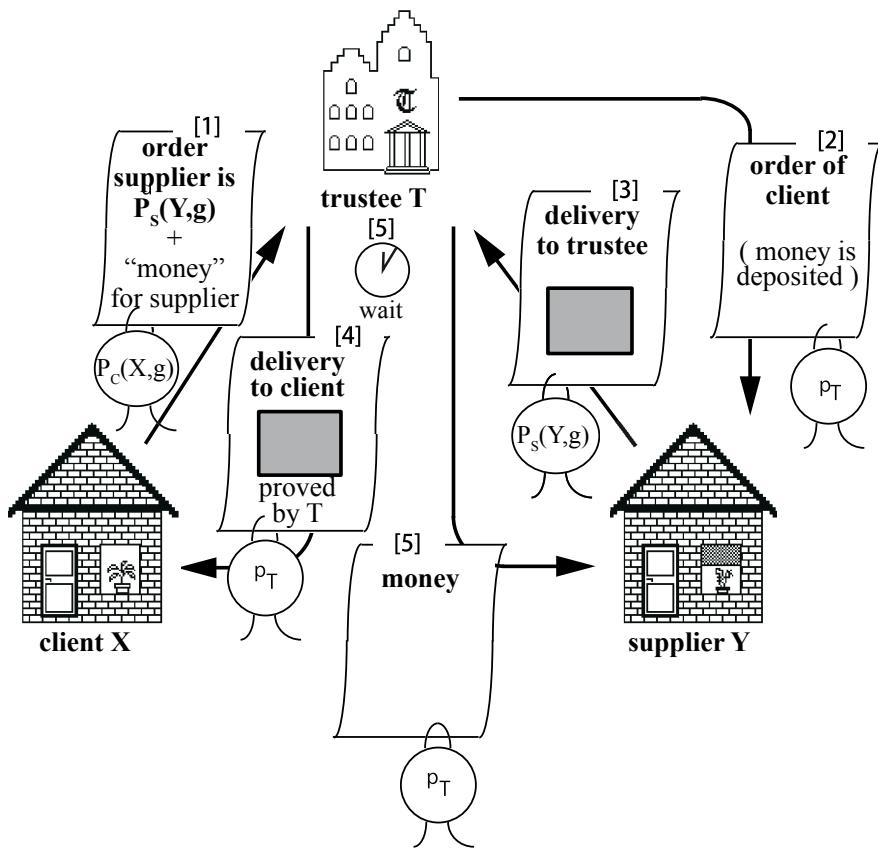


Figure 6.5: Fraud protection for completely anonymous business partners through an active trustee, who is not able to check the merchandise

Even this loss of interest can be prevented if  $T$  bears the cost of the interest and both  $X$  and  $Y$  split the cost of the difference between the actual interest and the usual interest as an interest deposit for  $T$ . Both interest deposits are later given to the winner of the claim. To force  $X$  and  $Y$  to pay this deposit or to forgo a complaint, the whole amount is given to the other party if one party refuse to pay the interest deposit [BüPf\_90].

In case  $X$  and  $Y$  just want to exchange goods without the trustee  $T$  being able to check the goods or being able to check them, the business has to be split up into two coordinated businesses of the kind "merchandise against money" to stick to the rule "money before merchandise" for the exchange. Coordination is achieved as  $T$  does not send the money to either  $X$  or  $Y$  before either both partners have declared that they are satisfied with the merchandise they received or the court decides the case [BüPf\_90].

This extended solution using the trustee can be assessed as follows:

- The trustee always actively participates in the transaction.

- + It is not necessary for one of the direct participants to trust the trustee, because he gets controlled by both participants, who are able to sue him for any kind of failure or fraud. The existence of enough evidence is given at any point and the enforcement of demands towards  $T$  can be secured in a similar fashion to the non-anonymous case.
- + All of  $X$ 's demands of  $Y$ , or the other way around, can be satisfied by relying on the values deposited with  $T$ .
- + The participants in a transaction can use transaction pseudonyms without additional costs.
- + Anonymity is given for direct participants in a transaction.

In case the service provider has no interest in staying anonymous, such as a newspaper publisher, he can of course be chosen as the trustee. This results in the same solution in which the customer pays immediately when he orders and the supplier has to refund him if he is not able or not willing to deliver the ordered merchandise.

All described concepts leave one problem open, namely how to represent money in open digital systems and how to transfer it anonymously. This will be covered in the following section.

## 6.4 Anonymous digital payment systems

A payment system should help its user to securely transfer money.

Independent from the particular payment system, money is just a quantified amount of rights. In order to discuss security and anonymity in payment systems, it is not necessary to define what and where the money is. In the following we will discuss money as rights.

It is especially unimportant for the security inside the payment system, whether these rights are based on a true balance or a limited loan, whereas a credit outside of the payment system naturally has to be sufficiently secured.

A payment system is secure (considering accessibility and integrity), if

- a user is able to transfer received rights,
- one only loses a right when he so wishes,
- one who is willing to pay decides on a recipient, and only this recipient can be the recipient of the transferred rights,
- one, when necessary, is able to account for a performed transfer of rights in front of a third party (problem of a receipt), and
- the users are not able to increase their amount of rights, even if they work together.

If you do not trust the user completely, then you have to at least check if the sender owns the right that he is going to transfer. Since only payment systems are considered, from which the transfer is accomplished through an exchange of digital messages (i. e. through a communication network) and the possibility of copying digital messages exists, though the rights must be deleted after a transfer, a simple check of the document is not sufficient. This makes a witness necessary to guarantee the actual validity of a certain right.

To make it possible for the witness to fulfill this task, he must be informed about every single transfer of rights. Even in a case where the business partners trust each other completely, there must not exist an opportunity for the transfer of rights without confirmation from the witness.

For the sections §6.4.1 to §6.4.4 we assume that the users do not want to trust the witness completely in regards to integrity or confidentiality.

If the users trust the witness completely then the problem is much easier, as will be shown in §6.4.5.

A payment system which is insecure through the above mentioned criteria will be described in §6.4.6

### 6.4.1 Basic scheme of a secure and anonymous digital payment system

In the following it will be discussed how to implement a secure and anonymous digital payment system with witnesses.

It will be assumed that user  $X$  and user  $Y$  want to transfer a right in the payment system and witness  $B$  (credit institute) confirms this transfer. To keep it simple we just consider one witness. This witness is liable for his statements, i. e. should new rights for money be created due to his wrong certifications, he has to provide the missing funds. If the witness refuses to certify existing rights, the holder of the rights can prove this to a third party (e. g. a court). This is achieved in choosing a witness with enough funds who is not anonymous. He takes on the role that credit institutes have in usual electronic-cash payment systems.

It can be assumed that the paying partner  $X$  and the recipient  $Y$  know each other through certain pseudonyms (cp. §6.1). These pseudonyms are specified from outside of the payment system and are worth securing (in general: role pseudonyms, e. g. client number and name of the service provider). The pseudonym for the witness is specified too, but must not be anonymous (public personal pseudonym). It was already defined, during earlier transactions, which pseudonym  $X$  is using to identify himself towards  $B$  as holder of a right he wants to transfer.

Therefore, let it be that:

- $p_Z(X, t)$  is the pseudonym of the paying partner  $X$  for a transfer  $t$  towards the recipient,
- $p_E(Y, t)$  is the pseudonym of recipient  $Y$  for transfer  $t$  towards the paying partner,

- $p_B$  is the pseudonym for witness  $B$  and is valid for many transactions and
- $p_Z^B(X, t)$  is the pseudonym of the paying partner  $X$  for transfer  $t$  towards the witness  $B$ .

With these preconditions, similar to today's protocols for interbank money transfer, the following steps for the transfer  $t$  of a right from  $X$  to  $Y$  can be defined as protocol. Step  $[i + 1]$  has to be performed after step  $[i]$ . Only steps 4 and 5 can be carried out in random order or even in parallel.

- [1] **Choosing the pseudonym.**  $Y$  chooses a pseudonym  $p_E^B(Y, t)$ , under which he wants to be known to  $B$  as the recipient of a right in transfer  $t$  and tells  $X$  that he wants to receive the right as  $p_E^B(Y, t)$ . Accordingly,  $X$  informs  $Y$  about the pseudonym  $p_Z^B(X, t)$  he wants to use to transfer the right. The necessary declarations are authenticated with  $p_E(Y, t)$  or respectively with  $p_Z(X, t)$ .
- [2] **The Customer's transfer order.**  $X$  orders  $B$  to transfer the right to  $p_E^B(Y, t)$ . This order is signed with  $p_Z^B(X, t)$ . For third party authentication,  $X$  encloses an authorization with the order saying that  $p_Z^B(X, t)$  possesses the right. The authorization itself is signed by  $B$  with  $p_B$ . Because all transfers have to be accredited by  $B$ , he is able to check if  $p_Z^B(X, t)$  still possesses the right or if it got transferred already.
- [3] **The witness's confirmation.** The witness  $B$  confirms for  $X$  and  $Y$  that the right got transferred from  $p_Z^B(X, t)$  to  $p_E^B(Y, t)$ , and addresses them with these pseudonyms.
- [4] **Receipt for the customer.** The recipient  $Y$  sends a receipt to  $X$  which only names  $p_Z(X, t)$  and  $p_E(Y, t)$  and is authenticated with  $p_E(Y, t)$ . This confirms that the right arrived.

In case  $Y$  refuses to send a receipt (which cannot be prevented since  $Y$  acts anonymously),  $X$  is able to get this confirmation from  $B$  (as in [3]) together with the confirmation from  $Y$  saying that he is willing to receive the right under the pseudonym  $p_E^B(Y, t)$  (from [1]) as a replacement receipt.

This possibility in particular distinguishes the receipt problem from the value exchange problem, where a third party, e. g. the trustee, can not generate one of the exchange items for replacement.

- [5] **Confirmation for the Recipient.**  $X$ , the one who pays, sends the payee  $Y$  a confirmation of the transfer, which only indicates  $p_Z(X, t)$  and  $P_E(Y, t)$  and was authenticated with  $p_Z(X, t)$ . Also,  $Y$  can use the confirmation from  $B$  (from step [3]) together with the confirmation from  $X$  (from step [1]) to state that he is willing to transfer the right to  $Y$ , to have a proof, that he had received the right from  $p_Z(X, t)$ .

- [6] **Transformation of the confirmation.**  $Y$  wants to use the confirmation  $p_E^B(Y, t)$  issued by  $B$  in step [2] for future transfers  $t'$ , which says that he received the right. To prevent linkability from possibly revealing his identity,  $p_E^B(Y, t)$  should not be used as  $p_E^B(Y, t')$ . By using convertible authorizations as described in §6.2.1.2.2, it can be assumed that  $Y$  is able to convert the authorization for a new pseudonym. But already in step [1] of transfer  $t$ , he has to choose pseudonym  $p_E^B(Y, t')$  and from that create conversion suitable pseudonym  $p_E^B(Y, t)$ .

**Implementation of authorizations suitable for one-time conversion using RSA** (or: How to use the attack according to Davida, (compare §3.6.3.1) for blind signatures (compare §3.9.5):

*example:* Choose pseudonym  $p$  (= testing key in a random digital signature system).<sup>1</sup>

Calculate  $p, h(p)$  using the collision resistant hash function  $h$ .

Be  $t, n$  the public testing key of the one authenticating the authorization,  
e. g. a credit institute.

Choose a random number  $r$  normally distributed in  $\mathbb{Z}_n^*$ . (As a reminder: This means  $1 \leq r < n$ , and  $r$  has a multiplicative inverse number mod  $n$ .)

Calculate  $(p, h(p)) \cdot r^t \pmod{n}$ .

Get this signed from the owner of  $t$ .

(Note: If  $(p, h(p)) \in \mathbb{Z}_n^*$ , the owner of  $t$  does not learn anything about  $(p, h(p))$ , because  $r^t$  is normally distributed in  $\mathbb{Z}_n^*$ . Otherwise the generator of the pseudonym does not need the help of the owner of  $t$ :  $ggT((p, h(p)), n)$  is a non-trivial factor of  $n$ . With this the pseudonym generator completely broke RSA, compare §3.1.3.1 and §3.6.1, so that he is able to sign his pseudonym autonomously.)

Get  $((p, h(p)) \cdot r^t)^s \equiv_n (p, h(p))^s \cdot r$

Multiply with the multiplicative inverse number<sup>2</sup> of  $r$  and get  $(p, h(p))^s$

The protocol is shown in Figure 6.6, where authentication again gets symbolized by a signet.

Since the confirmation of receipt from [6] is used to prove the receipt of the right in future transfers of that right i.e., the same amount of money, it makes sense to have rights of certain values (similar to real money) which can be combined so that they total the amount to be transferred. Of course  $B$  must to be allowed change money.

To hide the value of the converted receipts for their usage in [2],  $B$  uses a different signature for every value of  $N$ , i.e., a separate pseudonym  $p_{B,N}$ .

The security of the protocol derives from the transparency of the whole process. At all points, each of the three participants possesses enough documents to verify the current

---

<sup>1</sup>The identifier  $p$  for pseudonym is not to be mistake with  $p$ , which stands for one of the two prime numbers used in RSA for key generation.

<sup>2</sup>This will be calculated with the extended euclidean algorithm which is described in §3.4.1.1

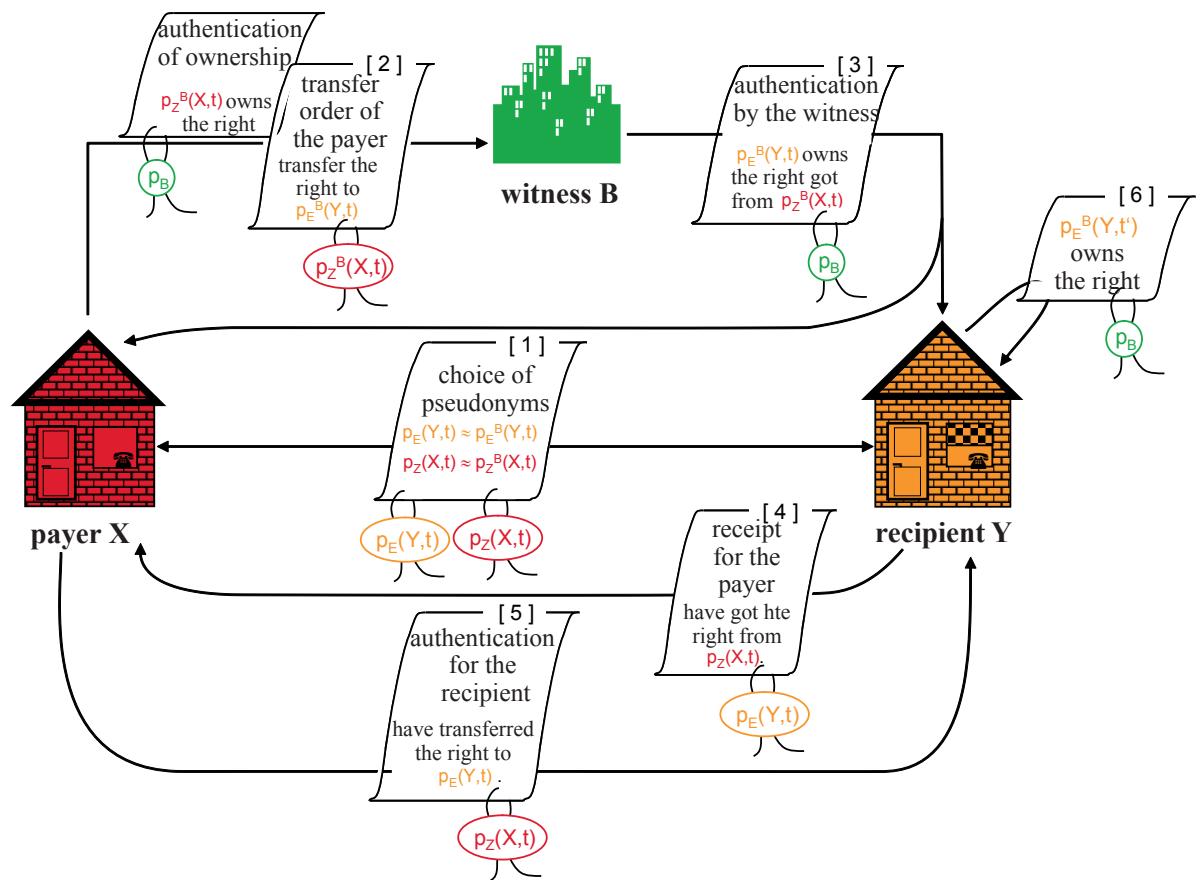


Figure 6.6: Basic scheme of a secure anonymous digital payment system

state of the transfer towards an objective third party, or he is able to set it up by showing around the partner's messages and resending his own (in case the arrival got denied). This applies to completed transactions as well as for transactions still in progress.

Apart from that, claims within transfers that have to be enacted, can only develop for the non-anonymous witness.

If both  $X$  and  $Y$  follow the protocol, then the protocol's anonymity is maximal, because none of them obtains any new information about the partner during the transfer: The witness does learn anything about the use of the pseudonyms that  $X$  and  $Y$  would otherwise use, but only two newly generated ones, which are only used for this particular transfer. Though  $X$  and  $Y$  additionally get to know the pseudonyms they use in communication with the witness, they do not gain new information, because the pseudonyms cannot become linked and they already knew that they would use different pseudonyms in the communication with the witness.

The protocols anonymity being maximal, does not always mean that the anonymity is strong. There are other ways to gain information.

First of all, there are other distinguishing marks for a user apart from the self-chosen pseudonyms, which have to be made known to the partners independent from the applied protocol. In our case this would be the amount and time for a payment. In particular, the costumer is only hidden from all other customers who could possibly own a right of the same value, which gives a reason for only permitting a limited number of fixed values for rights.

On the other side, the payment system is certainly not generating additional anonymity for pseudonyms, which could be used in other situations as well. In case e. g.  $X$  has to prove the transfer,  $p_Z(X, t)$  and  $p_E(Y, t)$  will become linked; this is the goal of the verification.

The users may decide freely how much of their maximal anonymity they are willing to lose in using the same pseudonyms in several transfers or linking them through declarations.

If  $Y$  refuses to hand over the receipt in [4],  $X$  is able to use the confirmation of the witness from [3] and the confirmation of  $Y$  from [1] as a replacement for the receipt. By using this replacement receipt towards a third party (different from  $Y$ ), the witness possibly learns of the assignments  $p_Z^B(X, t)$  towards  $p_Z(X, t)$  and  $p_E^B(Y, t)$  towards  $p_E(Y, t)$ , which in the case of a refusal, slightly restricts the anonymity.

If  $X$  and  $Y$  trust each other and a receipt is not necessary, the receipt of  $B$  confirming the completed transfer for  $X$  does not also have to be sent. Nevertheless, the confirmation of  $Y$  in [1] is still required, because  $X$  first learns about the pseudonym  $p_E^B(Y, t)$  of  $Y$  through this receipt and it has to become authenticated from  $Y$  with  $p_E(Y, t)$  for  $X$  to be sure that the rights are transferred to the correct recipient. Apart from that, in [1]  $X$  and  $Y$  have to agree on payment details, or they have to come to an understanding on the payment, because otherwise errors during the transfer or through  $B$  would not be recognized.

### 6.4.2 Restriction of anonymity through predefined accounts

It might be conceivable that fully anonymous payment systems like the one from §6.4.1 are not desired, because nobody (especially the tax office) will be able to estimate someone's income or property in a system without additional modifications. Same applies to cash. Even in previous non-cash payment systems, the principally available information about persons with several accounts at different institutions, very rarely gets linked.

This could be a reason to restrict users to only one or a fixed number of accounts for carrying out their transactions. This can be enforced by using non-anonymous accounts or by requiring the use of convertible authorizations when opening an account. Even under this condition it is possible to construct a secure payment system according to §6.4.1 which provides as much anonymity for its users as is achievable with these prerequisites.

The condition can be expressed by  $p_Z^B(X, t)$  or  $p_E^B(Y, t)$  representing all transactions performed by  $X$  or  $Y$ , i. e. they are independent from  $t$ . Now the protocol from §6.4.1 is not providing complete anonymity anymore, because  $X$  and  $Y$  exchange their unique account numbers  $p_Z^B(X, t)$  or  $p_E^B(Y, t)$  in step [1] witnessed by  $B$  knowing now the two accounts participating in the transaction.

The solution would be to anonymously introduce changing intermediate pseudonyms for every transfer so that next  $X$  draws the money from his account and transfers it to his intermediate pseudonym. Under this pseudonym he transfers the money to the intermediate pseudonym used by  $Y$ , who then transfers the money to his account. The individual transfers follow §6.4.1, in particular, the authorizations confirming the rights to be received are converted to other pseudonyms between the partial transfers. This gives the payer and the payee intermediate pseudonyms that fit together. Denote

- $p_K(X)$  as the default pseudonym for the account of  $X$
- $p_{ab}(X, t)$  as the pseudonym of the payer  $X$  in transfer  $t$ , with which he draws the money
- $p_{ZwZ}(X, t)$  as the intermediate pseudonym of payer  $X$  in transfer  $t$ , with which he pays  $Y$
- $p_{ZwE}(Y, t)$  as the intermediate pseudonym of payee  $Y$  in transfer  $t$ , with which he receives the right from  $X$
- $p_{ein}(Y, t)$  as the pseudonym with which he deposits the money in his account
- $p_K(Y)$  as the default pseudonym for the account of  $Y$

At first  $X$  chooses  $p_{ZwZ}(X, t)$ , calculates a fitting  $p_{ab}(X, t)$ , transfers the right from  $p_{ab}(X, t)$  to  $p_K(X)$  as the first partial transfer and converts the confirmation for  $p_{ZwZ}(X, t)$ . Steps [1], [4] and [5] of the protocol can be omitted, because  $X$  does not have to inform himself about his own pseudonyms and no receipts are required. The authorization from [2] also does not have to be enclosed, because  $B$  already knows the account balance himself; only in the case of a dispute would the authorization be required.

Now  $Y$  has to choose his pseudonym  $p_{ein}(Y, t)$  and has to calculate the corresponding pseudonym  $p_{ZwE}(Y, t)$ . Thereupon the rights get transferred from  $p_{ZwZ}(X, t)$  to  $p_{ZwE}(Y, t)$  according to the full protocol from §6.4.1 and receipts are created.

After  $Y$  converts the confirmation that states that he has received the right for pseudonym  $p_{ein}(Y, t)$ , he transfers it to  $p_K(Y)$  once again omitting steps [1], [4], [5], and additionally step [6].

The documents issued as receipts by  $B$  in the first and second partial transfer must not look the same. Otherwise  $Y$  would be able to directly use the confirmation from the first transfer for other payments, just bypassing the account. In terms of the convertible authorizations, this implies that  $B$  uses two different signatures (pseudonyms) for withdrawals and transfers. He is able to distinguish between the cases by checking if the provided pseudonyms of the payer belong to an account.

So that  $Y$  is forced to transfer his right to  $p_K(Y)$  within a certain time period (which would be required for e. g. annual taxation),  $B$ 's authorization for a transfer must not authorize a deposit without any time limit.  $B$  should change signing keys used for authentication after a specified period of time and refuse to accept authentications created with old keys after another one. In between the communication partners will have the time to transfer received rights to their account pseudonyms.

Because it is just a specialization, the security of this payment system derives from the security of the one introduced in §6.4.1.

The anonymity is maximal, because the pseudonyms  $p_K(X)$ ,  $p_K(Y)$ ,  $p_Z(X, t)$  and  $p_E(Y, t)$ , which are worthy of being secured, are independent from each other by using intermediate pseudonyms and convertible authorizations. In particular, this means that neither  $Y$  gets to know pseudonym  $p_K(X)$  nor  $X$  gets to know pseudonym  $p_K(Y)$  nor  $B$  gets to know  $p_Z(X, t)$  or  $p_E(Y, t)$  and that all of them are only able to link pseudonyms that they know from previous transfers with newly chosen ones which are never used again apart from one this time, i. e. contain no additional information.

Apart from that, the same notes from the end of §6.4.1 apply to anonymity and receipts. It has to be kept in mind, that the amount and time of a payment allow broader possibilities for linkage here, as in §6.4.1, namely the relation of the two accounts participating in a payment. To avoid linkage, there should be a random period of waiting time between the single partial transfers, and the full amount of a payment should not be withdrawn or deposited at once.

### 6.4.3 Suggestions from literature

The completely anonymous and secure payment systems from §6.4.1 and §6.4.2 are developed by combining elements of previously discussed anonymous digital payment systems. These are best described as weaker variants of the systems above which is why they are only discussed now.

It was DAVID CHAUM, the inventor of the mechanism to convert authorizations, who came up with the idea of using these mechanisms in digital payment systems, even

though he called them something different. Based on the mechanism he developed a series of related payment systems [Chau\_83, Cha8\_85, Chau\_87, Chau\_89].

All proposals made by CHAUM have in common that they work with fixed non-anonymous accounts and disregard receipts. Instead there are variations that, similar to §6.3.1, allow the payer or payee to be made public in cooperation with the partner or the witness.

Another essential difference between CHAUM's payment systems and the system described in §6.4.2, apart from the variation in [Chau\_89, §4.2.1], is that they do not use the option of choosing intermediate pseudonyms in such a way that there exists a digital signature for them. This makes it difficult to guarantee security for the users, especially towards the witness  $B$ , because the holder of a right is no longer able to decide, authorized with his signature, where the right should be transferred to. The simpler variations do not achieve the same level of security at all or only with limitations in anonymity. Another variation [Chau\_89, §4.2.2] at least achieves security for the witness  $B$  by asking  $B$  to sign that he will credit the right only to the payee, before he is presented the document certifying the ownership of the right. For this purpose, the payer hands over the document to the payee. The validity of  $B$ 's signature has to become restricted, because otherwise  $B$  could refuse to credit the right forever with the explanation that he has already certified somebody else, to whom he will transfer the right, but this one did not yet submit the document about the ownership.

The absence of nominal accounts in digital payment systems first appeared in the payment system of anonymous numbered accounts [Pfi1\_83, WaPf\_85], which otherwise exactly reproduces the usual transfers with digital signatures. Basically, this anonymity is not a quality of the payment system, because it is only telling something about linkage possibilities of pseudonyms in their environment and not about security in general (assuming overdrawing is prohibited). Generally, each strictly secure digital payment system with nominal accounts would be secure for anonymous number accounts too.

The usage of transaction related pseudonyms in communications with a credit institute ( $p_Z^B(X, t)$  and  $p_E^B(Y, t)$  according to the protocol above) was suggested in [Bürk\_86, BüPf\_87, BüPf\_89]. This payment system corresponds to the one in §6.4.1 without the conversion of signatures, which does not restrict its security, but its anonymity, even if only to a small account. This happens because for each transfer  $t$  the pseudonym  $p_Z^B(X, t)$  equals the pseudonym  $p_E^B(X, t')$  from a previous transfer  $t'$ . So the witness  $B$  is able to realize that both times the same person is involved. If the payer  $W$  in transfer  $t'$  and the payee  $Y$  in transfer  $t$  work together (which is even more likely when  $W$  and  $Y$  are the same person), even they could realize this and can link the actually secure-worthy pseudonyms  $p_Z(X, t)$  and  $p_E(X, t')$ .

This system would remain left over in case the mechanism for convertible authorizations from §6.2.1.2.2, which at the moment can only be implemented with the specific cryptographic signature system RSA, is broken, but the other systems for digital signatures are still secure.

This is not completely improbable, since it is possible to prove for at least some of the systems that it is as difficult to break them as it is to solve mathematical problems

which have been known as difficult to solve for a long time.

#### 6.4.4 Marginal conditions for anonymous payment systems

The utilization of an anonymous payment system should not bring about a comparison of the disadvantages to traditional payment systems, in particular, a user should be able to

- freely choose between several credit institutes as witness for his payments,
- transfer money randomly, especially into a conventional payment system outside the open system and vice versa,
- withdraw money for interest anonymously and
- anonymously borrow money at interest.

Beside the demands of users towards the provided services, demands of credit institutes and the state have to be considered, e. g. anonymous payment systems may not hinder capital and income dependable taxation.

In this context especially, the limits of useful anonymity have to be considered. For this text they are presumed to be outside the open digital system and are therefore not discussed in detail. Without this discussion it is not possible to present useful statements about the sense and possibility for such a thing as a taxation in spite of anonymity.

##### 6.4.4.1 Transfer between payment systems

In the payments systems described in §6.4.1 and §6.4.2 it was assumed that only one witness, i. e. one credit institute, is involved. From the point of view of economic policy and anonymity, payment systems should allow the payer and the payee to use different credit institutes and even different payment systems.

For a payer and a recipient of a payment using the same anonymous payment system, but different witnesses, the recipient  $Y$  has to appoint a witness  $B_E$  he trusts in step [1] of the protocol in §6.4.1 apart from choosing a pseudonym  $p_E^B(Y, t)$ . Furthermore, in step [2] the payer informs his witness  $B_Z$  about the witness  $B_E$ . Because  $B_Z$  and  $B_E$  are not anonymous to each other they are now able to cooperate and can be seen as single witness  $B$  where all communication between  $B$  and  $X$  is performed through  $B_Z$  and all communication between  $B$  and  $Y$  through  $B_E$ .

The transfer between a non-anonymous and an anonymous payment system can be realized easily: The provider of the non-anonymous payment system, a bank, acts as a mediator which can act in both payment systems. The transfer between two users  $X$  and  $Y$  would become divided into two separate transfers between  $X$  and the bank and the bank and  $Y$  respectively. The same works if a user wants to transfer his money into another payment system without revealing his identity in the anonymous system.

#### 6.4.4.2 Interest on the balance and accommodation of loans

For both payment systems in §6.4.1 and §6.4.2 the rights witnessed by  $B$  can be treated as a sight deposits managed by a bank  $B$ : The holder of a right can access it at any time, but  $B$  can detect every access. This leaves only a weak motivation for the bank to pay interest, which could lead to a combined approach where fees become set against interest.

Both payment systems could easily be extended so that credit can be instated for a specific period of time, making interest sensible for banks: this is trivial for fixed accounts; without fixed accounts the witness could use different signatures  $p_B^{z1}, p_B^{z2}, \dots$  to define different expiration dates.

To allow specialized types of investment dependent on the investment capital, the capital has to become organized in units that can be indicated by the bank. This is best achieved by using the payment system from §6.4.2 with fixed accounts, where interest can be payed as it is usually done these days.

When renouncing these kinds of investments and using the payment system from §6.4.1, each single right has to be treated as a single account, where the opening date of the account can be determined based on the witness' signature. The interest for a right can be payed by increasing its value or by paying off the interest at each transfer [Chau\_89].

The granting of loans and interest of credit is in principal no harder than it is today. The following applies to the coverage of a loan, stated in §6.2.5: If the one who takes out a loan wants to stay anonymous, he needs to commit something as security; whereas if he identifies himself towards the bank, he needs to transfer the money to himself once (e. g. to his second account) in favor of anonymity and afterwards he can work with the money as if it is usual capital assets.

As already indicated in §6.2.5, charges for account maintenance and other services provided by a bank are not much more difficult to bill than it is today: In the case of fixed accounts, the bank can simply debit the demanded amount from the balances it manages. For payment systems without fixed accounts, the banks have to meet their demands during each transfer. The payer includes a fee transfer order (digital stamp) for the witnessing bank in each transfer order.

#### 6.4.5 Secure devices as witnesses

It is easy to design an anonymous payment system which does not provide anonymity towards the witness: The protocol from §6.4.1 can be altered in a way that user  $X$  always chooses the pseudonyms  $p_Z^B(X, t)$  or  $p_E^B(X, t')$  for all payments whether he acts as payer or payee. He corresponds a fixed account number to a business relations pseudonym.  $B$  learns the pseudonyms  $p_Z^B(X, t)$  and  $p_E^B(X, t')$ . This saves the problem of replacement receipts in [4] and in [5] as well as the transformation of the confirmation in step [6].

In order to justify the assumption that the witnesses is trustworthy in regards to the anonymity of the user, the witness must, in general, be selected as a safe, unanalyzable

device, (cp. §6.2.1.2.2).

There are two variants which could be used:

The first variation uses a single central secure device, which witnesses all transactions.

Already, based on the general reasons listed in §1.2.2, this variant is less desirable. Additionally, there is no real advantage towards the untrustworthy witness from §6.4.1 and §6.4.2. Though the protocol becomes very simplified, as mentioned above, this only unburdens the computers, not the users of the payment system, and does not reveal any new possibilities.

In the second variation, each user gets their own secure device as an "electronic wallet" for witnessing (on behalf of the provider of the payment system) his transactions.

The usage of secure devices as electronic wallets was suggested in [EvGY\_84, Even\_89], but was due to the (apparently necessary) protocols, not yet anonymous. Anonymous versions of this procedure can be found in [BüPf\_87, BüPf\_89].

The only real advantage of electronic wallets (anonymous or non-anonymous) is the support of **off-line transactions** to enable a user to pay and receive payments spontaneously without being forced to contact a central coordinating instance in between. Because the systems discussed are considered to be open systems based on a communication networks, this is not a big advantage.

Much more problematic are the disadvantages (-) of electronic wallets:

- Because payments are not performed through a central witness, losing an electronic wallet would eventually lead to the loss of already received payments. To prevent a loss, relatively costly error tolerance measures have to be taken [WaP1\_87, WaPf\_90].
- The security of payment systems with electronic wallets considerably depends on the possibility of analyzing such devices. As already discussed in §6.2.1.2.2, the existence of durable secure devices is questionable and the evaluation of real existing devices, which are assumed to be secure, is not feasible.
- Using secure device does not mean that cryptographic techniques used, in particular, for mutual authentication of electronic wallets are not needed anymore. Hence, such secure devices will not be an alternative to encryption systems or authentication, should these be broken one day. In case only all asymmetric encryption systems and authentication are broken, it would make sense to use these secure devices in combination with symmetric cryptographic systems, because this would be enough to secure communication between the single devices.

When the disadvantages listed above and the few advantages of electronic wallet in open digital system of the kind described, their use does not seem to be sensible.

### 6.4.6 Payment systems with security based on the possibility of making someone public

Another payment system was introduced with [ChFN\_90] which only meets a weaker definition of security as the one used before:

- A user must not forward a right he received more than once. However, if he still forwards it multiple times, then he must face becoming public

The basic idea is the same as in §6.3.1: One hopes that after someone is made public his property is sufficient to meet the demands arising from compensation claims.

Despite of its weaker security definition, a short outline of the system will be given: On the one hand, the system is not more insecure as e. g. credit cards nowadays. On the other hand, this system seems to be the only alternative to the questionable electronic wallets from §6.4.5. The main difference to the payment systems mentioned above is that paying partner  $X$ , instead of the witness  $B$ , confirms the transfer for the recipient of payment  $Y$ . If  $X$  acts correctly, his anonymity will be preserved. If  $X$  acts incorrectly in transferring the right to another recipient  $Y^*$ , due to a cryptographic trick, the two confirmations for the transfer could be combined to reveal the identity of  $X$  with a high likelihood. In case the right is presented to  $B$  multiple times,  $B$  would be able to make  $X$  public.

*example:* Main idea: We denote

- $k$  as the security parameter,
- $r_i$  as randomly chosen ( $1 \leq i \leq k$ ),
- $I$  as the identity of the issuer of the bank notes,
- $C$  as a commitment scheme with informational theoretical secrecy.

A blindly<sup>3</sup> signed bank note is defined through

$$s_{bank}(C(r_1), C(r_1 \oplus I), C(r_2), C(r_2 \oplus I), \dots, C(r_k), C(r_k \oplus I))$$

The recipient of a bank note decides for  $i = 1, \dots, k$  independently for each if he want to have  $r_i$  or  $r_i \oplus I$  disclosed, so he gets  $k$  commitments disclosed. (Through this one-time pad the identity of the issuer is protected with informational theoretical security. The bank is due to the commitment scheme only complexity theoretical secure. Therefore, it may choose the security parameter.)

For a bank note distributed to two good recipients, the probability that at least one  $r_i$  and  $r_i \oplus I$  for an  $i$  arriving at the bank is in  $k$  exponentially close to 1 and the issuer can become identified.

---

<sup>3</sup>Before a bank blindly signs a bank note it has to check if the presented note is correctly constructed and in particular if it contains the identity  $I$  of the issuer of the bank note. In the case where the bank does not perform this check, the issuer could include none or a false identity, which would enable him to avoid his identity to be made public in case of multiple passing on of a right.

For the check the bank chooses a number of samples to be disclosed by the one who handed them in and only if these do not contain a attempt of cheating will the bank sign the following samples blindly [ChFN\_90]

Because this system does not depend on a witness it can be used for off-line transactions; this was the purpose suggested for it in [ChFN\_90].

To prevent a payer, who got to know about the choice of the payee, from forwarding this information to a second payee (who works with him together), DAVID CHAUM suggested that the identity of the payee should partially define the decisions of the recipient of a bank note. Now a bank note cannot be cashed twice without its issuer loosing his anonymity, because this variation guarantees for two different recipients choosing differently at least once.

To increase the security this can be combined with secure devices: The users have to use secure devices instead of usual computers to prevent multiple passing down of rights. The combined system fulfills the higher security definition as long as secure devices are really secure. Otherwise the security is the same as in the system without secure devices.

Payment systems without off-line capabilities where the payee cannot pass down a right without contacting the bank before are described in [ChFN\_90, Cha1\_89, CBHM\_90]. (According to DAVID CHAUM, an adequate system suggested in [OkOh\_89, OkOh1\_89] has been proven incorrect.)

## Recommended readings

Fraud secure value exchange: [AsSW\_97, Baum\_99]

Digital payment systems: [AJSW\_97, Chau\_92, PfWa2\_97]

Current research

Computes & Security, Elsevier

Congress Volumes ACM Conference on Computer an Communication Security, IEEE Symposium on Security and Privacy, IFIP/Sec, VIS, ESORICS, Crypto, Eurocrypt, Asiacrypt; Congress Volumes get published from ACM, IEEE, Chapmen & Hall London, in the series "Datenschutz Fachberichte" Vieweg Verlag Wiesbaden, in the series LNCS of Springer-Verlag Heidelberg

Links to recent research and comments to recent development:

<http://www.itd.nrl.navy.mil/ITD/5540/ieee/cipher>  
or subscription to cipher-request@itd.nrl.navy.mil

Datenschutz und Datensicherheit DuD, Vieweg-Verlag



## 7 Credentials

An overview for credentials is given in [Cha8\_85][Chau\_87]. However, it is not mentioned there that no organization can provide the credential signatures due to the common RSA module. They need to be provided by a signature organization, since whoever knows a pair of inverse RSA exponents, can factorize the modulus [Hors\_85, pg. 187][MeOV\_97, pg. 287].

The overview describes a case of a permanent and initially known set of credentials.

A detailed protocol description can be found in [Chau2\_90].

[ChEv\_87] describes how to conduct the validation if pseudonyms are correctly created, so that the probability of unidentified falsifications becomes exponentially small.

[Cha1\_88] shows how to display credentials that are not known right from start.

[Bura\_88] and [ChAn\_90] give additional types of credentials.

Unfortunately, I do not know of any application of these purely digital identities, as it is not sufficient that authorizations such as a “driver license” can only be converted between a person’s pseudonyms. The authorization “driver license” should also only be usable by the biological entity that passed the test. Consequently you have to extend the shown procedure for (almost?) all applications so that pseudonyms are coupled to body characteristics within the protocol part “arrange pseudonyms”. If there are not enough linkable body features, but enough that can be measured with precision, then the question lies outside the field of computer science. If so, then the extension of the shown protocols are canonical.

Described in [Bleu\_99] is an extension based on secure and therefore biometric hardware (cp. §6.2.1.2.2) that works with a *single* precisely measurable body characteristic.

## *7 Credentials*

# 8 Multiparty Computation Protocols

Let there be a common task, that  $n$  participants  $T_i$  want to perform a computation together. Every participant shall contribute its input  $I_i$  and receive the output  $O_i$  which is a function  $f_i$  of the input. These security properties should be achieved:

- Greatest possible **confidentiality**:

No participant should learn more about the input of other participants than he can ascertain from his own output.

- Greatest possible **integrity**:

No participant should be able to influence the output content after sending in his input.

- Greatest possible **availability**:

No participant, after sending in his input, should be able to disable other participant's ability to receive their own output.

All three security properties should also apply if *several* ( $n - 1$ ) participants consolidate:

- They should be restricted to learning nothing outside of what they can learn from their combined output.
- The last participant may no longer alter the output after sending in its input or withhold it from the others.

There are two completely different ways to achieve this:

1. All participants send their input to a **central computer** which everybody (must) trust. This central computer performs the computation and sends every participant their specific output. Communication between the computer and the participant is **encrypted** to guarantee concealment and authentication (cp. fig. 8.1). If the encryption and the central computer as well as the availability of the communication network are trusted, then the problem is solved. In regards to the availability and integrity of the output, they can, at least in principle, be controlled in retrospect if all messages were digitally signed and a cooperation between other participants and the central computer is ruled out. Otherwise they could "get" the necessary signatures for their actions (??). Unfortunately the central computer is nearly uncontrollable in regards to its trustworthiness (cp. §1.2.2 and §2.1).

2. The participants conduct a **Multi-Party Computation Protocol** (cp. §8.2). For such a protocol there is no central instance which everybody must trust. But even here, realistic trustfulness needs:
  - Either it must be trusted that an attacker comprises less than a third of all participants, as then information theoretical computation protocols are known ([BeGW\_88],[ChCD1\_88]).
  - Or you have to work with a cryptographic assumption in which the amount of attacking participants is irrelevant ([ChDG\_88]).
  - A newer protocol only requires that one of the two shown restrictions apply to the attacker ([Chau\_90]).

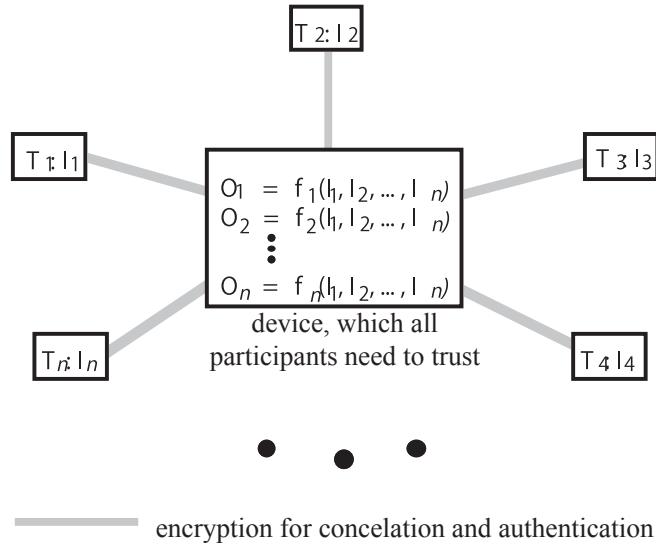


Figure 8.1: Centralized computation protocol between participants  $T_i$ . Every  $T_i$  sends an input  $I_i$  to a central computer everybody trust. It calculates the function  $f_j$  and sends every  $T_i$  its output  $O_i$ .

Roughly speaking, the calculation protocols work like this:

- The input is distributed among the participants by a threshold scheme (cp. §3.9.6).
- After this, parts may be added as well as multiplied to conduct any desired calculation.

The exact details would go beyond the scope of this script. What is important is the message that principally *everything* that is calculated centrally (with trusted computers) can be calculated decentrally and without global trust - even though the efforts with the distributed computation protocols of today are relatively high.

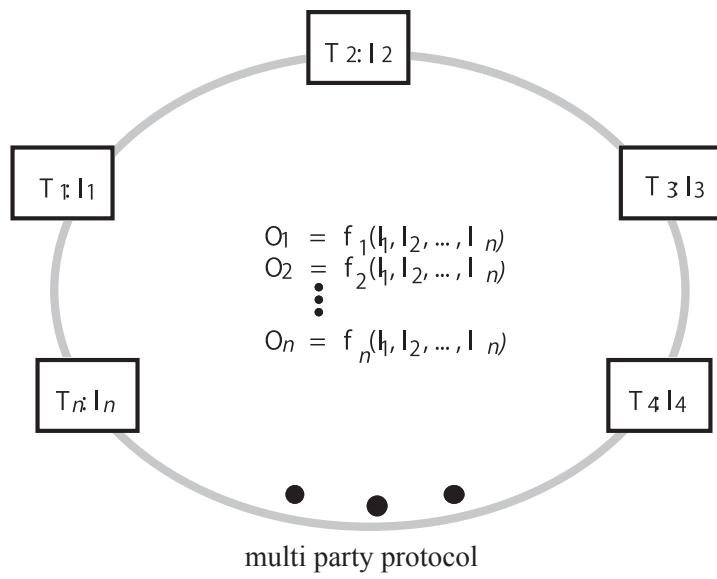


Figure 8.2: Distributed computation protocol between participants  $T_i$ . Every  $T_i$  sends in its input  $I_i$  and receives its output  $O_i$ .



## 9 Controllability of Security Technologies

None of us live completely on our own. You are reading my text now - and I myself know I am too. Besides, computer networks, as well as multilateral security, anticipate *several* participants. And these participants do not only have their individual intentions, but are also part of very different communities. Not only individuals have legitimate rights and claims towards their community, but also communities towards their members. This creates the following problem:

The security technologies presented here can be used to create democracy promoting multilateral security within an open society. They can also be used to build subcultures that bar themselves from the outside world. In the worst case, organized criminal elements could use the wildly spread security technology to plan, coordinate, and conduct their crimes. To prevent this, some politicians demand regulation of security technologies, or in other words, a demand for limiting the creation, distribution, and even usage of security mechanisms.

I don't want to try to decide what what is necessary for society, nor what is relatively good when fighting the bad. These are difficult questions that have no final answer: Democracy also often starts as the matter of some subcultures. Even a democratic superpower such as the US has strategical and economical interests that are supported by their regulation of security technologies.

I will go into a simple matter: Can security technology be regulated at all? Do export and import restrictions, or laws enforcing limited usage and capabilities, help or are they counterproductive?

I do not think much of **symbolic** politics that limit the capabilities of security technology for the law-abiding user without significantly hampering criminals. It would be better to issue a symbolic law against "malicious thoughts", since all criminal activities start with malicious thoughts - and at least no good people would be hindered through such a law, .

In the past two decades many attempts were made to limit the spread of **secure computers** (particularly secure operating systems) and **cryptography**. For example, the US-based DEC corporation discontinued the development of secure and distributed operating system ([GGKL\_89],[GKLR\_92][Wich\_93]) in 1994 because of the high risk of not receiving an export license. And they estimated that the costs were too high to develop two versions (a really secure one and a lesser secure one) in parallel. Starting in 1993 (it has started behind closed door already in 1986 [WaPP\_87][Riha2\_96]), the US has been officially discussing the regulation of cryptography (keyword *Key-Escrow Key-Recovery* as well as *Clipper Chip*), and in Germany the *Kryptogesetz* was being discussed publicly ([HuPf2\_96][WiHP\_97][HaMö\_98]). The spread of insecure computers is today well ad-

vanced. Just think about the PCs with operating systems lacking user access control, which makes them more susceptible to hosting worms and Trojan Horses.

Below, I will concentrate on the regulation of cryptography - since a lot of damage can arise not only to individuals but also to democracy due to ignorance and overzealous politics ([Pfit\_89]).

Discussed are the main requirements and technical design options based on the functional principals of cryptographic and steganographic techniques (cp. §3,§4), which are very important for the discussion of cryptographic regulation.<sup>1</sup>

## 9.1 User requirements on signature- and encryption keys

Digital signature systems are currently being introduced for legal affairs<sup>2</sup> (for the electronic communication involved) and for electronic archiving. Both areas require certain certification requirements of the signature keys. This is a prerequisite for the legal binding of digitally signed documents (along with the cryptographic properties of the signature system). Therefore the recipient must not only be able to verify the signature of the message with the proper public verification key, but also unambiguously assign the verification key to a real person<sup>3</sup>. Therefore these so called *Trust Centers* (institutions that provide certain security services) should also provide the functionality of a Certification Authority. The creation of the signature key-pairs in Trust Centers<sup>4</sup> can be made a prerequisite for participating in electronically transmitted legal affairs. This presents the opportunity to store all secret signing keys there. The applicability of signature systems within legal affairs is based on the assurance that, except for the holder of the secret key, nobody else can create valid signatures. Access to the secret key allows evidence modification and therefore devalues the whole security infrastructure.

Concealment key-pairs do not have to be legally certified. It may be useful to employ key certifications to guarantee the authenticity of the keys managed. This can prevent persons from providing public encryption keys under false names within the security infrastructure, which would otherwise enable the attacker to obtain all the data within that was encrypted with this key. But the user's demand from a encryption system is that they *mutually* trust the authenticity of each other's public encryption key, not that

---

<sup>1</sup>The following sub-chapters are copied from [HuPf\_96][HuPf\_98]

<sup>2</sup>The law on digital signatures in Germany (Art. 3 of the Informations- und Kommunikationsdienstes-Gesetzes (IuKDG) went into effect Aug. 1st, 1997.

<sup>3</sup>Therefore the public key and the identity of the owner are digitally signed by an *Certification Authority* which is the name *certification* of the public key. To verify such certificates, the receiver may need to obtain further certificates. The resulting certification tree is described in X.509 and partially in the legal text for the signature law.

<sup>4</sup>But this is not recommended since central component where different and independently running partial tasks are concentrated, tend to be weak points in the security architecture ([FJPP\_95]). Besides, it does not ease the certification.

## 9.2 Digital signatures allow the exchange of encryption keys

they can prove this in court. As soon as communication partners have exchanged<sup>5</sup> keys with credible authentication, they may use any other confidential communication system desired. The creation of encryption key-pairs in Trust Centers can not be enforced, as nobody is asked to rely on certificates of a Trust Center.

In the following, we will argue that users may conduct a key exchange on their own without any drawbacks regarding the requirements. The exchange is meant to be confidential towards any third person while using the security infrastructure for legally binding issues.

## 9.2 Digital signatures allow the exchange of encryption keys

Every security infrastructure for digital signatures allows a secure exchange of encryption keys: With the known algorithms [Schn\_96] that are already implemented in widely available programs (“Pretty Good Privacy” - *PGP*) everybody can generate encryption key-pairs. Afterwards everybody may certify their own public key to make the authentication verifiable, by signing the message “The public key ... belongs to ...” (message  $s_A(A, c_A)$  in fig. 9.1). Now the security infrastructure for digital signatures allows everybody to validate the integrity of the message<sup>6</sup>. Finally everything encrypted with the public encryption key can only be decrypted by the owner of the key (if integrity is assured). If this use of signatures systems is ruled out, then the only option is to undermine the security of the signatures by issuing digitally signed messages where the signer is *not* the owner. Eventually this would make any legal binding of documents signed within this system, invalid. Concealment keys that were distributed with help of

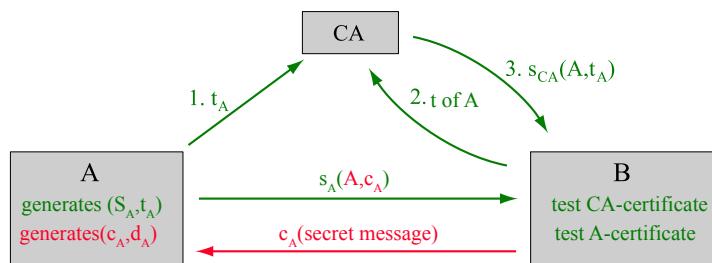


Figure 9.1: Secure digital signatures allow autonomous and secure concealment.

an security infrastructure for digital signatures allow the exchange of confidential messages, as well as keys for symmetric encryption systems. They can be used for efficient concealment.

<sup>5</sup>For example with help of government run security infrastructure or by personal exchange.

<sup>6</sup>by testing the certificate  $s_{CA}(A, t_A)$  of the certification authority as well as the certificate the participant A has produced.

### 9.3 Key-Escrow Systems can be used for an undetectable key exchange

Similar to using secure digital signatures for exchanging encryption keys, every system, in particular Key-Escrow or Key-Recovery systems in which intelligence services or other organizations have access to the secret key, can be used to negotiate keys for the use of additional encryption systems (9.2). Instead of passing the confidential messages directly to the encryption system, a redundantly encoded public key is transmitted. The receiver checks the key, similar to checking the certificate of digital signatures. As long as he does not detect a corruption of the public key, he either uses the public key to encrypt the message (9.2 2nd arrow) or for exchanging a further key for symmetric concealment (9.2 4th arrow). Next the messages encrypted with the key of the overlying encryption system are encrypted with the key of the underlying Key-Escrow system (9.2 3rd and 4th arrow). If the Key-Escrow is designed so that it can not or is not allowed

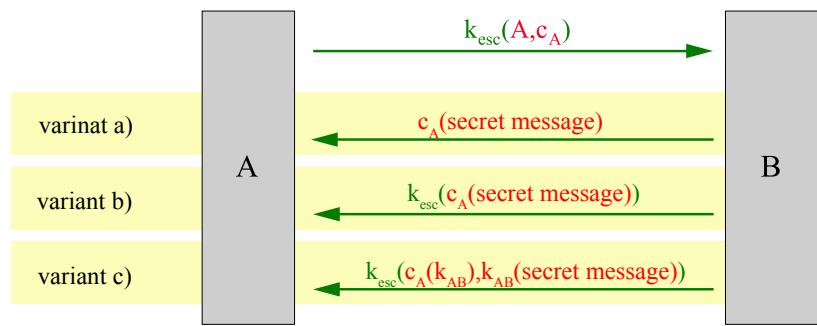


Figure 9.2: Key-Escrow concealment without permanent monitoring allows exchanging keys and therefore a) “normal” concealment, b) asymmetric concealment without Key-Escrow and c) symmetric concealment without Key-Escrow.

to decrypt earlier messages, then you will not have to take the detour over the public encryption keys. As seen in 9.3, a symmetric key can be exchanged directly - but this has to be done at the right time. The usage of additional encryption measures can only be

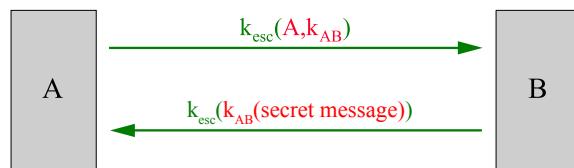


Figure 9.3: Key-Escrow concealment without retroactive decryption allows direct concealment without Key-Escrow.

discovered when the keys of the Key-Escrow system are revealed and used for decrypting telecommunications, which, due to current laws, would require legal permission. But Key-Escrow is - if issuance of the keys is limited as proposed - almost useless for the investigations of the Verfassungsschutz (German for a state run organization for the defense of the constitution), Bundesnachrichtendienst (federal intelligence service) and the MAD (military intelligence service). In particular, those cases where Key-Escrow mechanisms should be decrypt messages, additional encryption systems will probably be established above the Key-Escrow mechanisms. Therefore prohibition of any other encryption systems is also being discussed, though that at all be helpful: If there is only cautious surveillance of telecommunications, violations will be noticed too late. But if serious efforts are made, then the law of telecommunications secrecy will be substantially violated. And despite all the attempts of surveillance, users can still employ steganographics (§9.5), where the usage of crypto-systems is almost undetectable.

Aside from that, you should be aware of the fact that all people and organizations that have knowledge about the secret encryption key may decrypt all corresponding messages. In contrast to traditional wiretapping techniques, permanent access is possible, since the knowledge about the key cannot be erased. The central storage of such sensible information at the Key-Escrow agencies makes them a highly "attractive" point for business espionage and other criminal activities.

In short: Key-Escrow will stay mostly ineffective in the battle against crime - but requires barely feasible technical<sup>7</sup> and considerable organizational<sup>8</sup> efforts to keep the erosion of citizen's rights and the increased vulnerability of economy and society under a certain level.

## 9.4 Symmetric authentication allows concealment

Every bit of the message to be concealed is transmitted by sending 0 or 1, followed by a MAC ([Rive\_98]). The MAC of the correct bit fits, while the false one does not (cp. 9.4). Since no one except the sender and receiver can check which MACs fit and which do not<sup>9</sup>, the message is perfectly concealed.

Rivests proposal is a little more than twice as expensive as the solution from exercise 3-27. But it has the advantage that both bits with MACs can be put into separate messages with equal sequential numbers. Then on the one side, the authentication protocol of the receiver throws away the false bits, so that the receiver will not have to implement anything. Additionally the sender will not have to create the false bits and

---

<sup>7</sup>How can you implement a correctly working access control to the Key-Escrow databases but allow only short-term access to the keys? Experience shows that *every* large technical system contains bugs which puts the secrets of many people and organizations at risk.

<sup>8</sup>It is certainly interesting that a lot of people accuse Key-Escrow of involvement in scandals and successes based on the motto "Every man has his price".

<sup>9</sup>If someone could do better than with a probability of 0.5 it would be an basic approach towards breaking the authentication systems.

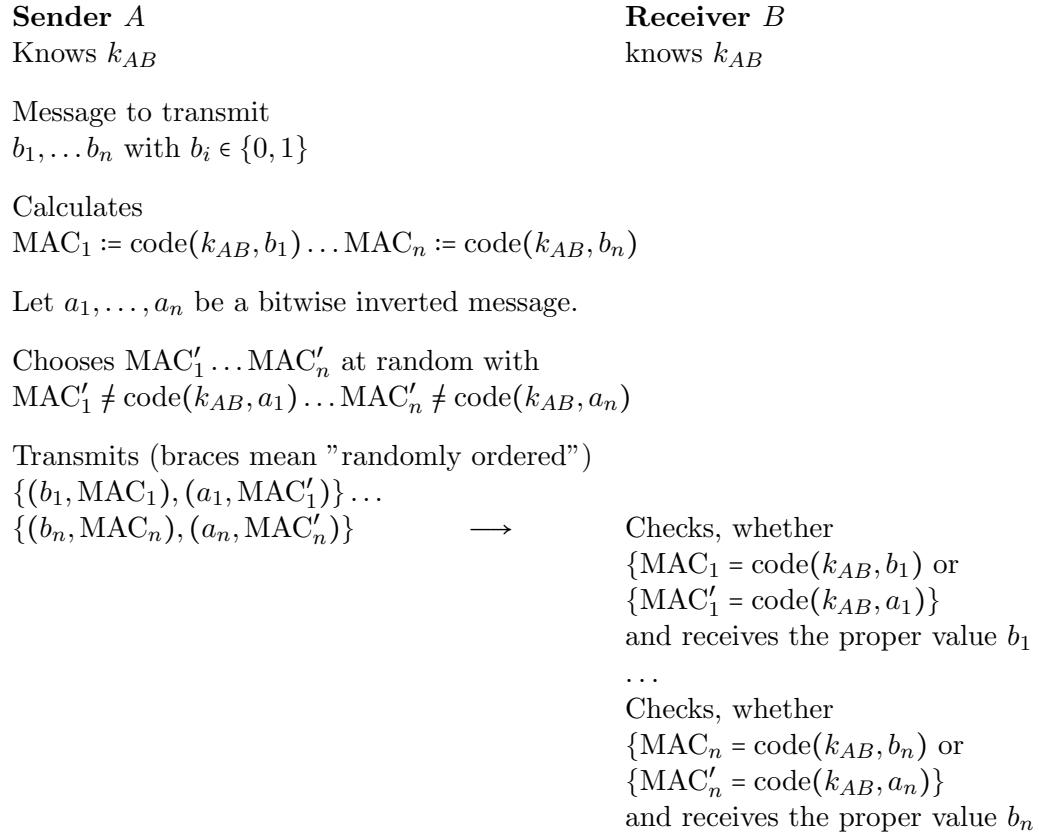


Figure 9.4: Concealment through symmetric authentication by Rivest.

MACs and put them into the message at all - this can be done by someone else as long as it is done in front of the area the attacker is observing. And to do so, he does not need the symmetric authentication key - he just uses random MACs, which will then, with a high probability, not fit(cp. 9.5). Finally sender and receiver could argue, even in a country with a total ban on concealment: "Concealment? We did not do it, we had no intentions to, and we did not even notice it!" As mentioned in the solution for exercise 3-27, more than one bit may be authenticated with a MAC. However it requires  $2^l$  bit values and MACs as well as test steps with  $l$  simultaneously authenticated bits per MAC. For here this is only a disadvantage, while for exercise 3-27 this may turn out to be an advantage.

An interesting question: What if somebody (like a prosecuting authority) forces you to hand over the authentication key? The answer is: Prevention. This means having a bit sequence among the multitude, with unsuspicious or pleasant content, for "them" - and only handing over this key. To make several bit sequences with "real" content protecting each other, some more tricky encodings of the bit sequences are necessary than in [Rive\_98].

## 9.5 Steganography: Finding confidential message is impossible

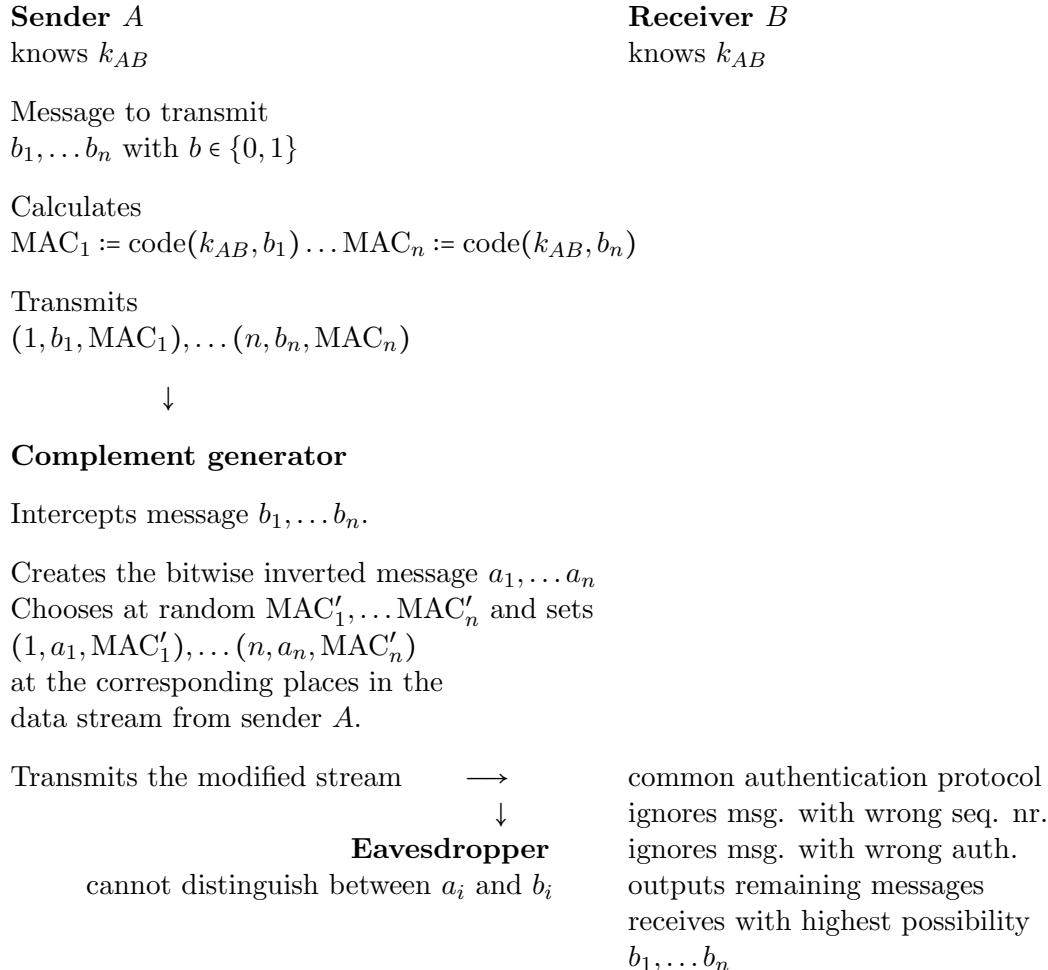


Figure 9.5: Concealment through symmetric authentication without participant activity

## 9.5 Steganography: Finding confidential message is impossible

Messages can not only be protected by cryptographic encryption systems, but also with steganography (cp. 4): The message to be protected is hidden within another message (which is necessarily longer). Even if cryptography - or in general any kind of uncontrollable communication or storage - were to be prohibited, you would not be able to prove anything if the steganographics employed is good enough. If it is not good enough, every evidence of cryptographics that stands trial can be directly used to actually improve it(cp. 4.1.3.3).

Steganography makes use of the fact that for every human communication the same

message may have different meanings for different receivers. With modern digital communication networks you can execute efficient techniques for exchanging messages without having to meet in person and without a third person observing. Of course using steganography to additionally encrypt the message to be transmitted, is also possible and recommended for improving security.

Data intensive telecommunication applications (mainly multimedia) are perfect carriers for steganographic techniques.

## 9.6 Measures to take after key loss

The loss<sup>10</sup> of keys used to create confidentiality or integrity for the *transmission* of messages poses no tough problem for the user: If the data sent can not be decrypted, then the receiver creates a new key-pair and re-informs the sender about the new authenticated public key. Afterwards, the sender encrypts the message again, but with the new key, and resends it. This method is much less expensive than any secure method of key backups<sup>11</sup> and much more secure than Key-Escrow and Key-Recovery systems. Those systems actually intended for surveying telecommunications are redundant as key backups.

The same applies to signature keys (cp. 9.6). If an owner loses his secret signing key he can not sign anything that would also fit for his public key, but he can always generate a new key-pair and get them certified by a CA. The receiver must be re-informed about this to use the new public key.

The loss of public keys is almost irrelevant since their public status allows widespread storage.

If the secret decryption (confidentiality) or test (protecting or the symmetric authentication of long term storage data<sup>12</sup>) key are lost, you will probably lose the data or its verifiability. The key's owner should take precautions such as key backups or Key-Escrow.

Key backups or Key-Escrow are only useful for applications where data can not be retrieved a different way, since the data's security will be harmed in all cases if somebody besides the data owner has a key (or parts of one). For archiving long term data, key backup techniques can be useful, taking the users' requirements into consideration. Every key backup should be designed to help the owner immediately take notice of a

---

<sup>10</sup>We will only consider situations where the key is completely lost. If the key came into unauthorized hands further measures must be applied, such as using blacklists from the authorities that have certified the key. Before using a key the user needs to ask whether the key is blacklisted.

<sup>11</sup>For this you can split your own key with a threshold scheme (cp. 3.9.6) and distribute the parts to seemingly trustworthy friends, organizations, etc. For reconstruction, a certain minimal amount (defined by the owner prior to the splitting) of parts is necessary. By defining an appropriate minimal amount, the few less trustworthy partners can not reconstruct the key, and the loss of a few part will not hinder reconstruction.

<sup>12</sup>For archiving purpose such as land registry or in the public health system.

	protection of communication	long term storage
concealment	key backup for functionality not needed, but additional security risk	key backup sensible
symmetrical authen- tication		
asymmetrical (digital signature)		

Figure 9.6: Where is key backup useful, unnecessary or an additional risk?

key reconstruction if he is not directly involved. Only this would guarantee that the owner has full control over his data. This is a very important first line of defense against misuse ([AABB\_97, Cap. 2.3]). However this requirement is *not* fulfilled by Key-Escrow systems.

To make Key-Escrow look like a solution for the user's key backup problem is not a serious matter: Two independent problems based on different security interests should be left divided and solved individually.

A deeper look into the possibilities of Key-Recovery and its risks can be found in [Weck\_98], [Weck1\_98], which contains the resulting recommendations.

## 9.7 Conclusions

Cryptography allows everybody, individual or organization, to efficiently protect the confidentiality and integrity of their information in areas without physical access. Without cryptography, thousands of curious system administrators could read the email traffic and intelligence services could eavesdrop on telecommunications for economic espionage. Without secure cryptography, information can be randomly manipulated.

The decentralized nature of global digital communication has deprived control and protection capabilities from the government of a single state. Cryptography gives back the ability for every individual to protect their own information by themselves. Within

an information society, cryptography concerns us all, since everybody is affected by its (in)security!

The usage of cryptography becomes obviously securer if users create the necessary keys through their own devices. Due to the technical facts presented here, the often stated argument that the generation and storage of these secret keys is a state affair that must be implemented by state-run Certification Authorities to effectively fight crime, is unsubstantial. The argument that this would serve the user's interests is, if examined accurately, inappropriate. This applies to Key-Escrow as well.

Regulating cryptographic techniques for concealment with the intent of crime fighting is not useful since you could not enforce it effectively<sup>13</sup>. It would affect the security interests of the people, economy, and society. The laws in the Signaturgesetz supporting regulation of cryptographies for securing integrity and evidence eligibility of digital documents are welcomed.

### Reading recommendations

- Further literature on 9.6: [AABB\_98]
- Discussion on the crypto controversy in Germany: *Wortprotokolle des Wiesbadener Forums Datenschutz am 19. Juni 1997*:[HaMö\_98]
- A current press overview: *Nicolas Reichelt: Verhindert ein Kryptographie-Gesetz*; <http://www.crypto.de/#press>
- About legal cryptography regulation attempts:
  - European Cryptography Resources  
<http://www.modeemi.cs.tut.fi/avs/eu-crypto.html>
  - Bert-Jaap Koops: Crypto Law Survey  
<http://cwis.kub.nl/frw/people/koops/lawsurv.htm>
  - Ulf Moeller: Rechtliche Situation, politische Diskussion  
<http://www.thur.de/ulf/krypto/verbot.html>

---

<sup>13</sup>[HaMö\_98, pg. 90] brought up the argument all the crypto regulation is not about the users of PCs, but of telephones. This is obviously not true since today's cellphones can already be programmed or performance hungry multimedia applications can be run on a laptop. Then they can be used like normal PCs and the cellphones could use PGP to encrypt all communications ([HaMö\_98, pg. 121f]).

## **10 Design of Multilateral Secure IT Systems**



# 11 Glossary

a	Länge der Ausgabeeinheit
$\Leftrightarrow$	(davorstehende Aussage) ist äquivalent zu (danach stehender Aussage)
A	vom Angreifer kontrollierte Station
B	Bank
b	Tiefe des Referenzenbaums bei GMR Blocklänge einer Blockchiffre verfügbare Bandbreite zur Signalisierung (in §5.5.2 wird b = 12 kbit/s angenommen)
—p—	Betrag von p Länge von p in Bits Anzahl der Elemente der Menge p
c	Chiffrierschlüssel eines asymmetrischen Konzelationssystems
CAN	Controller Area Network
$C_i$	28-Bit-Block (des Schlüssels von DES) ( $i = 1, \dots, 16$ )
code	Codieralgorithmus
const	Konstante
CRA	Chinesischer Restealgorithmus
d	Dechiffrierschlüssel eines asymmetrischen Konzelationssystems
$D_i$	28-Bit-Block (des Schlüssels von DES) ( $i = 1, \dots, 16$ )
$\delta$	Abstand vom rechten Rand der Rückkopplungseinheit
$\epsilon$	Element von
$\epsilon_R$	gleichverteilt zufällig (Random) gewählt aus, d.h. $x \in_R M$ bedeutet, daß das Element $x$ zufällig aus der Menge $M$ gewählt wird
ent	Entschlüsselungsalgorithmus
$\varepsilon$	Fehlerwahrscheinlichkeit
E	Expansionspermutation (von DES)
$E$	Ereignis
	Empfänger
$\exists$	es existiert ein
$\exp(\bullet)$	Exponentialfunktion, d.h. $\exp(x) = e^x \approx 2,718^x$
$f$	Faktor
$f(\bullet)$	Funktion

$\mathcal{F}$	Verschlüsselungsfunktion von DES Faktorisierungsalgorithmus bzw. probabilistischer polynomieller Algorithmus
$\forall$	für alle
$G$	Gruppe gerufener Partner
	Geheimnis
	Größe
	Großrechner
$g$	Generator einer zyklischen Gruppe
$gen$	Schlüsselgenerierungsalgorithmus
$ggT$	größter gemeinsamer Teiler
$G^*$	ungerichteter Graph
$h$	Hashfunktion
H	Head
$H$	Untergruppe
IP	Eingangspermutation (Initial Permutation) von DES
$IP^{-1}$	Ausgangspermutation von DES
IT-System	informationstechnisches System
$\times$	Kreuzprodukt von Mengen
$\mathcal{K}$	Schlüsselmenge
K	Kollisionsfinde-Algorithmus
$K_i$	Klartextblock
$k$	Teilschlüssel von DES ( $i = 1, \dots, 16$ )
$kgV$	geheimer Schlüssel
km	kleinste gemeinsame Vielfache
L	Kilometer
LAN	linke Hälfte eines DES-Blocks
$\mathcal{L}$	Local Area Network
$l$	Diskrete-Logarithmen-Ziehalgorithmus bzw. probabilistischer polynomieller Algorithmus
$\ell$	Länge eines Schlüssels
ld	Sicherheitsparameter
ln	Logarithmus dualis, Logarithmus zur Basis 2
$LS_i$	natürlicher Logarithmus
m	zyklischer Linksshift (in Iterationsrunde $i$ bei DES) ( $i = 1, \dots, 16$ )
$m$	Meter
MAC	Nachricht (message)
$MAC$	message authentication code (als Bezeichner eines Nachrichtenfeldes)
N	message authentication code (als Bezeichner des Wertes eines Nachrichtenfeldes)
	Nachricht

$N$	Nennwert einer Zahlung
$\mathbb{N}$	Menge der natürlichen Zahlen = $\{1, 2, 3, 4, \dots\}$
$\mathbb{N}_0$	Menge der natürlichen Zahlen einschließlich 0 = $\{0, 1, 2, 3, 4, \dots\}$
$\setminus$	ohne, d.h. Mengensubtraktion
$O(\bullet)$	Order-of-magnitude: eine Funktion $f(n)$ heißt größtenordnungsmäßig höchstens so schnell wachsend wie eine Funktion $g(n)$ , gesprochen "f ist ein groß O von g", notiert $f(n) = O(g(n))$ , wenn es Konstanten $const$ und $n_0$ gibt, so daß für alle $n \geq n_0$ gilt: $ f(n)  \leq const \cdot  g(n) $
o.B.d.A.	ohne Beschränkung der Allgemeinheit
$p$	Primzahl
$p$	Pseudonym
$P$	Prädiktor
$P$	Permutation (von DES)
$\S$	Abschnitt, Kapitel
PC	Personal Computer
PC-1	Permuted Choice 1 (bei DES)
PC-2	Permuted Choice 2 (bei DES)
PGP	Pretty Good Privacy
$\phi(n)$	Eulersche $\phi$ -Funktion: Speziell für $p, q$ prim und $p \neq q$ gilt: $\phi(p \cdot q) = (p - 1)(q - 1)$
$\pi(x)$	Anzahl der Primzahlen $\leq x$
$präf(\bullet)$	präfixfreie Abbildung
$\mathcal{Q}$	Polynom
$q$	Primzahl
$r$	Zufallszahl (verwendet als Blendungsfaktor) (random number)
$\mathcal{R}$	Rate-Algorithmus
$R$	Schlüsselregister
	Referenz einer GMR-Signatur
	rufender Partner
	Rolle
R	Rolle in einer Transaktion
	rechte Hälfte eines DES-Blocks
$\mathbb{R}$	Menge der reellen Zahlen
$\mathcal{S}$	Menge der Schlüsseltexte
$S$	Schlüsseltext
	Schlüsseltextblock
	Schlüsseltextzeichen
	Sender
	Signatur
S	steganographischer Algorithmus "Verbergen"

## 11 Glossary

$S^{-1}$	Umkehroperation zu S, d.h. ein Algorithmus, der die einge-betteten Daten extrahiert
$S_i$	Substitutionsbox (von DES) ( $i = 1, \dots, 8$ )
s	Sekunde
$s$	Signierschlüssel eines digitalen Signatursystems
	Startwert eines Pseudozufallsbitfolgengenerators
$Sig$	Signatur
$sign$	Signieralgorithmus
$a b$	a teilt b (ohne Rest), d.h. $\exists c \in \mathbb{Z} : b = ac$
$t$	Testschlüssel eines digitalen Signatursystems
T	Tail
$T$	Teilnehmerstation
$test$	Testalgorithmus
$\approx$	ungefähr gleich
$ver$	Verschlüsselungsalgorithmus
$\mathcal{W}$	Wurzelziehalgorithmus
$W$	Wahrscheinlichkeitsverteilung
$W(x)$	Wahrscheinlichkeit von x
$W(x s)$	bedingte Wahrscheinlichkeit von x, wenn man s kennt
w	Wurzel
$\mathcal{X}$	Menge der Klartexte
X	Person
x	Klartext
	Klartextzeichen
XOR	bitweises XOR = bitweise Addition mod 2
$z$	Zufallszahl
	zufällige Bitkette
Z	Schlüsselverteilzentrale
	Zufallsvariable
$\mathbb{Z}$	Menge der ganzen Zahlen = $\{\dots, -4, -3, -2, -1, 0, 1, 2, 3, 4, \dots\}$
$\mathbb{Z}_n$	Restklassen(ring) mod n
$\mathbb{Z}_n^*$	multiplikative Gruppe der zu n teilerfremden Restklassen mod n
$[x]$	kleinste ganze Zahl z mit $z \geq x$
$[y]$	größte ganze Zahl z mit $z \leq y$

hom

# Bibliography

- [AABB\_97] Hal Abelson, Ross Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Whitfield Diffie, John Gilmore, Peter G. Neumann, Ronald L. Rivest, Jeffery I. Schiller, Bruce Schneier: The Risk of Key Recovery, Key Escrow, and Trusted Third-Party Encryption. The World Wide Web Journal 2/3 (1997) 241-257.
- [AABB\_98] Hal Abelson, Ross Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Whitfield Diffie, John Gilmore, Peter G. Neumann, Ronald L. Rivest, Jeffery I. Schiller, Bruce Schneier: The Risks of Key Recovery, Key Escrow, and Trusted Third Party Encryption. Update June 1998; <http://www.cdt.org/crypto/risks98>.
- [ACGS\_84] Werner Alexi, Benny Chor, Oded Goldreich, Claus P. Schnorr: RSA/Rabin Bits are  $0.5 + 1/\text{poly}(\log N)$  Secure. 25th Symposium on Foundations of Computer Science (FOCS) 1984, IEEE Computer Society, 1984, 449-457.
- [ACGS\_88] Werner Alexi, Benny Chor, Oded Goldreich, Claus P. Schnorr: RSA and Rabin functions: Certain parts are as hard as the whole. SIAM Journal on Computing 17/2 (1988) 194-209.
- [AJSW\_97] N. Asokan, Phillip A. Janson, Michael Steiner, Michael Waidner: The State of the Art in Electronic Payment Systems. Computer 30/9 (1997) 28-35.
- [AbJe\_87] Marshall D. Abrams, Albert B. Jeng: Network Security: Protocol Reference Model and the Trusted Computer System Evaluation Criteria. IEEE Network 1/2 (1987) 24-33.
- [Adam2\_92] John A. Adam: Threats and countermeasures. IEEE spectrum 29/8 (1992) 21-28.
- [AlSc\_83] Bowen Alpern, Fred B. Schneider: Key exchange Using 'Keyless Cryptography'. Information Processing Letters 16 (1983) 79-81.
- [Alba\_83] A. Albanese: Star Network With Collision-Avoidance Circuits. The Bell System Technical Journal 62/3 (1983) 631-638.
- [Alke\_88] Horst Alke: DATENSCHUTZBEHÖRDEN: Alles registriert - nur beim Autotelefon? Registrierung durch die DBP beim "normalen

## Bibliography

- Telefon"; Vollspeicherung beim Autotelefondienst. Datenschutz und Datensicherung DuD 12/1 (1988) 4-5.
- [Amst\_83] Stanford R. Amstutz: Burst Switching – An Introduction. IEEE Communications Magazine 21/8 (1983) 36-42.
- [AnKu\_96] Ross Anderson, Markus Kuhn: Tamper Resistance - a Cautionary Note. 2nd USENIX Workshop on Electronic Commerce, 1996, 1-12.
- [AnLe\_81] Tom Anderson, Pete A. Lee: Fault Tolerance - Principles and Practice. Prentice Hall, Englewood Cliffs, New Jersey, 1981.
- [AnPe\_98] Ross J. Anderson, Fabien A. P. Petitcolas: On the Limits of Steganography. IEEE Journal on Selected Areas in Communications 16/4 (1998) 474-481.
- [Ande4\_96] Ross Anderson: Stretching the Limits of Steganography. Information Hiding, LNCS 1174, Springer-Verlag, Berlin 1996, 39-48.
- [AsSW\_97] N. Asokan, Matthias Schunter, Michael Waidner: Optimistic Protocols for Fair Exchange. 4th ACM Conference on Computer and Communications Security, Zürich, April 1997, 6-17.
- [BCKK\_83] Werner Bux, Felix H. Closs, Karl Kuemmerle, Heinz J. Keller, Hans R. Mueller: Architecture and Design of a Reliable Token-Ring Network. IEEE Journal on Selected Areas in Communications 1/5 (1983) 756-765.
- [BDFK\_95] Andreas Bertsch, Herbert Damker, Hannes Federrath, Dogan Kesdogan, Michael J. Schneider: Erreichbarkeitsmanagement. PIK, Praxis der Informationsverarbeitung und Kommunikation 18/3 (1995) 231-234.
- [BFD1\_95] Bundesbeauftragter für den Datenschutz (Hrsg.): BFD-Info 1: Bundesdatenschutzgesetz – Text und Erläuterung. 3. Auflage, PF200112, D-53131 Bonn, can be ordered there for free.
- [BFD5\_98] Bundesbeauftragter für den Datenschutz (Hrsg.): BFD-Info 5: Datenschutz und Telekommunikation. 2. Auflage, PF200112, D-53131 Bonn, can be ordered there for free.
- [BGJK\_81] P. Berger, G. Grugelke, G. Jensen, J. Kratzsch, R. Kreibich, H. Pichlmayer, J. P. Spohn: Datenschutz bei rechnerunterstützten Telekommunikationssystemen. Bundesministerium für Forschung und Technologie Forschungsbericht DV 81-006 (BMFT-FB-DV 81-006), Institut für Zukunftsforschung GmbH Berlin, September 1981.

## Bibliography

- [BGMR\_85] Michael Ben-Or, Oded Goldreich, Silvio Micali, Ronald L. Rivest: A Fair Protocol for Signing Contracts. 12th International Colloquium on Automata, Languages and Programming (ICALP), LNCS 194, Springer-Verlag, Berlin 1985, 43-52.
- [BMI\_89] Bundesministerium des Inneren: Rahmenkonzept zur Gewährleistung der Sicherheit bei Anwendung der Informationstechnik. Datenschutz und Datensicherung DuD /6 (1989) 292-299.
- [BMTW\_84] Toby Berger, Nader Mehravari, Don Towsley, Jack Wolf: Random Multiple-Access Communication and Group Testing. IEEE Transactions on Communications 32/7 (1984) 769-779.
- [Bött\_89] Manfred Böttger: Untersuchung der Sicherheit von asymmetrischen Kryptosystemen und MIX-Implementierungen gegen aktive Angriffe. Studienarbeit am Institut für Rechnerentwurf und Fehlertoleranz der Universität Karlsruhe 1989.
- [BüPf\_87] Holger Bürk, Andreas Pfitzmann: Value Transfer Systems Enabling Security and Unobservability. Interner Bericht 2/87, Fakultät für Informatik, Universität Karlsruhe 1987.
- [BüPf\_89] Holger Bürk, Andreas Pfitzmann: Digital Payment Systems Enabling Security and Unobservability. Computers & Security 8/5 (1989) 399-416.
- [BüPf\_90] Holger Bürk, Andreas Pfitzmann: Value Exchange Systems Enabling Security and Unobservability. Computers & Security 9/8 (1990) 715-721.
- [Bürk\_86] Holger Bürk: Digitale Zahlungssysteme und betrugssicherer, anonymer Wertetransfer. Studienarbeit am Institut für Informatik IV, Universität Karlsruhe, April 1986.
- [BaGo\_84] Friedrich L. Bauer, Gerhard Goos: Informatik, Eine einführende Übersicht. Zweiter Teil; Dritte Auflage völlig neu bearbeitet und erweitert von F. L. Bauer; Heidelberger Taschenbücher Band 91; Springer-Verlag, Berlin 1984.
- [BaHS\_92] Dave Bayer, Stuart Haber, W. Scott Stornetta: Improving the Efficiency and Reliability of Digital Time-Stamping. Sequences '91: Methods in Communication, Security, and Computer Sciences, Springer-Verlag, Berlin 1992, 329-334.
- [Bara\_64] Paul Baran: On Distributed Communications: IX. Security, Secrecy, and Tamper-Free Considerations. Memorandum RM-3765-PR, August 1964, The Rand Corporation, 1700 Main St, Santa Monica, California, 90406 Reprinted in: Lance J. Hoffman (ed.): Security and

## Bibliography

- Privacy in Computer Systems; Melville Publishing Company, Los Angeles, California, 1973, 99-123.
- [Baum\_99] Birgit Baum-Waidner: Haftungsbeschränkung der digitalen Signatur durch einen Commitment Service. Sicherheitsinfrastrukturen, DuD Fachbeiträge, Vieweg 1999, 65-80.
- [BeBE\_92] Charles H. Bennett, Gilles Brassard, Artur K. Ekert: Quantum Cryptography. Scientific American 267/4 (1992) 26-33.
- [BeBR\_88] Charles H. Bennett, Gilles Brassard, Jean-Marc Robert: Privacy amplification by public discussion. SIAM Journal on Computing 17/2 (1988) 210-229.
- [BeEG\_86] F. Belli, K. Echtle, W. Görke: Methoden und Modelle der Fehlertoleranz. Informatik Spektrum 9/2 (1986) 68-81.
- [BeGW\_88] Michael Ben-Or, Shafi Goldwasser, Avi Wigderson: Completeness theorems for non-cryptographic fault-tolerant distributed computation. 20th Symposium on Theory of Computing (STOC) 1988, ACM, New York 1988, 1-10.
- [BeRo\_95] Mihir Bellare, Phillip Rogaway: Optimal Asymmetric Encryption. Eu-rocrypt '94, LNCS 950, Springer-Verlag, Berlin 1995, 92-111.
- [BiSh3\_91] Eli Biham, Adi Shamir: Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology 4/1 (1991) 3-72.
- [BiSh4\_91] Eli Biham, Adi Shamir: Differential Cryptanalysis of the Full 16-round DES. Technical Report Nr. 708, December 1991, Computer Science Department, Technion, Haifa, Israel.
- [BiSh\_93] Eli Biham, Adi Shamir: Differential Cryptanalysis of the Data Encryption Standard. Springer-Verlag, New York 1993.
- [BlBS\_86] Lenore Blum, Manuel Blum, Michael Shub: A Simple Unpredictable Pseudo-Random Number Generator. SIAM Journal on Computing 15/2 (1986) 364-383.
- [BlFM\_88] Manuel Blum, Paul Feldman, Silvio Micali: Non-interactive zero-knowledge and its applications. 20th Symposium on Theory of Computing (STOC) 1988, ACM, New York 1988, 103-112.
- [BlMi\_84] Manuel Blum, Silvio Micali: How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. SIAM Journal on Computing 13/4 (1984) 850-864.

## Bibliography

- [BIPW\_91] Gerrit Bleumer, Birgit Pfitzmann, Michael Waidner: A remark on a signature scheme where forgery can be proved. Eurocrypt '90, LNCS 473, Springer-Verlag, Berlin 1991, 441-445.
- [Bleu\_90] Gerrit Bleumer: Vertrauenswürdige Schlüssel für ein Signatursystem, dessen Brechen beweisbar ist. Studienarbeit am Institut für Rechnerentwurf und Fehlertoleranz der Universität Karlsruhe 1990.
- [Bleu\_99] Gerrit Bleumer: Biometrische Ausweise – Schutz von Personenidentitäten trotz biometrischer Erkennung. Datenschutz und Datensicherheit DuD 23/3 (1999) 155-158.
- [BrDL\_91] Jørgen Brandt, Ivan Damgård, Peter Landrock: Speeding up Prime Number Generation. Asiacrypt '91, Abstracts, Fujiyoshida, November 11-14, 1992, 265-269.
- [Bras\_83] Gilles Brassard: Relativized Cryptography. IEEE Transactions on Information Theory 29/6, 877-894.
- [Bras\_88] Gilles Brassard: Modern Cryptology — A Tutorial. LNCS 325, Springer-Verlag, Berlin 1988.
- [Brau\_82] Ewald Braun: BIGFON - der Start für die Kommunikationstechnik der Zukunft. SIEMENS telcom report 5/2 (1982) 123-129.
- [Brau\_83] Ewald Braun: BIGFON - Erprobung der optischen Breitbandübertragung im Ortsnetz. SIEMENS telcom report 6/2 (1983) 52-53.
- [Brau\_84] Ewald Braun: Systemversuch BIGFON gestartet. SIEMENS telcom report 7/1 (1984) 9-11.
- [Brau\_87] E. Braun: BIGFON. Informatik-Spektrum 10/4 (1987) 216-217.
- [Bura\_88] Axel Burandt: Informationstheoretisch unverkettbare Beglaubigung von Pseudonymen mit beliebigen Signatursystemen. Studienarbeit am Institut für Rechnerentwurf und Fehlertoleranz der Universität Karlsruhe, Mai 1988.
- [CBHM\_90] David Chaum, Bert den Boer, Eugène van Heyst, Stig Mjølsnes, Adri Steenbeek: Efficient offline electronic checks. Eurocrypt '89, LNCS 434, Springer-Verlag, Berlin 1990, 294-301.
- [CTCPEC\_92] Canadian System Security Centre; Communications Security Establishment; Government of Canada: The Canadian Trusted Computer Product Evaluation Criteria. April 1992, Version 3.0e.
- [CZ\_98] Internet-Rechner knacken DES-Code; Alte Entschlüsselungszeit ist halbiert. Computer Zeitung Nr. 10, 5. März 1998, 7.

## Bibliography

- [ChAn\_90] David Chaum, Hans van Antwerpen: Undeniable signatures. *Crypto '89*, LNCS 435, Springer-Verlag, Berlin 1990, 212-216.
- [ChCD1\_88] David Chaum, Claude Crépeau, Ivan Damgård: Multiparty unconditional secure protocols. *20th Symposium on Theory of Computing (STOC) 1988*, ACM, New York 1988, 11-19.
- [ChDG\_88] David Chaum, Ivan B. Damgård, Jeroen van de Graaf: Multiparty Computations ensuring privacy of each party's input and correctness of the result. *Crypto '87*, LNCS 293, Springer-Verlag, Berlin 1988, 87-119.
- [ChEv\_87] David Chaum, Jan-Hendrik Evertse: A secure and privacy-protecting protocol for transmitting personal information between organizations. *Crypto '86*, LNCS 263, Springer-Verlag, Berlin 1987, 118-167.
- [ChFN\_90] David Chaum, Amos Fiat, Moni Naor: Untraceable Electronic Cash. *Crypto '88*, LNCS 403, Springer-Verlag, Berlin 1990, 319-327.
- [ChRo\_91] David Chaum, Sandra Roijakkers: Unconditionally Secure Digital Signatures. *Crypto '90*, LNCS 537, Springer-Verlag, Berlin 1991, 206-214.
- [Cha1\_83] David Chaum: Blind Signature System. *Crypto '83*, Plenum Press, New York 1984, 153.
- [Cha1\_84] David Chaum: A New Paradigm for Individuals in the Information Age. *1984 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, Washington 1984, 99-103.
- [Cha1\_88] D. Chaum: Blinding for unanticipated signatures. *Eurocrypt '87*, LNCS 304, Springer-Verlag, Berlin 1988, 227-233.
- [Cha1\_89] David Chaum: Online Cash Checks. *Eurocrypt '89*; Houthalen, 10.-13. April 1989, A Workshop on the Theory and Application of Cryptographic Techniques, Abstracts, 167-170.
- [Cha3\_85] David Chaum: The Dining Cryptographers Problem. Unconditional Sender Anonymity. Draft, received May 13, 1985.
- [Cha8\_85] David Chaum: Security without Identification: Transaction Systems to make Big Brother Obsolete. *Communications of the ACM* 28/10 (1985) 1030-1044.
- [Cha9\_85] David Chaum: Cryptographic Identification, Financial Transaction, and Credential Device. United States Patent, Patent Number 4,529,870; Date of Patent: Jul. 16, 1985, Filed Jun. 25, 1982.

## Bibliography

- [Chau2\_90] David Chaum: Showing credentials without identification: Transferring signatures between unconditionally unlinkable pseudonyms. Auscrypt '90, LNCS 453, Springer-Verlag, Berlin 1990, 246-264.
- [Chau\_81] David Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM 24/2 (1981) 84-88.
- [Chau\_83] David Chaum: Blind Signatures for untraceable payments. Crypto '82, Plenum Press, New York 1983, 199-203.
- [Chau\_84] David Chaum: Design Concepts for Tamper Responding Systems. Crypto '83, Plenum Press, New York 1984, 387-392.
- [Chau\_85] David Chaum: Showing credentials without identification. Signatures transferred between unconditionally unlinkable pseudonyms. Abstracts of Eurocrypt '85, April 9-11, 1985, Linz, Austria, IACR, General Chairman: Franz Pichler, Program Chairman: Thomas Beth.
- [Chau\_87] David Chaum: Sicherheit ohne Identifizierung; Scheckkartencomputer, die den Großen Bruder der Vergangenheit angehören lassen. Informatik-Spektrum 10/5 (1987) 262-277; Datenschutz und Datensicherung DuD 12/1 (1988) 26-41.
- [Chau\_88] David Chaum: The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. Journal of Cryptology 1/1 (1988) 65-75.
- [Chau\_89] David Chaum: Privacy Protected Payments - Unconditional Payer and/or Payee Untraceability. SMART CARD 2000: The Future of IC Cards, Proceedings of the IFIP WG 11.6 International Conference; Laxenburg (Austria), 19.-20. 10. 1987, North-Holland, Amsterdam 1989, 69-93.
- [Chau\_90] David Chaum: The Spymasters double-agent problem: Multiparty computations secure unconditionally from minorities and cryptographically from majorities. Crypto '89, LNCS 435, Springer-Verlag, Berlin 1990, 591-602.
- [Chau\_92] David Chaum: Achieving Electronic Privacy. Scientific American (August 1992) 96-101.
- [Chau\_95] David Chaum: Designated Confirmer Signatures. Eurocrypt '94, LNCS 950, Springer-Verlag, Berlin 1995, 86-91.
- [ClPf\_91] Wolfgang Clesle, Andreas Pfitzmann: Rechnerkonzept mit digital signierten Schnittstellenprotokollen erlaubt individuelle Verantwortungszuweisung. Datenschutz-Berater 14/8-9 (1991) 8-38.

## Bibliography

- [Clar\_88] A. J. Clark: Physical protection of cryptographic devices. Eurocrypt '87, LNCS 304, Springer-Verlag, Berlin 1988, 83-93.
- [Clem\_85] Rudolf Clemens: Die elektronische Willenserklärung - Chancen und Gefahren. Neue Juristische Wochenschrift NJW /324 (1985) 1998-2005.
- [Cles\_88] Wolfgang Clesle: Schutz auch vor Herstellern und Betreibern von Informationssystemen. Diplomarbeit am Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe, Juni 1988.
- [CoBi1\_95] David A. Cooper, Kenneth P. Birman: The design and implementation of a private message service for mobile computers. Wireless Networks 1 (1995) 297-309.
- [CoBi\_95] David A. Cooper, Kenneth P. Birman: Preserving Privacy in a Network of Mobile Computers. 1995 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos 1995, 26-38.
- [Cohe1\_89] Fred Cohen: Computational Aspects of Computer Viruses. Computers & Security 8/3 (1989) 325-344.
- [Cohe2\_92] F. B. Cohen: A Formal Definition of Computer Worms and Some Related Results. Computers & Security 11/7 (1992) 641-652.
- [Cohe\_84] Fred Cohen: Computer Viruses, Theory and Experiments. Proceedings of the 7th National Computer Security Conference, 1984, National Bureau of Standards, Gaithersburg, MD; USA, 240-263.
- [Cohe\_87] Fred Cohen: Computer Viruses - Theory and Experiments. Computers & Security 6/1 (1987) 22-35.
- [Copp2\_94] D. Coppersmith: The Data Encryption Standard (DES) and its strength against attacks. IBM Journal of Research and Development 38/3 (1994) 243-250.
- [Copp\_92] Don Coppersmith: DES and differential cryptanalysis. appeared in an IBM internal newsgroup (CRYPTAN FORUM).
- [Cove\_90] Minutes of the First Workshop on Covert Channel Analysis: Overview, Introduction and Welcomes, Retrospective, Worked Examples, Covert Channel Examples, Research Panel, Vendor Panel, Policy Panel, General Discussion, Research Working Group Discussion, Results of the Research Working Group, Policy Working Group Discussion, Results of the Policy Working Group, Summary Discussion. CIPHER Newsletter of the TC on Security & Privacy, IEEE Computer Society (Special Issue, July 1990) 1-35.

## Bibliography

- [CrSh\_98] Ronald Cramer, Victor Shoup: A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. Manuscript, Swiss Federal Institute of Technology (ETH), Zurich 1998.
- [DES\_77] Specification for the Data Encryption Standard. Federal Information Processing Standards Publication 46 (FIPS PUB 46), January 15, 1977.
- [DINISO8372\_87] DIN ISO 8372: Informationsverarbeitung - Betriebsarten für einen 64-bit-Blockschlüsselungsalgorithmus.
- [DaIF\_94] Don Davis, Ross Ihaka, Philip Fenstermacher: Cryptographic randomness from air turbulence in disk drives. Crypto '94, LNCS 839, Springer-Verlag, Berlin 1994, 114-120.
- [DaPa\_83] D. W. Davies, G. I. P. Parkin: The Average Cycle Size of the Key Stream in Output Feedback Encipherment. Cryptography, Proceedings Burg Feuerstein 1982, LNCS 149, Springer-Verlag, Berlin 1983, 263-279.
- [DaPr\_84] D. W. Davies, W. L. Price: Security for Computer Networks, An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer. John Wiley & Sons, New York 1984.
- [DaPr\_89] Donald W. Davies, Wyn L. Price: Security for Computer Networks, An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer. (2nd ed.) John Wiley & Sons, New York 1989.
- [Damg\_88] Ivan Bjerre Damgård: Collision free hash functions and public key signature schemes. Eurocrypt '87, LNCS 304, Springer-Verlag, Berlin 1988, 203-216.
- [Damg\_92] Ivan Damgård: Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks. Crypto '91, LNCS 576, Springer-Verlag, Berlin 1992, 445-456.
- [Davi\_85] Danald Watts Davies: Apparatus and methods for granting access to computers. UK Patent Application, 2 154 344 A, Application published 4.9.1985.
- [Denn\_82] Dorothy Denning: Cryptography and Data Security. Addison-Wesley, Reading 1982; reprinted with corrections, January 1983.
- [Denn\_84] Dorothy E. Denning: Digital Signatures with RSA and Other Public-Key Cryptosystems. Communications of the ACM 27/4 (1984) 388-392.

## Bibliography

- [Denn\_85] Dorothy E. Denning: Commutative Filters for Reducing Inference Threats in Multilevel Database Systems. Proceedings of the 1985 Symposium on Security and Privacy, April 22-24, 1985, Oakland, California, IEEE Computer Society, 134-146.
- [DiHe\_76] Whitfield Diffie, Martin E. Hellman: New Directions in Cryptography. IEEE Transactions on Information Theory 22/6 (1976) 644-654.
- [DiHe\_79] Whitfield Diffie, Martin E. Hellman: Privacy and Authentication: An Introduction to Cryptography. Proceedings of the IEEE 67/3 (1979) 397-427.
- [Dier3\_91] Rüdiger Dierstein: Programm-Manipulationen: Arten, Eigenschaften und Bekämpfung. Deutsche Forschungsanstalt für Luft- und Raumfahrt (DLR), Zentrale Datenverarbeitung, D-8031 Oberpfaffenhofen; Institutsbericht IB 562/6, Juli 1986, Neufassung 1991.
- [Dobb\_98] Hans Dobbertin: Cryptanalysis of MD4. Journal of Cryptology 11/4 (1998) 253-271.
- [DuD3\_99] DuD Report: 57. Konferenz des DSB des Bundes und der Länder vom 25./26. März 1999: Entschlüsseungen. Datenschutz und Datensicherheit DuD 23/5 (1999) 301-305.
- [DuDR2\_92] DuD Report: Dark Avenger Mutation Engine. Datenschutz und Datensicherung DuD 16/8 (1992) 442-443.
- [DuD\_86] DuD: AKTUELL. Datenschutz und Datensicherung DuD 10/1 (1986) 54-56.
- [EFF\_98] Electronic Frontier Foundation (EFF): "EFF DES cracker" machine brings honesty to crypto debate - Electronic Frontier Foundation proves that DES is not secure. Press Release, San Francisco, July 17, 1998.
- [EMMT\_78] W. F. Ehrsam, S. M. Matyas, C. H. Meyer, W. L. Tuchman: A cryptographic key management scheme for implementing the Data Encryption Standard. IBM Systems Journal 17/2 (1978) 106-125.
- [Eber\_98] Ulrich Eberl: Der unverlierbare Schlüssel. Forschung und Innovation — Die SIEMENS-Zeitschrift für Wissenschaft und Technik /1 (1998) 14-19.
- [EcGM\_83] Klaus Echtle, Winfried Görke, Michael Marhöfer: Zur Begriffsbildung bei der Beschreibung von Fehlertoleranz-Verfahren. Universität Karlsruhe, Fakultät für Informatik, Institut für Informatik IV, Interner Bericht Nr. 6/83 Mai 1983.

## Bibliography

- [Echt\_90] Klaus Echtle: Fehlertoleranzverfahren. Studienreihe Informatik, Springer-Verlag, Berlin 1990.
- [ElGa\_85] Taher ElGamal: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information Theory 31/4 (1985) 469-472.
- [EnHa\_96] Erin English, Scott Hamilton: Network Security Under Siege; The Timing Attack. Computer 29/3 (1996) 95-97.
- [EvGL\_85] Shimon Even, Oded Goldreich, Abraham Lempel: A Randomized Protocol for Signing Contracts. Communications of the ACM 28/6 (1985) 637-647.
- [EvGY\_84] S. Even, O. Goldreich, Y. Yacobi: Electronic Wallet. 1984 International Zurich Seminar on Digital Communications, Applications of Source Coding, Channel Coding and Secrecy Coding, March 6-8, 1984, Zurich, Switzerland, Swiss Federal Institute of Technology, Proceedings IEEE Catalog no. 84CH1998-4, 199-201.
- [Even\_89] Shimon Even: Secure Off-Line Electronic Fund Transfer Between Non-trusting Parties. SMART CARD 2000: The Future of IC Cards, Proceedings of the IFIP WG 11.6 International Conference; Laxenburg (Austria), 19.-20. 10. 1987, North-Holland, Amsterdam 1989, 57-66.
- [FGJP\_98] Elke Franz, Andreas Graubner, Anja Jerichow, Andreas Pfitzmann: Comparison of Commitment Schemes Used in Mix-Mediated Anonymous Communication for Preventing Pool-Mode Attacks. 3rd Australasian Conference on Information Security and Privacy (ACISP '98), LNCS 1438, Springer-Verlag, Berlin 1998, 111-122.
- [FJKPS\_96] Hannes Federrath, Anja Jerichow, Dogan Kesdogan, Andreas Pfitzmann, Otto Spaniol: Mobilkommunikation ohne Bewegungsprofile. it+ti 38/4, 24-29, Republished in: G. Müller, A. Pfitzmann (Hrsg.), Mehrseitige Sicherheit in der Kommunikationstechnik, Addison-Wesley-Longman 1997, 169-180.
- [FJKP\_95] Hannes Federrath, Anja Jerichow, Dogan Kesdogan, Andreas Pfitzmann: Security in Public Mobile Communication Networks. Proc. of the IFIP TC 6 International Workshop on Personal Wireless Communications, Verlag der Augustinus Buchhandlung Aachen, 1995, 105-116.
- [FJMP\_96] Elke Franz, Anja Jerichow, Steffen Möller, Andreas Pfitzmann, Ingo Stierand: Computer Based Steganography: How it works and why therefore any restrictions on cryptography are nonsense, at best. Information Hiding, First International Workshop, Cambridge, UK, May/June 1996, LNCS 1174, Springer Verlag, Heidelberg 1996, 7-21.

## Bibliography

- [FJMP\_97] Hannes Federrath, Anja Jerichow, Jan Müller, Andreas Pfitzmann: Unbeobachtbarkeit in Kommunikationsnetzen. Verlässliche IT-Systeme, GI-Fachtagung VIS '97, DuD Fachbeiträge, Vieweg 1997, 191-210.
- [FJPP\_95] Hannes Federrath, Anja Jerichow, Andreas Pfitzmann, Birgit Pfitzmann: Mehrseitig sichere Schlüsselerzeugung. Proc. Arbeitskonferenz Trust Center 95, DuD Fachbeiträge, Vieweg 1995, 117-131.
- [FMSS\_96] S. M. Furnell, J. P. Morrissey, P. W. Sanders, C. T. Stockel: Applications of keystroke analysis for improved login security and continuous user authentication. 12th IFIP International Conference on Information Security (IFIP/Sec '96), Chapman & Hall, London 1996, 283-294.
- [FO\_87] FO: Ein Fall für Prometheus. ADAC motorwelt /4 (1987) 50-53.
- [FaLa\_75] David J. Farber, Kenneth C. Larson: Network Security Via Dynamic Process Renaming. Fourth Data Communications Symposium, 7-9 October 1975, Quebec City, Canada, 8-13-8-18.
- [FeJP\_96] Hannes Federrath, Anja Jerichow, Andreas Pfitzmann: Mixes in mobile communication systems: Location management with privacy. Information Hiding, LNCS 1174, Springer-Verlag, Berlin 1996, 121-135.
- [FePf\_97] Hannes Federrath, Andreas Pfitzmann: Bausteine zur Realisierung mehrseitiger Sicherheit. in: Günter Müller, Andreas Pfitzmann (Ed.): Mehrseitige Sicherheit in der Kommunikationstechnik, Addison-Wesley-Longman 1997, 83-104.
- [FePf\_99] Hannes Federrath, Andreas Pfitzmann: Stand der Sicherheitstechnik. in: Kubicek et. al. (Hrsg.): Multimedia@Verwaltung, Jahrbuch Telekommunikation und Gesellschaft 1999, Hüting, 124-132.
- [Fede\_98] Hannes Federrath: Vertrauenswürdiges Mobilitätsmanagement in Telekommunikationsnetzen. Dissertation, TU Dresden, Fakultät Informatik, Februar 1998.
- [Fede\_99] Hannes Federrath: Sicherheit mobiler Kommunikation. DuD Fachbeiträge, Vieweg 1999, 1-263.
- [FiSh\_87] Amos Fiat, Adi Shamir: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. Crypto '86, LNCS 263, Springer-Verlag, Berlin 1987, 186-194.
- [FoPf\_91] Dirk Fox, Birgit Pfitzmann: Effiziente Software-Implementierung des GMR-Signatursystems. Proc. Verlässliche Informationssysteme (VIS'91), Informatik-Fachberichte 271, Springer-Verlag, Berlin 1991, 329-345.

## Bibliography

- [Folt\_87] Hal Folts: Open Systems Standards. IEEE Network 1/2 (1987) 45-46.
- [FrJP\_97] Elke Franz, Anja Jerichow, Andreas Pfitzmann: Systematisierung und Modellierung von Mixen. Verlässliche IT-Systeme, GI-Fachtagung VIS '97, DuD Fachbeiträge, Vieweg 1997, 171-190.
- [FrJW\_98] Elke Franz, Anja Jerichow, Guntram Wicke: A Payment Scheme for Mixes Providing Anonymity. Trends in Distributed Systems for Electronic Commerce (TREC), LNCS 1402, Springer-Verlag, Berlin 1998, 94-108.
- [FrPf\_98] Elke Franz, Andreas Pfitzmann: Einführung in die Steganographie und Ableitung eines neuen Stegoparadigmas. Informatik-Spektrum 21/4 (1998) 183-193.
- [GGKL\_89] Morrie Gasser, Andy Goldstein, Charlie Kaufman, Butler Lampson: The Digital Distributed System Security Architecture. Proc. 12th National Computer Security Conference 1989, 305-319.
- [GGPS\_97] Gunther Gattung, Rüdiger Grimm, Ulrich Pordesch, Michael J. Schneider: Persönliche Sicherheitsmanager in der virtuellen Welt. in: Günter Müller, Andreas Pfitzmann (Hrsg.): Mehrseitige Sicherheit in der Kommunikationstechnik, Addison-Wesley-Longman 1997, 181-205.
- [GKLR\_92] M. Gasser, C. Kaufman, J. Linn, Y. Le Roux, J. Tardo: DASS: Distributed Authentication Security Service. Education and Society, R. Aiken (ed.), Proc. 12th IFIP World Computer Congress 1992, Information Processing 92, Vol. II, Elsevier Science Publishers B.V. (North-Holland), 1992, 447-456.
- [GaJo\_80] Michael R. Garey, David S. Johnson: Computers and Intractability - A Guide to the Theory of NP-Completeness. 2nd Printing, W.H. Freeman and Company, New York 1980.
- [GiLB\_85] David K. Gifford, John M. Lucassen, Stephen T. Berlin: The Application of Digital Broadcast Communication to Large Scale Information Systems. IEEE Journal on Selected Areas in Communications 3/3 (1985) 457-467.
- [GiMS\_74] E. N. Gilbert, F. J. MacWilliams, N. J. A. Sloane: Codes which detect deception. The Bell System Technical Journal 53/3 (1974) 405-424.
- [GlVi\_88] Cezary Glowacz, Heinrich Theodor Vierhaus: Der Las-Vegas-Chip. Der GMD-Spiegel 18/2-3 (1988) 6-8.
- [GoGM\_86] Oded Goldreich, Shafi Goldwasser, Silvio Micali: How to construct random functions. Journal of the ACM 33/4 (1986) 792-807.

## Bibliography

- [GoKi\_86] Shafi Goldwasser, Joe Kilian: Almost All Primes Can be Quickly Certified. 18th Symposium on Theory of Computing (STOC) 1986, ACM, New York 1986, 316-329.
- [GoMR\_88] Shafi Goldwasser, Silvio Micali, Ronald L. Rivest: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM Journal on Computing 17/2 (1988) 281-308.
- [GoMT\_82] Shafi Goldwasser, Silvio Micali, Po Tong: Why and How to Establish a Private Code on a Public Network. 23rd Symposium on Foundations of Computer Science (FOCS) 1982, IEEE Computer Society, 1982, 134-144.
- [Groß\_91] Michael Groß: Vertrauenswürdiges Booten als Grundlage authentischer Basissysteme. Proc. Verlässliche Informationssysteme (VIS'91), Informatik-Fachberichte 271, Springer-Verlag, Berlin 1991, 190-207.
- [HöPf\_85] Gunter Höckel, Andreas Pfitzmann: Untersuchung der Datenschutzeigenschaften von Ringzugriffsmechanismen. 1. GI Fachtagung Datenschutz und Datensicherung im Wandel der Informationstechnologien, IFB 113, Springer-Verlag, Berlin 1985, 113-127.
- [Höck\_85] Gunter Höckel: Untersuchung der Datenschutzeigenschaften von Ringzugriffsmechanismen. Diplomarbeit am Institut für Informatik IV, Universität Karlsruhe, August 1985.
- [HaMö\_98] Rainer Hamm, Klaus Peter Möller (Hrsg.): Datenschutz durch Kryptographie - ein Sicherheitsrisiko? Nomos Verlagsgesellschaft, Baden-Baden 1998.
- [HaSt\_91] Stuart Haber, W. Scott Stornetta: How to Time-Stamp a Digital Document. Journal of Cryptology 3/2 (1991) 99-111.
- [Hast\_86] Johan Håstad: On Using RSA with low exponent in a public key network. Crypto '85, LNCS 218, Springer-Verlag, Berlin 1986, 403-408.
- [Hast\_88] Johan Håstad: Solving simultaneous modular equations of low degree. SIAM Journal on Computing 17/2 (1988) 336-362.
- [HeKW\_85] Franz-Peter Heider, Detlef Kraus, Michael Welschenbach: Mathematische Methoden der Kryptoanalyse. DuD Fachbeiträge 8, Vieweg 1985.
- [HePe\_93] Eugène van Heyst, Torben Pryds Pedersen: How to Make Efficient Fail-stop Signatures. Eurocrypt '92, LNCS 658, Springer-Verlag, Berlin 1993, 366-377.

- [Herd\_85] Siegfried Herda: Authenticity, Anonymity and Security in OSIS. An Open System for Information Services. Proceedings der 1. GI Fachtagung Datenschutz und Datensicherung im Wandel der Informationstechnologien, München, Oktober 1985, herausgegeben von P.P.Spies, IFB 113, Springer-Verlag, Berlin 1985, 35-50.
- [Hopk\_89] John Hopkinson: Information Security - "The Third Facet". Proc. 1st Annual Canadian Computer Security Conference, 30 January - 1 February, 1989, Ottawa, Canada, Hosted by The System Security Centre, Communications Security Establishment, Government of Canada, 109-129.
- [Horg\_85] John Horgan: Thwarting the information thieves. IEEE spectrum 22/7 (1985) 30-41.
- [Hors\_85] Patrick Horster: Kryptologie. Reihe Informatik/47, Herausgegeben von K. H. Böhling, U. Kulisch, H. Maurer, Bibliographisches Institut, Mannheim 1985.
- [HuPf2\_96] Michaela Huhn, Andreas Pfitzmann: Krypto(de)regulierung. DatenschutzNachrichten (DANA), Herausgeber: Deutsche Vereinigung für Datenschutz e. V., ISSN 0173-7767, 19/6 (1996) 4-13.
- [HuPf\_96] Michaela Huhn, Andreas Pfitzmann: Technische Randbedingungen jeder Kryptoregulierung. Datenschutz und Datensicherheit DuD 20/1 (1996) 23-26.
- [HuPf\_98] Michaela Huhn, Andreas Pfitzmann: Verschlüsselungstechniken für das Netz. in: Claus Leggewie, Christa Maar (Hrsg.): Internet & Politik, Bollmann-Verlag, Köln 1998, 438-456.
- [ISO7498-2\_87] ISO: Information processing systems — Open Systems Interconnection Reference Model — Part 2: Security architecture. DRAFT INTERNATIONAL STANDARD ISO/DIS 7498-2; ISO TC 97, Submitted on 1987-06-18.
- [ISO8731-1\_87] ISO: Banking — Approved algorithms for message authentication — Part 1: DEA. ISO International Standard, First edition 01.06.1987.
- [ITG\_87] ITG-Empfehlungen: ISDN-Begriffe. Nachrichtentechnische Zeitschrift ntz 40/11 (1987) 814-819.
- [ITSEC2\_91] European Communities - Commission: ITSEC: Information Technology Security Evaluation Criteria. (Provisional Harmonised Criteria, Version 1.2, 28 June 1991) Office for Official Publications of the European Communities, Luxembourg 1991 (ISBN 92-826-3004-8).

## Bibliography

- [ITSEC2d\_91] Europäische Gemeinschaften - Kommission: Kriterien für die Bewertung der Sicherheit von Systemen der Informationstechnik (ITSEC). (Vorläufige Form der harmonisierten Kriterien, Version 1.2, 28. Juni 1991) Amt für amtliche Veröffentlichungen der Europäischen Gemeinschaften, Luxemburg 1991 (ISBN 92-826-3003-X), erschien auch im Bundesanzeiger, Herausgegeben vom Bundesminister der Just.
- [JMPP\_98] Anja Jerichow, Jan Müller, Andreas Pfitzmann, Birgit Pfitzmann, Michael Waidner: Real-Time Mixes: A Bandwidth-Efficient Anonymity Protocol. IEEE Journal on Selected Areas in Communications 16/4 (1998) 495-509.
- [KFBG\_90] T. Kropf, J. Frößl, W. Beller, T. Giesler: A Hardware Implementation of a Modified DES-Algorithm. Microprocessing and Microprogramming 30 (1990) 59-65.
- [Köhle\_86] Helmut Köhler: Die Problematik automatisierter Rechtsvorgänge, insbesondere von Willenserklärungen (Teil I). Datenschutz und Datensicherung DuD 10/6 (1986) 337-344.
- [Kahn\_96] David Kahn: The History of Steganography. Information Hiding, First International Workshop, Cambridge, UK, May/June 1996, LNCS 1174, Springer-Verlag, Heidelberg 1996, 1-5.
- [Karg\_77] Paul A. Karger: Non-Discretionary Access Control for Decentralized Computing Systems. Master Thesis, Massachusetts Institute of Technology, Laboratory for Computer Science, 545 Technology Square, Cambridge, Massachusetts 02139, May 1977, Report MIT/LCS/TR-179.
- [KeMM\_83] Heinz J. Keller, Heinrich Meyr, Hans R. Müller: Transmission Design Criteria for a Synchronous Token Ring. IEEE Journal on Selected Areas in Communications 1/5 (1983) 721-733.
- [Ken1\_81] Stephen T. Kent: Security Requirements and Protocols for a Broadcast Scenario. IEEE Transactions on Communications 29/6 (1981) 778-786.
- [Kesd\_99] Dogan Kesdogan: Vertrauenswürdige Kommunikation in offenen Umgebungen. Dissertation, RWTH Aachen, Mathematisch-Naturwissenschaftliche Fakultät, 1999.
- [KlPi\_97] Herbert Klimant, Rudi Piotraschke: Informationstheoretische Bewertung steganographischer Konzelationssysteme. Verlässliche IT-Systeme, GI-Fachtagung VIS '97, DuD Fachbeiträge, Vieweg 1997, 225-232.

## Bibliography

- [Klei\_75] Leonard Kleinrock: Queueing Systems - Volume I: Theory. John Wiley and Sons, New York 1975.
- [Knut\_97] Donald E. Knuth: The Art of Computer Programming, Vol. 2: Seminumerical Algorithms (3rd ed.). Addison-Wesley, Reading 1997.
- [Koch\_98] Paul C. Kocher: On Certificate Revocation and Validation. 2nd International Conference on Financial Cryptography (FC '98), LNCS 1465, Springer-Verlag, Berlin 1998, 172-177.
- [Kran\_86] Evangelos Kranakis: Primality and Cryptography. Wiley-Teubner Series in Computer Science, B. G. Teubner, Stuttgart 1986.
- [Krat\_84] Herbert Krath: Stand und weiterer Ausbau der Breitbandverteilnetze. telematica 84, 18.-20. Juni 1984, Stuttgart, Kongreßband Teil 2 Breitbandkommunikation, Herausgeber: W. Kaiser, VDE-Verlag GmbH, Neue Mediengesellschaft Ulm mbH, 3-17.
- [Lüne\_87] Heinz Lüneburg: Kleine Fibel der Arithmetik. Lehrbücher und Monographien zur Didaktik der Mathematik, Band 8, BI Wissenschaftsverlag, Mannheim 1987.
- [LaMa\_92] X. Lai, J. L. Massey: Hash functions based on block ciphers. Eurocrypt '92, Extended Abstracts, 24.-28. Mai 1992, Balatonfüred, Ungarn, 53-66.
- [LaSP\_82] Leslie Lamport, Robert Shostak, Marshall Pease: The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems 4/3 (1982) 382-401.
- [Lamp\_73] Butler W. Lampson: A Note on the Confinement Problem. Communications of the ACM 16/10 (1973) 613-615.
- [LeMa1\_89] Arjen K. Lenstra, Mark S. Manasse: Factoring - where are we now? IACR Newsletter, A Publication of the International Association for Cryptologic Research, 6/2 (1989) 4-5.
- [LeMa1\_90] Arjen K. Lenstra, Mark S. Manasse: Factoring by electronic mail. Eurocrypt '89, LNCS 434, Springer-Verlag, Berlin 1990, 355-371.
- [Leib\_99] Otto Leiberich: Vom diplomatischen Code zur Falltürfunktion. Spektrum der Wissenschaft /3 (1999) 106–108.
- [LiFl\_83] John O. Limb, Lois E. Flamm: A Distributed Local Area Network Packet Protocol for Combined Voice and Data Transmission. IEEE Journal on Selected Areas in Communications 1/5 (1983) 926-934.

## Bibliography

- [Lips\_81] John D. Lipson: Elements of algebra and algebraic computing. Addison-Wesley Publishing Company, Advanced Book Program; Reading, Mass. 1981.
- [LuPW\_91] Jörg Lukat, Andreas Pfitzmann, Michael Waidner: Effizientere fail-stop Schlüsselerzeugung für das DC-Netz. Datenschutz und Datensicherung DuD 15/2 (1991) 71-75.
- [LuRa\_89] Michael Luby, Charles Rackoff: A Study of Password Security. Journal of Cryptology 1/3 (1989) 151-158.
- [MöPS\_94] Steffen Möller, Andreas Pfitzmann, Ingo Stierand: Rechnergestützte Steganographie: Wie sie funktioniert und warum folglich jede Reglementierung von Verschlüsselung unsinnig ist. Datenschutz und Datensicherung DuD 18/6 (1994) 318-326.
- [MüPf\_97] Günther Müller, Andreas Pfitzmann (eds.): Mehrseitige Sicherheit in der Kommunikationstechnik - Verfahren, Komponenten, Integration. Addison-Wesley-Longman Verlag GmbH, 1997, 5-579.
- [MaPf\_87] Andreas Mann, Andreas Pfitzmann: Technischer Datenschutz und Fehlertoleranz in Kommunikationssystemen. Kommunikation in Verteilten Systemen, GI/NTG-Fachtagung, Aachen 1987, IFB 130, Springer-Verlag, Berlin 1987, 16-30; Überarbeitung in: Datenschutz und Datensicherung DuD /8 (1987) 393-405.
- [Mann\_85] Andreas Mann: Fehlertoleranz und Datenschutz in Ringnetzen. Diplomarbeit am Institut für Informatik IV, Universität Karlsruhe, Oktober 1985.
- [Marc\_88] Eckhard Marchel: Leistungsbewertung von überlagerndem Empfangen bei Mehrfachzugriffsverfahren mittels Kollisionsauflösung. Diplomarbeit am Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe, April 1988.
- [MeMa\_82] Carl H. Meyer, Stephen M. Matyas: Cryptography - A New Dimension in Computer Data Security. (3rd printing) John Wiley & Sons, 1982.
- [MeOV\_97] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone: Handbook of Applied Cryptography. CRC Press, Boca Raton 1997.
- [Merr\_83] Michael John Merritt: Cryptographic Protocols. Ph. D. Dissertation, School of Information and Computer Science, Georgia Institute of Technology, February 1983.
- [Meye\_81] Meyers Lexikonredaktion: Meyers Großes Taschenlexikon in 24 Bänden (Lizenzausgabe "Farbiges Großes Volkslexikon", Mohndruck, Gütersloh). B.I.-Taschenbuchverlag, Mannheim 1981.

## Bibliography

- [Meye\_87] Meyers Lexikonredaktion: Meyers Großes Taschenlexikon in 24 Bänden. 2. neu bearbeitete Auflage, B.I.-Taschenbuchverlag, Mannheim 1987.
- [Mill3\_94] Benjamin Miller: Vital signs of identity. IEEE spectrum 31/2 (1994) 22-30.
- [Mt2] Matthäus: 2,16; The Bible.
- [NaYu\_90] Moni Naor, Moti Yung: Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. 22nd Symposium on Theory of Computing (STOC) 1990, ACM, New York 1990, 427-437.
- [NeKe\_99] Heike Neumann, Volker Kessler: Formale Analyse von kryptographischen Protokollen mit BAN-Logik. Datenschutz und Datensicherheit DuD 23/2 (1999) 90-93.
- [Nied\_87] Arnold Niedermaier: Bewertung von Zuverlässigkeit und Senderanonimität einer fehlertoleranten Kommunikationsstruktur. Diplomarbeit am Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe, September 1987.
- [OkOh1\_89] Tatsuaki Okamoto, Kazuo Ohta: Disposable Zero-knowledge Authentications and Their Applications to Untraceable Electronic Cash. Crypto '89, August 20-24 1989, Abstracts, 443-458.
- [OkOh\_89] Tatsuaki Okamoto, Kazuo Ohta: Divisible Zero Knowledge Interactive Proofs and Commutative Random Self-Reducibility. Eurocrypt '89; Houthalen, 10.-13. April 1989, A Workshop on the Theory and Application of Cryptographic Techniques, Abstracts, 95-108.
- [PPSW\_95] Andreas Pfitzmann, Birgit Pfitzmann, Matthias Schunter, Michael Waidner: Vertrauenswürdiger Entwurf portabler Benutzerendgeräte und Sicherheitsmodule. Verlässliche IT-Systeme, GI-Fachtagung VIS '95, DuD Fachbeiträge, Vieweg, Braunschweig 1995, 329-350.
- [PPSW\_97] Andreas Pfitzmann, Birgit Pfitzmann, Matthias Schunter, Michael Waidner: Trusting Mobile User Devices and Security Modules. Computer 30/2 (1997) 61-68.
- [PSWW\_96] Andreas Pfitzmann, Alexander Schill, Guntram Wicke, Gritta Wolf, Jan Zöllner: 1. Zwischenbericht SSONET. <http://mephisto.inf.tu-dresden.de/RESEARCH/ssonet/reports.html>, 31.08.96.
- [PSWW\_98] Andreas Pfitzmann, Alexander Schill, Andreas Westfeld, Guntram Wicke, Gritta Wolf, Jan Zöllner: A Java-Based Distributed Platform for Multilateral Security. Trends in Distributed Systems for Electronic Commerce (TREC), LNCS 1402, Springer-Verlag, Berlin 1998, 52-64.

## Bibliography

- [PWP\_90] Birgit Pfitzmann, Michael Waidner, Andreas Pfitzmann: Rechtssicherheit trotz Anonymität in offenen digitalen Systemen. Datenschutz und Datensicherung DuD 14/5-6 (1990) 243-253, 305-315.
- [Papa\_86] Stavros Papastavridis: Upper and Lower Bounds for the Reliability of a Consecutive-k-out-of-n:F System. IEEE Transactions on Reliability 35/5 (1986) 607-610.
- [Paul\_78] Wolfgang J. Paul: Komplexitätstheorie. Teubner Studienbücher Informatik, Band 39, B. G. Teubner Verlag, Stuttgart 1978.
- [Pede\_96] Torben P. Pedersen: Electronic Payments of Small Amounts. Security Protocols 1996, LNCS 1189, Springer-Verlag, Berlin 1997, 59-68.
- [PfAß1\_90] Andreas Pfitzmann, Ralf Aßmann: More Efficient Software Implementations of (Generalized) DES. Interner Bericht 18/90, Fakultät für Informatik, Universität Karlsruhe 1990.
- [PfAß\_93] Andreas Pfitzmann, Ralf Aßmann: More Efficient Software Implementations of (Generalized) DES. Computers & Security 12/5 (1993) 477-500.
- [PfPW1\_89] Andreas Pfitzmann, Birgit Pfitzmann, Michael Waidner: Telefon-MIXe: Schutz der Vermittlungsdaten für zwei 64-kbit/s-Duplexkanäle über den (2 64 + 16)-kbit/s-Teilnehmeranschluß. Datenschutz und Datensicherung DuD 13/12 (1989) 605-622.
- [PfPW\_89] Andreas Pfitzmann, Birgit Pfitzmann, Michael Waidner: Garantierter Datenschutz für zwei 64-kbit/s-Duplexkanäle über den (2 64 + 16)-kbit/s-Teilnehmeranschluß durch Telefon-MIXe. Tagungsband 3 der 4. SAVE-Tagung, 19.-21. April 1989, Köln, 1417-1447.
- [PfPf\_90] Birgit Pfitzmann, Andreas Pfitzmann: How to Break the Direct RSA-Implementation of MIXes. Eurocrypt '89, LNCS 434, Springer-Verlag, Berlin 1990, 373-381.
- [PfPf\_92] Andreas Pfitzmann, Birgit Pfitzmann: Technical Aspects of Data Protection in Health Care Informatics. Advances in Medical Informatics, J. Noothoven van Goor and J. P. Christensen (Eds.), IOS Press, Amsterdam 1992, 368-386.
- [PfWa2\_97] Birgit Pfitzmann, Michael Waidner: Strong Loss Tolerance of Electronic Coin Systems. ACM Transactions on Computer Systems 15/2 (1997) 194-213.
- [PfWa\_86] Andreas Pfitzmann, Michael Waidner: Networks without user observability – design options. Eurocrypt '85, LNCS 219, Springer-Verlag, Berlin 1986, 245-253.

## Bibliography

- [PfWa\_91] Birgit Pfitzmann, Michael Waidner: Fail-stop-Signaturen und ihre Anwendung. Proc. Verlässliche Informationssysteme (VIS'91), Informatik-Fachberichte 271, Springer-Verlag, Berlin 1991, 289-301.
- [Pfi1\_83] Andreas Pfitzmann: Ein dienstintegriertes digitales Vermittlungs-/Verteilnetz zur Erhöhung des Datenschutzes. Fakultät für Informatik, Universität Karlsruhe, Interner Bericht 18/83, Dezember 1983.
- [Pfi1\_85] Andreas Pfitzmann: How to implement ISDNs without user observability - Some remarks. Fakultät für Informatik, Universität Karlsruhe, Interner Bericht 14/85.
- [Pfit8\_96] Birgit Pfitzmann: Digital Signature Schemes - General Framework and Fail-Stop Signatures. LNCS 1100, Springer-Verlag, Berlin 1996.
- [Pfit\_83] Andreas Pfitzmann: Ein Vermittlungs-/Verteilnetz zur Erhöhung des Datenschutzes in Bildschirmtext-ähnlichen Neuen Medien. GI '83 13. Jahrestagung der Gesellschaft für Informatik 3. bis 7. Oktober 1983, Universität Hamburg, IFB 73, Springer-Verlag, Berlin, 411-418.
- [Pfit\_85] Andreas Pfitzmann: Technischer Datenschutz in diensteintegrierenden Digitalnetzen - Problemanalyse, Lösungsansätze und eine angepaßte Systemstruktur. Proceedings der 1. GI Fachtagung Datenschutz und Datensicherung im Wandel der Informationstechnologien, München, Oktober 1985, herausgegeben von P.P.Spies, IFB 113, Springer-Verlag, Berlin 1985, 96-112.
- [Pfit\_86] Andreas Pfitzmann: Die Infrastruktur der Informationsgesellschaft: Zwei getrennte Fernmeldenetze beibehalten oder ein wirklich datenschützendes errichten? Datenschutz und Datensicherung DuD 10/6 (1986) 353-359.
- [Pfit\_89] Andreas Pfitzmann: Diensteintegrierende Kommunikationsnetze mit teilnehmerüberprüfbarem Datenschutz. Universität Karlsruhe, Fakultät für Informatik, Dissertation, Feb. 1989, IFB 234, Springer-Verlag, Berlin 1990.
- [Pfit\_93] Birgit Pfitzmann: Sorting Out Signature Schemes - and some Theory of Secure Reactive Systems. Extended Abstract, Institut für Informatik, Universität Hildesheim, 4.1.1993; submitted for publication.
- [PoKL\_78] G. J. Popek, C. S. Kline: Design Issues for Secure Computer Networks. Operating Systems, An Advanced Course, Edited by R. Bayer, R. M. Graham, G. Seegmüller; LNCS 60, 1978; Nachgedruckt als Springer Study Edition, 1979; Springer-Verlag, Berlin, 517-546.

## Bibliography

- [PoSc\_97] Ulrich Pordesch, Michael J. Schneider: Sichere Kommunikation in der Gesundheitsversorgung. in: Günter Müller, Andreas Pfitzmann (Hrsg.): Mehrseitige Sicherheit in der Kommunikationstechnik, Addison-Wesley-Longman 1997, 61-80.
- [RDFR\_97] Martin Reichenbach, Herbert Damker, Hannes Federrath, Kai Rannenberg: Individual Management of Personal Reachability in Mobile Communication. 13th IFIP International Conference on Information Security (IFIP/Sec '97), Proc., Chapman & Hall, London 1997, 164-174.
- [RSA\_78] Ronald L. Rivest, Adi Shamir, Leonard Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21/2 (1978) 120-126, reprinted: 26/1 (1983) 96-99.
- [RWHP\_89] Alexander Roßnagel, Peter Wedde, Volker Hammer, Ulrich Pordesch: Die Verletzlichkeit der "Informationsgesellschaft". Sozialverträgliche Technikgestaltung Band 5, Westdeutscher Verlag, Opladen 1989.
- [RaPM\_96] Kai Rannenberg, Andreas Pfitzmann, Günter Müller: Sicherheit, insbesondere mehrseitige IT-Sicherheit. it+ti 38/4 (1996) 7-10.
- [Rann\_94] Kai Rannenberg: Kolleg "Sicherheit in der Kommunikationstechnik" eingerichtet. Manuscript, V. 2.0, 9.9.1994, per e-mail erhalten am 28.11.1994.
- [Rann\_97] Kai Rannenberg: Tragen Zertifizierung und Evaluationskriterien zu mehrseitiger Sicherheit bei? in: Günter Müller, Andreas Pfitzmann (Hrsg.): Mehrseitige Sicherheit in der Kommunikationstechnik, Addison-Wesley-Longman 1997, 527-562.
- [Rede\_84] Helmut Redeker: Geschäftsabwicklung mit externen Rechnern im Bildschirmtextdienst. Neue Juristische Wochenschrift NJW /42 (1984) 2390-2394.
- [Rede\_86] Helmut Redeker: Rechtliche Probleme von Mailbox-Systemen. GI - 16. Jahrestagung II, "Informatik-Anwendungen - Trends und Perspektiven"; Berlin, Oktober 1986, Proceedings herausgegeben von G. Hommel und S. Schindler, IFB 127, Springer-Verlag, Berlin 1986, 253-266.
- [Rei1\_86] Peter O'Reilly: Burst and Fast-Packet Switching: Performance Comparisons. IEEE INFOCOM '86, Fifth Annual Conference "Computers and Communications Integration - Design, Analysis, Management", April 8-10, 1986, Miami, Florida, 653-666.
- [Riel\_99] Herman te Riele: Factorization of a 512-bits RSA key using Number Field Sieve. e-Mail of Aug. 26, 1999.

## Bibliography

- [Riha2\_96] Karl Rihaczek: Die Kryptokontroverse: das Normungsverbot. Datenschutz und Datensicherheit DuD 20/1 (1996) 15-22.
- [Riha\_84] Karl Rihaczek: Datenverschlüsselung in Kommunikationssystemen - Möglichkeiten und Bedürfnisse. DuD Fachbeiträge 6, Vieweg 1984.
- [Riha\_85] Karl Rihaczek: Der Stand von OSIS. Datenschutz und Datensicherung DuD 9/4 (1985) 213-217.
- [Rive\_98] Ronald L. Rivest: Chaffing and Winnowing: Confidentiality without Encryption. MIT Lab for Computer Science, March 18, 1998.
- [RoSc\_96] Alexander Roßnagel, Michael J. Schneider: Anforderungen an die mehrseitige Sicherheit in der Gesundheitsversorgung und ihre Erhebung. it+ti 38/4 (1996) 15-19.
- [Roch\_87] Edouard Y. Rocher: Information Outlet, ULAN versus ISDN. IEEE Communications Magazine 25/4 (1987) 18-32.
- [Rose\_85] K. H. Rosenbrock: ISDN - Die folgerichtige Weiterentwicklung des digitalisierten Fernsprechnetzes für das künftige Dienstleistungsangebot der Deutschen Bundespost. GI/NTG-Fachtagung "Kommunikation in Verteilten Systemen - Anwendungen, Betrieb und Grundlagen -", 11.-15. März 1985, Tagungsband 1, D. Heger, G. Krüger, O. Spaniol, W. Zorn (Hrsg.), IFB 95, Springer-Verlag, Berlin, 202-221.
- [Rul1\_87] Christoph Ruland: Datenschutz in Kommunikationssystemen. DATACOM Buchverlag, Pulheim, 1987.
- [SFJK\_97] Reiner Sailer, Hannes Federrath, Anja Jerichow, Dogan Kesdogan, Andreas Pfitzmann: Allokation von Sicherheitsfunktionen in Telekommunikationsnetzen. in: Günter Müller, Andreas Pfitzmann (Hrsg.): Mehrseitige Sicherheit in der Kommunikationstechnik, Addison-Wesley-Longman 1997, 325-357.
- [SIEM\_87] SIEMENS: Internationale Fernsprechstatistik 1987. Stand 1. Januar 1986; SIEMENS Aktiengesellschaft N ÖV Marketing, Postfach 70 00 73, D-8000 München 70, Mai 1987.
- [SIEM\_88] SIEMENS: Vermittlungsrechner CP 113 für ISDN. SIEMENS telcom report 11/2 (1988) 80.
- [ScS1\_84] Christian Schwarz-Schilling (ed.): ISDN - die Antwort der Deutschen Bundespost auf die Anforderungen der Telekommunikation von morgen. Herausgeber: Der Bundesminister für das Post- und Fernmeldewesen, Bonn, 1984.
- [ScSc\_73] A. Scholz, B. Schoeneberg: Einführung in die Zahlentheorie. (5. Aufl.) Sammlung Göschen, de Gruyter, Berlin 1973.

## Bibliography

- [ScSc\_83] Richard D. Schlichting, Fred B. Schneider: Fail-Stop Processors: An Approach to Designing Fault-Tolerant Computing Systems. ACM Transactions on Computer Systems 1/3 (1983) 222-238.
- [ScSc\_84] Christian Schwarz-Schilling (ed.): Konzept der Deutschen Bundespost zur Weiterentwicklung der Fernmeldeinfrastruktur. Herausgeber: Der Bundesminister für das Post- und Fernmeldewesen, Stab 202, Bonn, 1984.
- [ScSc\_86] Christian Schwarz-Schilling (ed.): Mittelfristiges Programm für den Ausbau der technischen Kommunikationssysteme. Herausgeber: Der Bundesminister für das Post- und Fernmeldewesen, Bonn, 1986.
- [Schö\_84] Helmut Schön: Die Deutsche Bundespost auf ihrem Weg zum ISDN. Zeitschrift für das Post- und Fernmeldewesen /6 1984.
- [Schö\_86] Helmut Schön: ISDN und Ökonomie. Jahrbuch der Deutschen Bundespost 1986.
- [Schn\_96] Bruce Schneier: Applied Cryptography: Protocols, Algorithms, and Source Code in C. (2nd ed.) John Wiley & Sons, New York 1996.
- [Sedl\_88] H. Sedlak: The RSA cryptography processor. Eurocrypt '87, LNCS 304, Springer-Verlag, Berlin 1988, 95-105.
- [Sha1\_49] C. E. Shannon: Communication Theory of Secrecy Systems. The Bell System Technical Journal 28/4 (1949) 656-715.
- [Sham\_79] Adi Shamir: How to Share a Secret. Communications of the ACM 22/11 (1979) 612-613.
- [Shan\_48] C. E. Shannon: A Mathematical Theory of Communication. The Bell System Technical Journal 27 (1948) 379-423, 623-656.
- [Silv\_91] Robert D. Silverman: Massively Distributed Computing and Factoring Large Integers. Communications of the ACM 34/11 (1991) 95-103.
- [Sim3\_88] Gustavus J. Simmons: A Survey of Information Authentication. Proceedings of the IEEE 76/5 (1988) 603-620.
- [Simm\_88] Gustavus J. Simmons: Message authentication with arbitration of transmitter/receiver disputes. Eurocrypt '87, LNCS 304, Springer-Verlag, Berlin 1988, 151-165.
- [Steg\_85] H. Stegmeier: Einfluß der VLSI auf Kommunikationssysteme. GI-/NTG-Fachtagung "Kommunikation in Verteilten Systemen - Anwendungen, Betrieb und Grundlagen -", 11.-15. März 1985, Tagungsband 1, D. Heger, G. Krüger, O. Spaniol, W. Zorn (Hrsg.), IFB 95, Springer-Verlag, Berlin, 663-672.

## Bibliography

- [Stel\_90] D. Stelzer: Kritik des Sicherheitsbegriffs im IT-Sicherheitsrahmenkonzept. *Datenschutz und Datensicherung* DuD14/10 (1990) 501-506.
- [SuSY\_84] Tatsuya Suda, Mischa Schwartz, Yechiam Yemini: Protocol Architecture of a Tree Network with Collision Avoidance Switches. *Links for the Future; Science, Systems & Services for Communications*, P. Dewilde and C. A. May (eds.); *Proceedings of the International Conference on Communications - ICC '84*, Amsterdam, The Netherlands, May 14-17, 1984, IEEE, Elsevier Science Publishers.
- [Sze\_85] Daniel T. W. Sze: A Metropolitan Area Network. *IEEE Journal on Selected Areas in Communications* 3/6 (1985) 815-824.
- [Tane\_81] Andrew S. Tanenbaum: *Computer Networks*. Prentice-Hall, Englewood Cliffs, N. J., 1981.
- [Tane\_88] Andrew S. Tanenbaum: *Computer Networks*. 2nd ed., Prentice-Hall, Englewood Cliffs 1988.
- [Tane\_96] Andrew S. Tanenbaum: *Computer Networks*. 3rd ed., Prentice-Hall, Upper Saddle River 1996.
- [Tele\_00] Deutsche Telekom: Das kleine Lexikon (Beilage zu "Der Katalog. Telekommunikation für zu Hause und unterwegs."). Frühjahr/Sommer 2000, 8.
- [Tele\_98] Telemediarecht: Telekommunikations- und Multimediarecht. Deutscher Taschenbuch Verlag, 1. Auflage 1998, 1-321.
- [ThFe\_95] Jürgen Thees, Hannes Federrath: Methoden zum Schutz von Verkehrsdaten in Funknetzen. *Verlässliche IT-Systeme, GI-Fachtagung VIS '95*, DuD Fachbeiträge, Vieweg, Braunschweig 1995, 180-192.
- [Thom\_84] Ken Thompson: Reflections on Trusting Trust. *Communications of the ACM* 27/8 (1984) 761-763.
- [Thom\_87] Karl Thomas: Der Weg zur offenen Kommunikation. *nachrichten elektronik+telematik net special "ISDN - eine Idee wird Realität"*; Sondernummer Oktober 87, R. v. Decker's Verlag, 10-17.
- [ToWo\_88] Martin Tompa, Heather Woll: How to Share a Secret with Cheaters. *Journal of Cryptology* 1/2 (1988) 133-138.
- [VaVa\_85] Umesh V. Vazirani, Vijay V. Vazirani: Efficient and Secure Pseudo-Random Number Generation (extended abstract). *Crypto '84*, LNCS 196, Springer-Verlag, Berlin 1985, 193-202.

## Bibliography

- [VoKe\_83] Victor L. Voydock, Stephen T. Kent: Security Mechanisms in High-Level Network Protocols. ACM Computing Surveys 15/2 (1983) 135-171.
- [VoKe\_85] Victor L. Voydock, Stephen T. Kent: Security in High-level Network Protocols. IEEE Communications Magazine 23/7 (1985) 12-24.
- [WaP1\_87] Michael Waidner, Birgit Pfitzmann: Anonyme und verlusttolerante elektronische Brieftaschen. Interner Bericht 1/87 der Fakultät für Informatik, Universität Karlsruhe 1987.
- [WaPP\_87] Michael Waidner, Birgit Pfitzmann, Andreas Pfitzmann: Über die Notwendigkeit genormter kryptographischer Verfahren. Datenschutz und Datensicherung DuD 11/6 (1987) 293-299.
- [WaPf1\_89] Michael Waidner, Birgit Pfitzmann: The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability. Universität Karlsruhe 1989.
- [WaPf\_85] Michael Waidner, Andreas Pfitzmann: Betrugssicherheit trotz Anonymität. Abrechnung und Geldtransfer in Netzen. 1. GI Fachtagung Datenschutz und Datensicherung im Wandel der Informationstechnologien, IFB 113, Springer-Verlag, Berlin 1985, 128-141; Überarbeitung in: Datenschutz und Datensicherung DuD 10/1 (1986) 16-22.
- [WaPf\_89] Michael Waidner, Birgit Pfitzmann: Unconditional Sender and Recipient Untraceability in spite of Active Attacks - Some Remarks. Fakultät für Informatik, Universität Karlsruhe, Interner Bericht 5/89, März 1989.
- [WaPf\_90] Michael Waidner, Birgit Pfitzmann: Loss-Tolerance for Electronic Wallets. 20th Int. Symp. on Fault-Tolerant Computing (FTCS) 1990, IEEE Computer Society Press, Los Alamitos 1990, 140-147.
- [Waer\_67] B.L. van der Waerden: Algebra II. Heidelberger Taschenbücher 23, Springer-Verlag, Berlin 1967, 5. Auflage.
- [Waer\_71] B.L. van der Waerden: Algebra I. Heidelberger Taschenbücher 12, Springer-Verlag, Berlin 1971, 8. Auflage.
- [Waid\_85] Michael Waidner: Datenschutz und Betrugssicherheit garantierende Kommunikationsnetze. Systematisierung der Datenschutzmaßnahmen und Ansätze zur Verifikation der Betrugssicherheit. Diplomarbeit am Institut für Informatik IV, Universität Karlsruhe, August 1985, Interner Bericht 19/85 der Fakultät für Informatik.

## Bibliography

- [Waid\_90] Michael Waidner: Unconditional Sender and Recipient Untraceability in spite of Active Attacks. Eurocrypt '89, LNCS 434, Springer-Verlag, Berlin 1990, 302-319.
- [Walk\_87] Bernhard Walke: Über Organisation und Leistungskenngrößen eines dezentral organisierten Funksystems. Kommunikation in Verteilten Systemen; Anwendungen, Betrieb, Grundlagen; GI/NTG-Fachtagung, Aachen, Februar 1987, IFB 130, herausgegeben von N. Gerner und O. Spaniol, Springer-Verlag, Berlin, 578-591.
- [WeCa\_79] Mark N. Wegman, J. Lawrence Carter: New Classes and Applications of Hash Functions. 20th Symposium on Foundations of Computer Science (FOCS) 1979, IEEE Computer Society, 1979, 175-182.
- [WeCa\_81] Mark N. Wegman, J. Lawrence Carter: New Hash Functions and Their Use in Authentication and Set Equality. Journal of Computer and System Sciences 22 (1981) 265-279.
- [Weck1\_98] Gerhard Weck: Key Recovery - Empfehlungen. Informatik-Spektrum 21/3 (1997) 157-158.
- [Weck\_98] Gerhard Weck: Key Recovery - Möglichkeit und Risiken. Informatik-Spektrum 21/3 (1997) 147-157.
- [Wein\_87] Steve H. Weingart: Physical Security for the microABYSS System. Proc. 1987 IEEE Symp. on Security and Privacy, April 27-29, 1987, Oakland, California, 52-58.
- [West\_97] Andreas Westfeld: Steganographie am Beispiel einer Videokonferenz. in: Günter Müller, Andreas Pfitzmann (Hrsg.): Mehrseitige Sicherheit in der Kommunikationstechnik, Addison-Wesley-Longman 1997, 507-526.
- [WiHP\_97] Gutram Wicke, Michaela Huhn, Andreas Pfitzmann, Peter Stahlknecht: Kryptoregulierung. Wirtschaftsinformatik 39/3 (1997) 279-282.
- [Wich\_93] Peer Wichmann: Die DSSA Sicherheitsarchitektur für verteilte Systeme. Datenschutz und Datensicherung DuD 17/1 (1993) 23-27.
- [Wien\_90] Michael J. Wiener: Cryptanalysis of Short RSA Secret Exponents. IEEE Transactions on Information Theory 36/3 (1990) 553-558.
- [ZFPW\_97] Jan Zöllner, Hannes Federrath, Andreas Pfitzmann, Andreas Westfeld, Guntram Wicke, Gritta Wolf: Über die Modellierung steganographischer Systeme. Verlässliche IT-Systeme, GI-Fachtagung VIS '97, DuD Fachbeiträge, Vieweg 1997, 211-223.

## *Bibliography*

- [ZSI\_89] Zentralstelle für Sicherheit in der Informationstechnik (Hrsg.): IT-Sicherheitskriterien; Kriterien für die Bewertung der Sicherheit von Systemen der Informationstechnik (IT). 1. Fassung vom 11.1.1989; Köln, Bundesanzeiger 1989.
- [ZSI\_90] Zentralstelle für Sicherheit in der Informationstechnik (Hrsg.): IT-Evaluationshandbuch; Handbuch für die Prüfung der Sicherheit von Systemen der Informationstechnik (IT). 1. Fassung vom 22.2.1990; Köln, Bundesanzeiger 1990.
- [Zimm\_93] Philip Zimmermann: PGP - Pretty Good Privacy, Public Key Encryption for the Masses, User's Guide; Volume I: Essential Topics, Volume II: Special Topics. Version 2.2 - 6 March 1993, now Version 2.6.2, 11 Oct. 1994; see <http://www.mantis.co.uk/pgp/pgp.html>; for sources.

# A Exercises

## 1 Exercises for “Introduction”

### 1-1 Link between spatial distribution and distribution in terms of control- and implementation structure.

Information technical systems (IT-systems) that are spatially distributed are also exclusively realized in a distributed way regarding their control- and implementation structure. Why?

(As explanation: *control structure* = Which instances are issuing orders, instructions etc. to other instances; Convince yourself whether or not to execute and how. *implementation structure* = How are these instances implemented, e.g., many on one computer, e.g., single instances on many cooperating computers, etc.)

[Solution at page 447.](#)

### 1-2 Advantages and disadvantages of a distributed system in terms of security

- Which advantages and disadvantages does a distributed system offer in terms of security?
- Which security properties are supported by spatial distribution, and which by distribution in terms of control- and implementation structure?

[Solution at page 447.](#)

### 1-3 Advantages and disadvantages of an open system in terms of security

- Which advantages and disadvantages does an open system offer in terms of security?
- How are these advantages and disadvantages combined in terms of security in open distributed systems?

[Solution at page 448.](#)

### 1-4 Advantages and disadvantages of a service-integrating system in terms of security

- Which advantages and disadvantages does a service-integrating system offer in terms of security?
- How are these advantages and disadvantages in terms of security combined in open distributed service-integrating systems?

[Solution at page 448.](#)

## A Exercises

### 1-5 Advantages and disadvantages of a digital system in terms of security

- a) Which advantages and disadvantages does a digital system offer in terms of security? Differentiate between digital transmission/storage on the one hand and computer-driven transmission, computer-driven processing on the other hand. What is changing if computers can be programmed freely (program being in the RAM instead of being in the ROM)?
- b) What relationships exist between digital systems and the properties 1) distributed, 2) open and 3) service-integrating?

[Solution at page 449.](#)

### 1-6 Definition of requirements for four exemplary applications in the medical domain

Create protection goals that are as complete as possible for the following applications!

- a) Medical records of many patients are stored in a database and are accessed by physicians from all over the world.
- b) During medical surgeries external experts are consulted via videophone. Which additional requirements arise if the surgery team is dependent on advice from the site?
- c) Patients are to be medically monitored in their private apartments. So that, on the one hand, they are able to spend less time in the hospital and on the other hand, it can be used to quickly and automatically call for help, e.g., in case of unconsciousness.
- d) Medical and psychological fact-databases can be consulted by patients so as to become informed in a more comfortable, detailed, and up-to-date way than through books and to save the doctor's time (which they often do not take regardless).

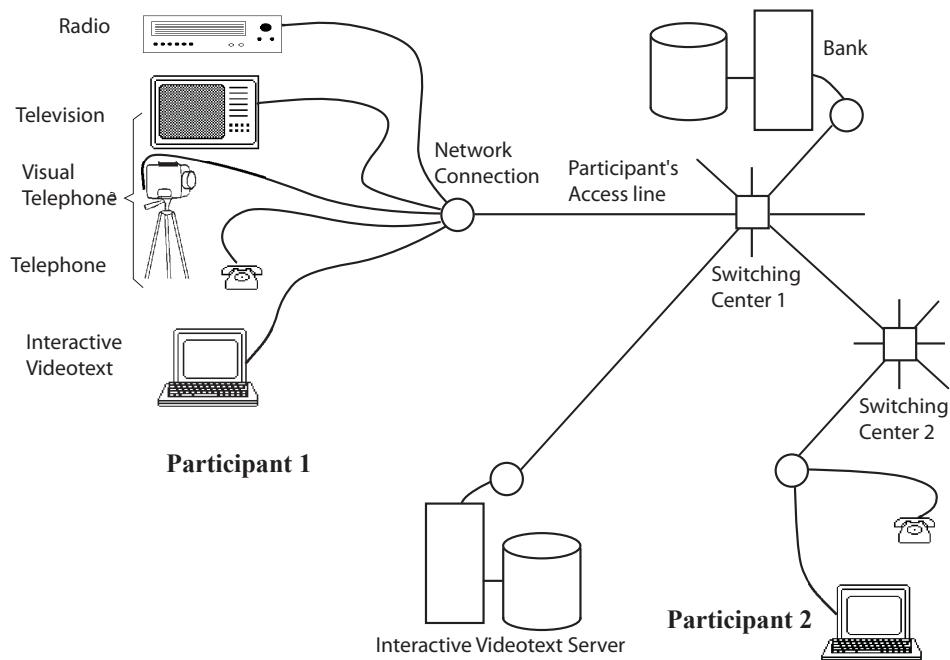
Are you sure of the completeness of your protection goals?

[Solution at page 450.](#)

### 1-7 Where and when can computers be expected?

Mark the locations in the following figure by filling out the number  $n$  where you expect computers as of year  $n$ . Mark by adding / <role declaration>, who programmed this computer and who is able to program this computer. What does this mean for security?

## 2 Exercises for “Security in single computers and its limits”



[Solution at page 451.](#)

### 1-8 Deficits of legal provisions

Why do legal provisions not suffice for, e.g., legal certainty and data protection?

[Solution at page 451.](#)

### 1-9 Alternative definitions/characterizations of multilateral security

In §1.4 a definition of multilateral security is given. Since this definition is rather a colloquial characterization than a strict definition of a scientific issue: Come up with at least 3 alternative definitions/characterizations or collect them from literature. You can make a find in [Cha8\_85, Chau\_92, PWP\_90, MüPf\_97, FePf\_99].

[Solution at page 452.](#)

## 2 Exercises for “Security in single computers and its limits”

### 2-1 Attackers beyond the wisdom of textbooks

Think of attackers that do not obey the currently known laws of nature, as demanded in §2.1.1, and against whom security cannot be achieved in principle.

[Solution at page 454.](#)

## A Exercises

### 2-2 (In)Dependence of confidentiality, integrity, availability with identification as example

In §1.2.1 it was differentiated between the protection goals confidentiality, integrity, and availability. Discuss on the basis of the example identification (§2.2.1), if “secure” IT-systems can be expected for which no requirements in terms of at least one of the three protection goals exist.

[Solution at page 454.](#)

### 2-3 Send random number, expect encryption – does this work conversely, too?

In §2.2.1 a small example protocol is given for identification:

- generate a random number, send it;
- await encryption of the random number. Check it.

Is it possible to execute this protocol conversely:

- generate random number, encrypt it, send the encrypted random number;
- await decrypted random number. Check it.

What will be changed by this inversion? (Hint: Which assumptions about the generation of the random numbers must be made in each version?)

[Solution at page 455.](#)

### 2-4 Necessity of logging in case of access control

- a) Log data (secondary data) are personal data as well as primary data. Therefore log data must be protected against unauthorized access like the primary data. It is a vicious circle – we are now again supposed to log the access to the secondary data (tertiary data) and so on. Why is this problem solved in a satisfactory way in spite of recursivity?
- b) What should one aspire?
- c) In terms of authorization, similar things as described in a) apply for the log data: To control who is allowed to set rights (authorization), rules themselves must be set (and enforced), who is allowed to do this, etc. It looks suspiciously like a vicious circle again: Do similar things (as in a) apply regarding the solution?

[Solution at page 455.](#)

### 2-5 Limiting the success of modifying attacks

In §1.2.5 it was mentioned that modifying attacks can be recognized in principle, but their success can not be prevented completely. Which measures must be taken to make modifying attacks as ineffective as possible? Consider especially how you would organize backups for data and programs.

[Solution at page 456.](#)

## 2-6 No hidden channel anymore?

Some top-ranking officers are extremely worried about keeping their plans for the defense of their country secret. So they decide after having read the first two chapters of this script: We will never trust that delivered devices are free of Trojan horses that want to reveal our defense secrets. Hence we will put these devices in airtight rooms and close all channels that go outside by

- a) disconnecting wires with physical switches (reasons are clear!) in times of peace,
- b) keeping the current power consumption constantly in the computer center (thus information cannot escape outside by modulating power consumption),
- c) not giving the devices back to the manufacturer for repair (because information can be stored in them, too).

They announce to their minister of defense: Finally, no bit will escape from our computers in times of peace! Are they right?

[Solution at page 457.](#)

## 2-7 Change of perspective: the diabolical terminal

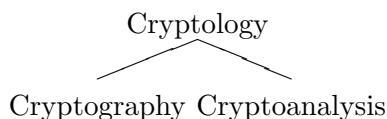
- a) The security of each participant can not be better than the terminal, i.e., the device one is interacting with directly, is secure for him/her. Suppose you are Advocatus Diaboli: Come up with a terminal with as many properties as possible that undermine the security of its user, but which the user still wishes to use.
- b) Compare your suggestions from a) with PCs that are offered to you today.

[Solution at page 457.](#)

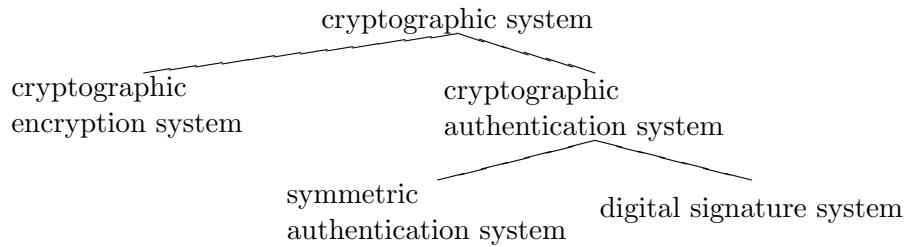
# 3 Exercises for “Cryptography”

## 3-1 Terms

Trees of terms are shown in the following figure. Explain the textual structure of those terms and the advantages and disadvantages of using the short terms *crypto-system* resp. *encryption* and *decryption* in general for cryptographic systems or especially only for encryption systems.



## A Exercises



[Solution at page 459.](#)

### 3-2 Domains of trust for asymmetric encryption systems and digital signatures

Domains of trust and areas of attack are marked in a way in Figure 3.1 that no statement is made about the generation of the key. It is clear that it must take place within a domain of trust. Whether in the domain of trust of the encrypter, of the decrypter or in an additional one stays open and should be different depending on the application used:

In case of encryption of a local storage medium the generation of the key, encryption, and decryption should take place in one and the same device – then the key does ever not need to leave the device. Thus the key can be optimally protected in accordance with confidentiality and integrity. (Please remember that encryption does not contribute anything to availability and that backup copies, which are stored on this medium, of the plaintext or of the ciphertext as well as of the decryption key, must be made.) In case of direct exchange of the keys between encrypter and decrypter outside of the considered communication network, e. g., via USB stick, the key should be generated within one of the domains of trust.

In case of key exchange within the communication network according to Figure 3.3, the generation of the key can take place at the encrypter, the decrypter, or at the key exchange center. It does not matter in which of the three domains of trust this takes place because the key exists unencrypted in each domain of trust anyway while the keys are distributed.

Everything that has been said up to now is of course not only valid for symmetric encryption systems, but also for symmetric authentication.

And now the question: How would you map the domains of trust in case of asymmetric encryption systems and digital signature systems? Do it – and explain your decision.

[Solution at page 459.](#)

### 3-3 Why keys?

It is stated in §3.1.1 that cryptographic systems use keys.

- Would it work without keys, too? If yes, how?
- What disadvantages would arise?
- What are the advantages of using keys, in your opinion?

[Solution at page 461.](#)

### 3-4 Adjustment of keys in case of autonomous key generation by participants?

Exercise 3-2 illustrates why the key generation should take place in the device where the secret part of the key pair is used in case of asymmetric cryptographic systems.

Occasionally there are objections: Since the generated key pairs must be unique (obviously this is especially important when using digital signature systems) they can not be generated autonomously in a decentralized way by participants because doubled keys (two participants generate independent of each other the same key pair) can not be eliminated completely. In order to achieve this, so-called TrustCenters should – technically-organizational units that provide different services that are (hopefully) trustworthy and beneficial for security – also take over the function of key generation, too.

What do you think about this objection and suggestion?

[Solution at page 462.](#)

### 3-5 Increasing security by using multiple cryptographic systems

What can you do if you have multiple cryptographic systems, all based on different security assumptions, but you do not trust any one of them (resp. any one of the security assumptions) enough to rely on it alone? Differentiate between the protection goals secrecy (i.e., you have several encryption systems) and authentication (i.e., you have several authentication systems). Which effort for computation, storage, and transmission does this measure cause? (The last one only roughly for encryption.)

[Solution at page 462.](#)

### 3-6 Protocols for key exchange

- a) All described protocols for key exchange assume that both participants that want to exchange a key have exchanged keys for a symmetric or asymmetric cryptosystem with a *common* key exchange center or a *common* public-key register, respectively. What must be changed if this is not the case? For instance, both participants could live in different countries and could have exchanged their keys only with key exchange centers resp. public-key registers in their own countries.
- b) *Optional:* How should your solution for the exchange of symmetric keys be designed if each participant wants to use multiple key exchange centers in order to put as little trust in each individual key exchange center as possible?
- c) Parallel use of key exchange centers was demonstrated and explained for symmetric cryptosystems. Is such a parallel use useful for asymmetric cryptosystems as well? How should it happen? What does it accomplish?
- d) *Optional:* Does a protocol for key exchange exist that is secure despite of the following attacker model: Either the attacker receives passive assistance of all key exchange centers, i.e., they reveal all exchanged keys, or (exclusive!) the attacker is complexity-theoretically unlimited, i.e., he is able to break any asymmetric encryption system (and any normal digital signature system). It is implied additionally

## A Exercises

that the attacker wiretaps all communication in the network. How does a protocol for key distribution, if applicable, look like? Which cryptographic system should be used for encryption, which for authentication after key exchange?

- e) What happens if an attacker commits a modifying attack during symmetric key exchange? This could happen by changing messages on the wire or by a key exchange center that distributes inconsistent symmetric keys. What should be done to defend participants from this? Treat the case of one key exchange center first, then the case of multiple centers in sequence (for instance in case of multiple hierarchical key distribution, cp. exercise part a) ) and then the case of multiple centers in parallel (cp. §3.1.1.1).
- f) Freshness of keys: How far and how can it be guaranteed that an attacker is not able to persuade participants to use old (and for instance compromised) keys that have already been replaced by new keys? (It is implied that keys of the “right” partner are involved, too.)
- g) Exchange of public keys without public-key register: anarchic key exchange à la PGP

We assume that we can use no public-key registers (be it that the country or the telecommunication provider are not operating one, be it that some citizens do not want to trust central public-key registers, be it that there are public-key registers, but these are generating the key pairs themselves – Honi soit qui mal y pense). Nevertheless (or hence) public keys are supposed to be exchanged in a participant group everybody can join.

As repetition: What must be payed attention to in the case of exchanging public keys? Who could provide the function of the trustworthy public-key register? How could key exchange take place then?

What must be done if not everybody trusts everybody else in the same way?

What must be done when a participant discovers that his secret key is known by somebody else?

What should a participant do when his secret key is destroyed?

*Optional:* What must be done when a participant discovers that he had certified a false assignment of participant  $\Leftrightarrow$  public key?

Is it possible to combine the method created in this part of the exercise with the method created in part b) of this exercise?

[Solution at page 465.](#)

### 3-7 What kind of Trust is necessary for which function of a TrustCenter?

Some authors (as we will see in a moment) assign the following functions to so-called TrustCenters:

- a) *key generation:* The generation of key pairs for participants

- b) *key certification*: Certification of the relation between public key and participant, cp. §3.1.1.2
- c) *directory service*: Keeping certified public keys ready for request from arbitrary participants.
- d) *key revocation*: Blocking public keys, among others on request of the key owner, i.e., issuing a certificated statement that (and from when on etc.) the public key is blocked.
- e) *time stamp service*: Digitally certification of submitted messages incl. current date and time.

Discuss which kinds of trust of the participant this requires. The distinction between **blind trust** (there is no way for the participant to check whether his/her trust is justified) and **checking trust** (participant is able to check whether his trust is justified) could be useful. How are these properties related to the attacker model described in §1.2.5, and especially to the distinction between *observing* and *modifying attacks*?

[Solution at page 476.](#)

### 3-8 Hybrid encryption

You want to exchange huge files and secure this exchange – for efficiency reasons – with a hybrid cryptosystem. How will you proceed if

- a) only secrecy
- b) as well as authentication
- c) only authentication

is important for you?

Contemplate case a) first, where a sender  $S$  sends files to a recipient  $R$ , then case b), where  $S$  sends files one after another to  $R$ , and finally case c), where  $R$  answers, i.e., sends files to  $S$ .

[Solution at page 477.](#)

### 3-9 Situations from practice (for systematics of cryptographic systems)

Specify for the situations which cryptographic system you would use (encryption/authentication, symmetric/asymmetric) and how you would carry out key distribution. Additionally you could specify roughly how critical in terms of security and time the system is. (E.g., Vernam cipher in hardware: 10 Gbit/s, in software 100 Mbit/s; information-theoretically secure authentication codes in hardware 5 Gbit/s, in software 100 Mbit/s; DES in hardware: 200 Mbit/s, in software up to 10 Mbit/s, depending on the PC you are using; RSA in hardware: 200 kbit/s, in software 6000 bit/s (GMR,  $s^2 - \text{mod} - n$ -generator probably likewise).)

## A Exercises

- a) Andreas looks for flats in Hildesheim. He wants to write Birgit an email with advantages and disadvantages of the flats and wants to receive a decision if he is supposed to sign a tenancy agreement. (Both are so hoarse that they are not able to make a phone call.)
- b) David again has a brilliant idea for a new cryptographic system and wants to send it to a few trustworthy cryptographers (via file transfer). Unfortunately less ingenious cryptographers are lurking in most computer centers in order to steal David's ideas.
- c) A secure operating system flushes out data to a removable disk. When the data is read back, the operating system has to make sure the data has not been modified.
- d) A fan of computers wants to order a new built-to-order computer on the basis of a digital catalog with a digital message.
- e) A mechanical engineering company sends a fax to another company with a non-legally binding inquiry, as to whether the company was able to manufacture a certain piece and at what price.

[Solution at page 478.](#)

### 3-10 Vernam cipher

- a) Calculate a small example, whereby we agree upon for now and for the following that processed bits are written from left to right.
- b) The security of the Vernam cipher (with the addition of modulo 2) was proven in the lecture. Prove this for addition in arbitrary groups.
- c) Decrypt the ciphertext 01011101, which was encrypted with the key 10011011. Is the result unique?
- d) An attacker intercepts the ciphertext 010111010101110101011101 and wants to invert the third bit from the right of the corresponding plaintext. What must the attacker do if the ciphertext is computed modulo 2? What must the attacker do if the ciphertext is computed modulo 4, modulo 8, modulo 16, resp., thus 2, 3, 4 bits, resp., are added "together". In which sense is this exercise for modulo 4 and modulo 16 solvable, in which sense not?
- e) Why is no adaptive active attack possible on the *key* of a Vernam cipher?
- f) There are 16 functions  $\{0,1\} \times \{0,1\} \rightarrow \{0,1\}$ . Which are suitable as encryption functions or decryption functions, resp., for the Vernam cipher? Specify necessary and sufficient conditions for functions of arbitrary Vernam ciphers first, then apply them to the binary case.

- g) A younger cryptographer has the following idea for decreasing the effort of key exchange: Instead of a random key, I do use a part of a book which my partner owns, too. Then we only have to agree on book, page, line, and row of the beginning and we can add or subtract, resp., modulo the number of characters in the alphabet of the book as long as the book is long enough.  
Which advice and why will you give as an older cryptographer to your younger friend?
- h) In some countries it is allowed to send encrypted messages, but one has to store the key used and has to hand it out on request to an authority of law enforcement or the like. Does this make sense for the Vernam cipher? (It is implied of course that the authority wiretaps all ciphertexts and stores them in the right order and is able to assign them to sender and recipient.)
- i) How long is it possible to encrypt with the information-theoretically secure Vernam cipher
  - 1) text communication (typing velocity: 40 bit/s),
  - 2) voice communication (with the normal ISDN-encoding: 64 kbit/s), or
  - 3) high-resolution full-video-communication (140 Mbit/s), resp.,

if one has exchanged

- 1) a 3.5" floppy disk (1.4 MByte),
- 2) CD-R (700 MByte),
- 3) DVD one sided (4.7 GByte),
- 4) Blu-ray-ROM (50 GByte)

full of random bits?

As subsumption: The first two media are standard technology at least since 1990 and widespread in large quantities. DVD is very popular today, the next generation optical media is Blu-ray with a capacity of 50 GB. It serves as an example for estimating what will be available for the end-user in the next years.

See p. #50

See p. #74

See p. #74

[Solution at page 479.](#)

### 3-11 Symmetrically encrypted Message Digest = MAC ?

- a) Every now and then one hears about the following recipe for generating the Message Authentication Code (MAC):
  - 1) Create a check digit for the message using a publicly known procedure and encrypt the check digit with a symmetric encryption system.

## A Exercises

The recipient decrypts the check digit and checks whether it fits by calculating the appropriate check digit for the received message using a publicly known procedure. If both check digits are equal, the message is regarded as authentic.

What do you think about this recipe?

(As a hint: Take the most possible secure procedure for creation of check digits, i.e., a collision-resistant hash-function, and the most secure symmetric encryption system, i.e., the Vernam cipher, and then contemplate whether the combination is secure.)

- b) What do you think about the following improvement: Instead of encrypting the hash-value with a symmetric key, the symmetric key is hashed together with the message (as for example in the Transport Layer Security (TLS) protocol). If you have the opinion that this is not a secure construction for all collision-resistant hash-functions: Which additional property must the collision-resistant hash-function have? (As a starting point for your consideration: Could the hash-value reveal something about the symmetric key?)
- c) If the construction given in b) for symmetric authentication is secure for a particular hash-function: Are you then able to construct a symmetric encryption system using this hash-function? (This would be exciting, because the following opinion is widespread in the secret service: good collision-resistant hash-functions are allowed to be known by enemies, while good encryption systems must be kept secret from them. In Germany, for instance, scientific publications in the cryptographic area by the BSI exclusively deal with hash-functions, cp. e.g., [Dobb\_98].)

Solution at page 485.

### 3-12 Authentication codes

- a) Calculate a small example for an authentication code with 4 bits by using the authentication code represented in the following table. (Figure 3.15). H (Head) and T (Tail) are the plaintext characters.

Text with MAC				
	H,0	H,1	T,0	T,1
00	H	-	T	-
01	H	-	-	T
10	-	H	T	-
11	-	H	-	T

- b) Why is the authentication code information-theoretically secure? (You can continue or summarize the proof shown in the lecture, or represent the code easier first.)

- c) If one uses this code in order to authenticate a message that consists of several bits (as in exercise part a) or example 2 from the lecture), will an attacker then be able to change the message unnoticed just by permuting the order of the bits including their MACs?
- d) Calculate a small example for secrecy and authentication of 4 bits using the authentication code given in the following table. (It effects secrecy and authentication as well.)  
 Why is this code information-theoretically secure for arbitrary attacks on authentication and only information-theoretically secure for observing attacks on secrecy? (It may be implied that the attacker gets to know if its attack was successful or not in case of a modifying attack.)

		ciphertext			
		00	01	10	11
k	00	H	-	T	-
	01	T	-	-	H
	10	-	T	H	-
	11	-	H	-	T

- e) *Optional:* Modify the given secrecy and authentication code in order to make it resist modifying attacks regarding to secrecy. Is your system more efficient than just using an authentication code and the Vernam cipher?
- f) The given authentication codes have two disadvantages: Both are working on single bits (H and T) so that the MAC must be attached on each bit which is not efficient for many applications. Furthermore forging attempts are only detected and denied with a probability of 0.5. A probability close to 1 would be desirable. The following authentication code by Mark N. Wegman and J. Lawrence Carter [WeCa\_81] does not have this disadvantage: Messages  $m$  are elements of a huge finite field  $K$  (for instance addition and multiplication modulo a huge prime number, which is known to everybody). Keys are pairs  $(a, b)$  of elements of  $K$  (and are only used once). MACs are created as:

$$MAC := a + b * m$$

Please prove: If an attacker observes  $m$  and  $MAC$  then he is not able to create the appropriate  $MAC'$  for another message  $m'$  more purposefully than by guessing. His success probability is at most  $1/|K|$ .

[Solution at page 486.](#)

### 3-13 Mathematical secrets

- a) (*Only roughly:*) Generation of prime numbers is often programmed in the following way: Only at the beginning an odd random number  $p$  is chosen. If this number is not prime, 2 is added to that number, the primality test is repeated, and so on.

## A Exercises

Is it clear that systems that are based on the factorization assumption are still secure in this case?

(If someone wants to make this more exactly, (s)he should use an assumption of Cramer which says  $\pi(x + \log^2 x) - \pi(x) > 0$ . Thereby  $\pi(y)$  stands for the number of prime numbers smaller than or equal to  $y$ .)

- b) Someone programs – because of the procedure mentioned in a) – prime number generation accidentally in this way: after having found a prime number  $p$ , the search for the second prime number  $q$  goes on by adding 2 to the first prime number. How will you then factorize the product  $n$ ?
- c) Calculate using Square and Multiply:  $3^{19} \bmod 101$ . (Sometimes it is useful that  $100 \equiv -1$ .)
- d) Calculate  $3^{123} \bmod 77$  using Fermat's little theorem.
- e) Calculate the inverse of  $24 \bmod 101$  using the extended Euclidean algorithm.
- f) What about the inverse of  $21 \bmod 77$ .
- g) Calculate  $y \bmod 77$  with  $y \equiv 2 \bmod 7 \wedge y \equiv 3 \bmod 11$ , and  $z$  with  $z \equiv 6 \bmod 7 \wedge z \equiv 10 \bmod 11$ .
- h) Test if the following residue classes are quadratic residues (Calculating with negative representatives of residue classes simplifies your work):  $13 \bmod 19, 2 \bmod 23, -1 \bmod 89$ . If yes and if the simple algorithm from the script is applicable, extract the root.
- i) Extract all 4 roots from  $58 \bmod 77$ . (You know the secret  $77 = 7 * 11$ .)
- j) Utilize that  $2035^2 \equiv 4 \bmod 10379$  to factorize 10379.  
(For enthusiasts: How to generate such an exercise by hand?)
- k) Let be  $n = p * q$ , where  $p$  and  $q$  are prime numbers,  $p \neq q$  and  $p \equiv q \equiv 3 \bmod 4$ , and let  $x$  be a quadratic residue modulo  $n$ . Show that exactly one of the 4 roots of  $x$  modulo  $n$  is again a quadratic residue.
- l) Show: If the factorization assumption would be false then the quadratic residuosity assumption would be false, too.

[Solution at page 488.](#)

### 3-14 Which attacks on which cryptosystems are to be expected?

- a) A notary attaches a timestamp to arbitrary documents that are presented to him and signs the resulting document digitally in order to enable everybody to check the notarization of each document. Which kind of cryptographic system is he supposed to use, which kinds of attacks is the cryptographic system used supposed to resist?

- b) A newspaper receives articles by its correspondents confidentially. Which cryptographic system can be used, which attacks must be resisted by each system? (We do ignore that in practical life articles should be authentic, too.)

[Solution at page 492.](#)

**3-15 How long must files be protected? Dimensioning of cryptographic systems; reactions to false dimensioning; Publishing public keys of a signature system or of an asymmetric encryption system?**

- a) Which relation between the time frame within which data is supposed to be protected by cryptographic systems and the dimensioning of those cryptographic systems does exist? What does that mean for encryption systems for personal medical data? And what does that mean for digital signature systems if signatures are supposed to stay valid for a life time?
- b) What must be done in case of encryption systems, what in case of digital signature systems when it is conceivable that the system will be broken soon. (Hint: Do not forget to specify how to proceed with too weakly encrypted resp. signed data.)
- c) We suppose that the authentic publishing of public keys is costly, cp. for instance exercise 3-6c), so that you can only afford to publish either a public key of a digital signature system or a public key of an asymmetric encryption system. Which one will you publish and why?
- d) Is your solution for c), which you probably find under the aspect of functionality of cryptographic protocols, also right in light of b)'s solution?

[Solution at page 492.](#)

**3-16 crypto-policy by choice of a convenient length of key?**

Often a convenient, fixed key length is proposed to simplify fighting crime (espionage too, but there it is not talked about that publicly), but also to enable citizens and companies to achieve “sufficient” protection of confidentiality of data by encryption systems for companies and citizens: As long as curious neighbors and competitive companies cannot afford the effort for breaking. But the state that has more financial power and more computing capabilities can afford this. Do you think that this proceeding is reasonable or hopeless in principle? Please give reasons for your answer. (As background information: States such as the USA and Germany surely have no legal restrictions for production and national sales of cryptographic products, but put their export under the provision of authorization. USA used to grant export licenses only if the key length of symmetric encryption systems was 40 bits or less. Even nowadays certain high-performance computers underlie export restrictions. For this purpose nothing is officially public in Germany — but the proceeding is probably the same.)

[Solution at page 496.](#)

## A Exercises

### 3-17 Authentication and encryption – in which order?

Confidentiality and integrity must be guaranteed for many applications. In which order should encryption and authentication take place? Does it make a difference if symmetric or asymmetric cryptosystems are used? (It is assumed for the whole exercise that encryption and authentication run in the same devices. Thus both are implemented equally trustworthily from the user's point of view. Additionally, it is possible to configure the user interface independently from the order, for instance plaintext is always shown in case of authentication.)

[Solution at page 497.](#)

### 3-18 $s^2 - \text{mod} - n - \text{generator}$

- a) Calculate a small example for the  $s^2 - \text{mod} - n - \text{generator}$  used as a symmetric encryption system. Suggestion: The key be  $n = 77$  and the seed  $s = 2$ , the plaintext be  $0101_2$ .
- b) Decrypt the message  $1001_2, s_4 = 25$ , which was encrypted with the  $s^2 - \text{mod} - n - \text{generator}$  as an asymmetric encryption system. Your secret decryption key is  $p = 3, q = 11$  (Only use algorithms for decryption that are still efficiently executable for  $|p| = |q| \geq 300$ . Simply trying out all possible seeds for instance, only shows that you understood how it is possible to *break* the  $s^2 - \text{mod} - n - \text{generator}$  if a too small key length is chosen. This is also valid for passing on to exponentiate with 2 and observing how the cycle is closing. You are supposed to show how *decryption* works!)
- c) What happens when one uses  $s^2 - \text{mod} - n - \text{generator}$  as a symmetric encryption system and sends the ciphertext across a wire that is not fault-tolerant, and a bit completely switches through physical error? And what will happen if a bit gets lost?
- d) Are you able to construct a symmetric authentication system using the  $s^2 - \text{mod} - n - \text{generator}$  that needs less key exchange than a real authentication code? If applicable, how does it work? How secure is your system?
- e) How many key exchanges are necessary if one wants to send several messages consecutively using the  $s^2 - \text{mod} - n - \text{generator}$  as a symmetric encryption system? What must be stored by everyone?
- f) *Optional:* If the  $s^2 - \text{mod} - n - \text{generator}$  reaches the inner state  $s_i = 1$  sometime then all next states are = 1 because  $1^2 = 1$ . Thus from there all bits are 1. It is clear that this is allowed to happen only very rarely otherwise the  $s^2 - \text{mod} - n - \text{generator}$  would be cryptographically insecure. Calculate the probability that the  $s^2 - \text{mod} - n - \text{generator}$  reaches the inner state 1 after  $i$  steps. (Tip: There is hardly anything to calculate, you just have to find a simple argument.)

[Solution at page 497.](#)

**3-19 GMR**

- a) Let your secret key be  $p = 11, q = 7$ . Sign the message  $m = 01$  with the reference  $R = 28$ . (The length of the message be always 2 bit.) Is this possible with  $R = 28$ ? If no, take  $R = 17$  instead.
- b) Assume that altogether 8 messages are to be signed. Which parts of the reference tree and appropriate signatures is the signer supposed to newly create when reaching the 4th message? And at the 5th? Which parts will be sent, what is the receiver supposed to test? (It would be best to paint a picture and mark.)
- c) Sketch a proof for the collision resistance of the family of  $f_{pref(m)}$  mentioned in §3.5.3. Where is the prefix-free coding needed in your proof?

[Solution at page 499.](#)

**3-20 RSA**

- a) Calculate a small example for RSA as an encryption system.
- b) Construct a small example for RSA as a signature system.
- c) How will the plaintexts 0 and 1 be encrypted, and how will the ciphertexts 0 and 1 be decrypted? What do we learn?
- d) How would you make an indeterministically encrypting encryption system from RSA? How many random bits do you need? Why are, for instance, 8 random bits not enough? Why is it not a good idea to use time as random bits?
- e) *Optional:* Can you see if your example from d) is secure against active attacks, cp. §3.6.3? What must be done against active attacks, if applicable? What would the message format then look like?
- f) *Optional:* How would you create an indeterministically signing signature system from RSA? What would that be useful for? Or could this even cause problems?
- g) How does the computation effort per encrypted bit in dependence of the security parameter  $l$  grow if the algorithms shown in §3.4.1.1 are used for implementation? Distinguish as borderline cases if extremely short or extremely long messages are involved.
- h) Sketch why the secret operation of RSA, thus signing resp. decrypting, is 4 times faster when one calculates modulo  $p$  and  $q$  separately, such when calculating modulo  $n$ .
- i) By which factor is the public operation of RSA faster if one uses  $2^{16} + 1$  as an exponent instead of a random number?

## A Exercises

- j) What do you think about the following change of key generation of RSA (taken from [MeOV\_97, page 287]): instead of  $\phi(n) = (p - 1)(q - 1)$  the least common multiple of  $(p-1)(q-1)$ , i.e.,  $\text{lcm}(p - 1, q - 1)$ ? Does this change anything in terms of the security of RSA? What is changing in the computation effort of RSA?
- k) *Optional:* Calculate a small example for the attack of Johan Håstad.
- l) You can find a proof, for instance in [MeOV\_97, page 287], that breaking RSA completely, i.e., finding the secret key  $d$ , is equivalent to factorizing the modulus. Why does this say unfortunately say little about the security of RSA?

[Solution at page 504.](#)

## 3-21 DES

- a) A simple DES-like cryptosystem is given by: block length 6; 2 iteration rounds; no initial permutation;  $f$  be described by  $f(R, K_i) = R * K_i \bmod 8$ ;  $K$  be  $(K_1, K_2)$ .  
The key be 011010. Encrypt the plaintext 000101 and decrypt it again.
- b) How would you implement S-Boxes in software to be able to decrypt as fast as possible?
- c) *Optional:* And how would you implement the 32-bit-permutations?
- d) known-plaintext attack: If you know some plaintext-ciphertext-pairs that were encrypted using DES, how and at what cost (according to number of en- and decryptions of DES) could you figure out the key?
- e) chosen-plaintext attack: Does something change in comparison d) if you are allowed to choose an additional plaintext block and receive the appropriate cipher block? How high is the cost now?
- f) How can you break DES if DES contains no substitutions but only permutations and additions modulo 2? Contemplate if your attack would work more generally for each linear block cipher?

[Solution at page 509.](#)

## 3-22 Modes

- a) Confidentiality and authentication is reachable when using CBC: the first by transmitting plaintext blocks exactly, the latter by transmitting the plaintext and the last ciphertext block only. One could think both is possible at the same time by simply transmitting the ciphertext block: On the one hand secrecy would be secured, but on the other hand the receiver would be able to compute the plaintext and thus he would have all pieces of information for authentication at his disposal. Why does that not suffice in principle?

- b) Show with a simple attack that the scheme is still insecure when one attaches a block full of zeros to the plaintext before it is encrypted to receive at least a minimum of redundancy.
- c) Contemplate the mode PCBC in §3.8.2.5. Show that your attack will not work here (when one proceeds in the same way, thus attaching a block full of zeros, then transmitting the ciphertext and wanting secrecy as well as authentication)
- d) Determine for the modes described in §3.8.2 what an attacker is able to achieve when he has possibilities for *adaptive active attacks*. Assume that he is not able to break the used block cipher. (It is assumed as usual that the attacker knows the cryptosystems used. In particular he knows the length of the used cryptosystems and the mode used.)
- e) You want to work with ECB, CBC (may it be for secrecy, may it be for authentication) or PCBC, but you do not know if you will receive plaintexts of appropriate length (i.e., length of plaintext is divisible by the block length without a remainder). The advice “Then we fill with zeros up to the block length” given often is not satisfying for you, because it is supposed to make a difference if one receives “I owe you € 10” or “I owe you € 100000”. Design a coding
  - 1) that maps plaintexts of arbitrary length to such an appropriate length,
  - 2) that is uniquely invertible,
  - 3) whose expansion of length is minimal.
- f) OFB allows neither random access nor ability of parallelizing because the state of the shift register must be computed from scratch each time. Design an obvious modification for OFB that allows random access and parallelizing.

[Solution at page 511.](#)

### 3-23 Mirror attack

We assume that we want to substitute passwords for LOGIN with something better, but we have stupid terminals and an insecure network between terminals and mainframe. Therefore each participant has a “pocket calculator” which implements a secure symmetric block cipher (cp. footnote at §2.2.1) though it does not have an electrical access to the terminal. Therefore all information transfer via Display and keyboard has to run over the participant. What will you do if you assume that the mainframe itself is secure? Which attacks will not be tolerated by your designed system?

[Solution at page 514.](#)

### 3-24 End-to-end encryption

Which problems do arise if one wants to carry out an end-to-end encryption using an office computer, thus a computer to which others also have access? Which approaches do you see for a solution?

[Solution at page 516.](#)

## A Exercises

### 3-25 End-to-end encryption without exchanging keys before

Bob and Alice, our couple of cryptographers, are – as always – separated and this time Alice is traveling by air.

- a) And once again the luggage got lost, thus key and crypto devices are not available. How can the two communicate with some measure of confidentiality and authenticity? (A tip: You have to assume that the attacker can not be everywhere at all times.)
- b) The baggage was recovered and returned. But someone wanted to find out the keys in the crypto devices, thus they were erased automatically. How could Alice and Bob improve their communication now?

[Solution at page 516.](#)

### 3-26 Identification and authentication by an asymmetric encryption system

How can a participant  $P$ , whose public key  $c_P$  of an asymmetric encryption system is known by a participant  $U$ , be identified by  $U$ , cp. §2.2.1?

- a) Design a simple cryptographic protocol. (The existence of such a protocol is amazing since asymmetric encryption systems were only defined with the aim *confidentiality*. But actually the protocol that is searched for exists.)
- b) Against which attacks is the encryption system in your small cryptographic protocol supposed to be secure?
- c) Are you able to improve your small cryptographic protocol in a way that it identifies not only  $T$  but also authenticates messages  $N_i$ ? (As incentive: Such a protocol exists.) Is your authentication “system” asymmetric or symmetric, i.e., does it comply with digital signatures? Where is the disadvantage to “normal” authentication systems?
- d) Is the asymmetric encryption system using your improved protocol supposed to resist such potent active attacks as in b)?

[Solution at page 517.](#)

### 3-27 Secrecy by symmetric authentication systems

How can a participant  $A$  who has only a symmetric authentication system in common with a participant  $B$  communicate with him secretly? Please do not just take the confidentially exchanged key  $k_{AB}$  for a symmetric encryption system, it is actually just a matter of an authentication system. (The existence of such a protocol is amazing since symmetric authentication systems were only defined with the aim *integrity*. But actually the protocol that is searched for exists.)

See p. #369

[Solution at page 519.](#)

### 3-28 Diffie–Hellman key exchange

- a) *Main idea:* What is the main idea of the Diffie–Hellman key exchange?
- b) *Anonymity:* Diffie–Hellman key exchange, as it was described in §3.1.1.1, distinguishes itself, according to §3.9.1, from asymmetric encryption systems in the ability of the sender to stay anonymously (or not) towards the receiver of the encrypted message.

In case of asymmetric encryption systems this is the trivial case: The sender obtains the public key for itself, encrypts the message using this key, sends it to the receiver. The receiver decrypts the message independently from the one who had encrypted it.

This is different in case of Diffie–Hellman key exchange. The secret key which is assigned to the relationship of sender and receiver is used for symmetric en- and decryption. Thus it seems that no anonymity is possible for sender to receiver.

Consider a modification of the Diffie–Hellman key exchange where anonymity for the sender to the receiver is possible. And what must be done if the sender wants to receive an answer anonymously?

- c) *Individual choice of parameters:* Diffie–Hellman key exchange, as described in §3.9.1, distinguishes from asymmetric encryption systems according to §3.1.1.1 in the property that parameters that are critical for security (concretely  $p$  and  $g$ ) are known in general and are therefore equal for all participants. The question of who chooses them and if that person is able to get special power arises instantly. Because it is not manageable if one chooses  $p$  and  $g$  randomly or somehow specially that extracting the discrete logarithm is particularly easy for one. If one lets a external instance choose  $p$  and  $g$  one needs a stronger security assumption than the discrete logarithm assumption. Alternatively many instances can choose  $p$  and  $g$  together, but this requires a cryptographic protocol again.

Consider in the scope of this exercise if you are able to modify the Diffie–Hellman key exchange in a way that each participant is able to choose all parameters that are critical for security.

- d) DSA (DSS) as a basis: On May 19th, 1994 a standard for digital signatures within all and with all public places in the USA was set once and for all by the National Institute of Standards and Technology (NIST), which is the successor of the National Bureau of Standards that standardized DES in 1977: the **Digital Signature Standard (DSS)**, given by the **Digital Signature Algorithm (DSA)** [Schn\_96, page 483-495]. The steps in the case of key generation, - certification and -publishing that are interesting for us are:

- 1) a huge prime number  $p$  and a randomly chosen number  $g \in \mathbb{Z}_l^*$  are public (and possibly equal for all participants)
- 2) Each participant chooses a number  $x$  with a length of app. 159 bit and keeps it secretly as a part of his signature key.

## A Exercises

- 3) Each participant computes  $g^x \bmod p$  which will be certified and published as a part of his test key.

Is this enough to perform Diffie–Hellman key exchange?

[Solution at page 520.](#)

### 3-29 Blind Signatures with RSA

Calculate a small example for blindly achieved signatures with RSA. You can save calculating effort by using results of exercise [3-20b](#)).

[Solution at page 521.](#)

### 3-30 Threshold scheme

Analyze the secret  $G = 13$  in 5 parts that  $k = 3$  are necessary to reconstruct it. Use the smallest prime number  $p$  that is possible and as random coefficients  $a_1 = 10$ , and  $a_2 = 2$ . (In practice it is not allowed to choose  $p$  in such strong dependence to  $G$ . This choice of  $G$  is supposed to ease your calculating and to show that you understood the conditions for  $p$  in relation to  $G$ .)

Reconstruct the secret by use of the parts  $(1, q(1)), (3, q(3))$  and  $(5, q(5))$ .

[Solution at page 522.](#)

### 3-31 Provably secure usage of several encryption systems?

Now that you know that there are informational secure and efficient threshold schemes, it is worth paying attention to exercise [3-5](#) again. How do you combine multiple encryption systems that the resulting total system is at least provable as secure as the most secure used encryption system? (Thereby you are allowed to suppose that the used encryption systems are designed for uniform distributed plaintexts – as they arise in case of minimal length coding, and that “Security of encryption system” means security in case of uniform distributed random plaintexts. Likewise you are allowed to suppose that the parts that are generated by the information-theoretically secure and efficient threshold scheme are uniformly distributed random plaintexts.)

Compare the effort of the provable secure combination with the one described in exercise [3-5](#).

Compare each security achieved by the combination.

Does an especially simple implementation of a threshold scheme come into your mind for exactly this application?

[Solution at page 523.](#)

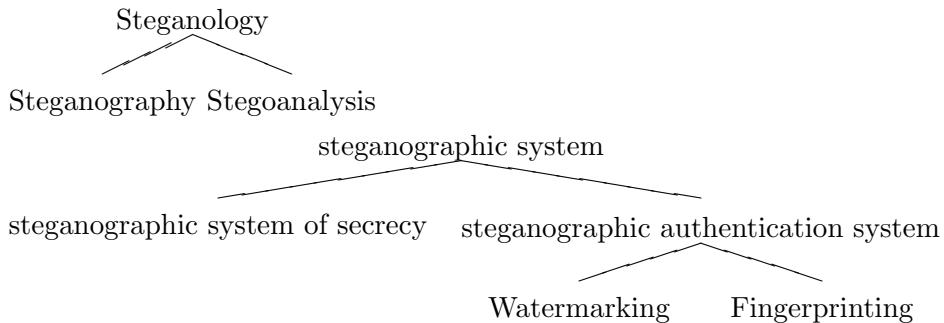
## 4 Exercises for “Basics of Steganography”

### 4-1 Terms

In the following picture, trees of terms are shown that were similarly created to the trees shown in the section about cryptography, cp. exercise [3-1](#). The term *Steganology* is

newly created and in need of further exposure. I thought about taking the shorter term *Stegology* but decided to use the longer but grammatical correct term. Some authors use the term *information hiding* for the area of Steganology and therefore especially for the area of Steganography. They divide information hiding into two parts: *steganography* (that which I call steganographic secrecy) and *copyright marking* (that which I call steganographic authentication). Similar to referring to *cryptoanalysis* as *cryptanalysis* in German, the shorter term *steganalysis* is used for *stegoanalysis*.

Describe the textual structure of the trees of terms and the advantages and disadvantages of using the shorter term *stegosystem* in general for steganographic systems or especially for steganographic systems of secrecy.



[Solution at page 524.](#)

#### 4-2 Properties of in- and output of cryptographic and steganographic systems

- Draw a scheme (block diagram) which contains general parameters (all input and output values) of an encryption system. Where are the differences between symmetric and asymmetric encryption systems? Which assumptions about the inputs is a good cryptosystem allowed to make, which guidelines is it supposed to fulfill regarding to its outputs?
- Draw a scheme (block diagram) which contains general parameters (all input and output values) of a steganographic system of secrecy. Where are the differences between symmetric and asymmetric encryption systems? Which assumptions about the inputs is a good stegosystem allowed to make, which guidelines is it supposed to fulfill in regards to its outputs?
- What are the main differences between cryptographic and steganographic systems of secrecy?

[Solution at page 524.](#)

#### 4-3 Is encryption superfluous when good steganography is used?

Why is it actually unnecessary to encrypt the secret message before it is embedded when a secure steganographic system is used? And why would you do it in practice despite this?

[Solution at page 527.](#)

## A Exercises

### 4-4 Examples for steganographic systems

Name some examples for steganographic systems. How do you evaluate the security of your enumerated systems?

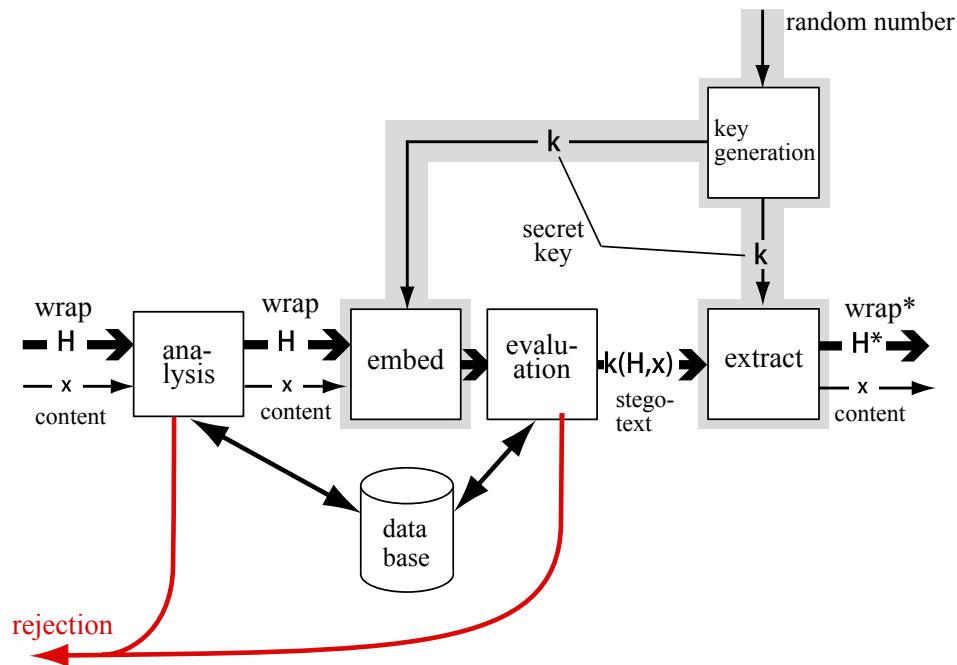
[Solution at page 527.](#)

### 4-5 Modeling steganographic systems

A group of students discusses the modeling of steganographic systems shown in the Figures 4.2, 4.3, 4.4, and 4.5: Substantial things were forgotten, says *Anja Foresight*. We need an analysis before embedding which is supposed to cooperate with a database of all so far used wrappers. This is the only way to avoid that improper wraps are used for embedding.

Right, agrees *Berta Viewback*. But it would be much better to do the analysis after embedding – let us call this algorithm **Evaluation**. The task of evaluation is to check if the stegotext is inconspicuous. If yes, the stegotext is outputted. If no, it is rejected.

Do not argue, notices *Caecilie Baseman*. We just do the analysis before as well as the evaluation after embedding. This is the general case and we do agree, right? Secure is secure! Let me plot it for you:



*Doerthe Reinhardt* raises her eyebrows and contradicts: No, I do not like any of your improvement suggestions. The original model is sufficient.

Who do you agree with – and above all: How do you justify your choice?

[Solution at page 527.](#)

#### 4-6 Steganographic secrecy with public keys?

Ross Anderson describes in [Ande4\_96, AnPe\_98, page 480] the following procedure for steganographic secrecy:

Given is a wrap in which absolutely any key can be embedded. Then usually a rate exists at which bits can be embedded without the stegoanalyzer noticing it. ... Alice knows Bob's public cipher key. She can take her message which is to be kept in secret, encrypt it with Bob's public key, and then embed the resulting ciphertext. Each possible receiver will then simply try to decrypt the extracted message, but only Bob will succeed. In practice the value encrypted with the public key could be a control block consisting of a session key and some filling material. The session key would be used for operating a conventional steganographic system.

Ross Anderson calls this public key steganography. Do you agree with this term? Please give reasons for your decision.

[Solution at page 528.](#)

#### 4-7 Enhancement of security by using several steganographic systems

What can you do if you have several steganographic systems, each underlying a different security assumption, and you do not want to trust one of these steganographic systems solely (resp. one of the security assumptions)? Differentiate between the protection goals secrecy (i.e., you have several encryption systems) and authentication (i.e., you have several authentication systems).

[Solution at page 528.](#)

#### 4-8 Crypto- and steganographic secrecy combined?

Steganographic secrecy overlaps the functionality of cryptographic secrecy, so that, if steganographic secrecy can be assumed as secure, no combination with cryptographic secrecy is useful. Is a combination useful and how should it happen if applicable when doubts exist concerning the security of the steganographic secrecy?

[Solution at page 529.](#)

#### 4-9 Crypto- and steganographic authentication in which order?

Steganographic authentication does not protect the plaintext from unrecognized small altering, a digital signature is doing this but can be easily removed by anyone. Can you combine both? If applicable, in which way?

[Solution at page 529.](#)

#### 4-10 Stego-capacity of compressed and non-compressed signals

Is it possible to embed in terms of percent more undetected in compressed or in non-compressed signals? Please differentiate between lossy and lossless compression and give reasons for your answer.

[Solution at page 529.](#)

## 5 Exercises for chapter 5 “Security in communication networks”

### 5-1 Quantification of unobservability, anonymity, and unlinkability

In §5.1.2 definitions for unobservability, anonymity, and unlinkability are given and it is emphasized that their reliability depends on the underlying attacker model and where appropriate, on a classification. It is defined that unobservability, anonymity, as well as unlinkability are perfect if the relevant probability is the same for the attacker before and after his observations. Try to quantify unobservability, anonymity, and unlinkability, i.e., find useful a) parameters resp. b) gradations between each extreme:

- the attacker finds out nothing – the attacker finds out everything, and
- the attacker knows nothing – the attacker knows everything

[Solution at page 529.](#)

### 5-2 End-to-End or link encryption

Due to disclosures about plentiful wiretapping by secret services (e.g., former GDR secret service Stasi) the management of a company decides to encrypt all future telecommunication between the offices. How can and should this happen? Are your measures dealing rather with link encryption than with end-to-end encryption? Or do you suggest to even use both?

[Solution at page 529.](#)

### 5-3 First encrypting and then encoding fault-tolerance or the other way round?

In many communication networks error-detecting and even error-correcting codes are used that are adapted to the dumb error behavior of the transmission channels. According to the channel one has to reckon that zeros become ones or ones become zeros, that errors arise separately or in bursts, etc. Should one encrypt first and then encode fault-tolerant in case of networks with adapted fault-tolerance encoding, or should one prefer the reverse order? Explain your answer.

[Solution at page 530.](#)

### 5-4 Treatment of address- and error-recognizing fields in case of encryption

Given a plaintext message 00110110, which is supposed to be sent to the address 0111 (in this exercise each coding given is binary). The usual message format consists of a 4-bit-address, an 8-bit-message and a 2-bit-check-character for detecting transmission errors in the address or the message.

Address	Message	Check-char.

The 2-bit-check-character is created by two-bit-wise addition modulo 4 of address and message. For the above address and plaintext-message the check character would result in 10, thus altogether the message 01110011011010.

- a) Let the message above be encrypted end-to-end with a one-time pad. Let the one-time pad be 011001111110001101010001101.... What does the end-to-end encrypted message look like if it is encrypted using one-time pad modulo 2?
- b) Let the message above be connection encrypted using one-time pad. What does the encrypted message look like if it is encrypted using one-time pad modulo 2?
- c) Let the message above be encrypted end-to-end and connection encrypted with the above one-time pad. What does the encrypted message look like if it is encrypted using one-time pad modulo 2?

[Solution at page 530.](#)

### **5-5 Encryption in case of connection-oriented and connection-less communication services**

In the case of communication services there is a distinction made between *connection-oriented* and *connection-less* communication services. The first guarantee that messages are received in the same order as they were sent, the latter do not guarantee this. Do you have to consider this when choosing an encryption algorithm and its mode? If yes, what do you have to pay attention to?

[Solution at page 530.](#)

### **5-6 Requesting and Overlaying**

- a) Compute a small example for the method “Requesting and Overlaying”.
- b) Why must there be encryption between participants and servers? Must the answer of the server to the participant also be encrypted?
- c) How do you implement “Requesting and Overlaying” by use of usual cryptographic means if the transmission bandwidth from participant to server is very narrow? And what if the transmission bandwidth from server to participant, but not among the servers, is very narrow? Can you combine both solutions skillfully?
- d) What do you think about the following suggestion: To confuse the servers the participant generates randomly an additional request-vector, sends it to an additional server, receives its answer and ignores this. None of the used  $s + 1$  servers know if it is used as a normal or as an additional, ignored server.

[Solution at page 530.](#)

## A Exercises

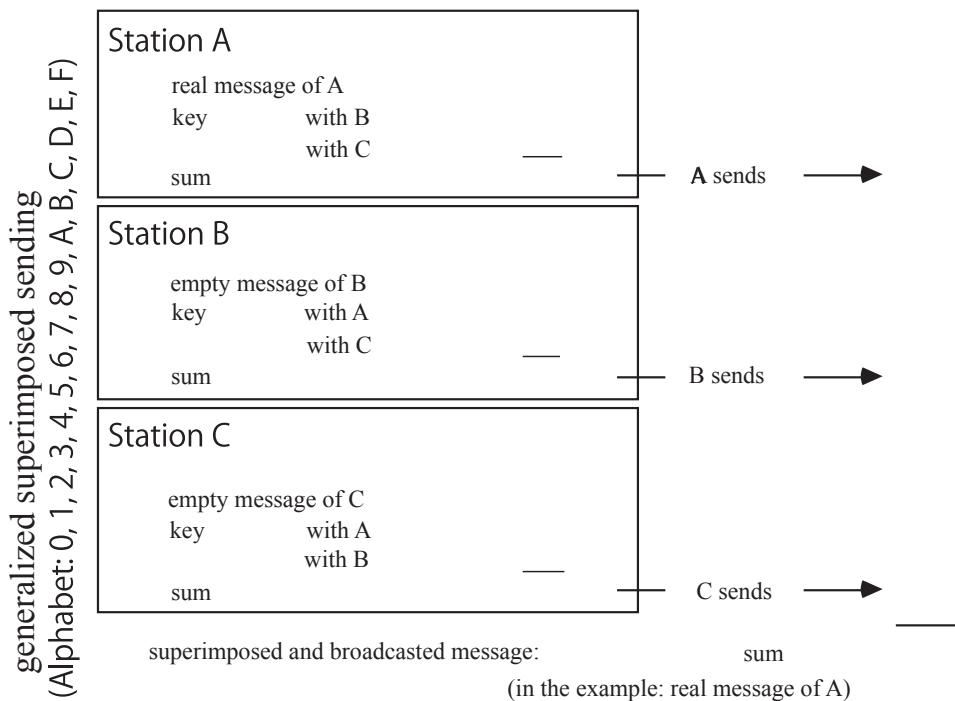
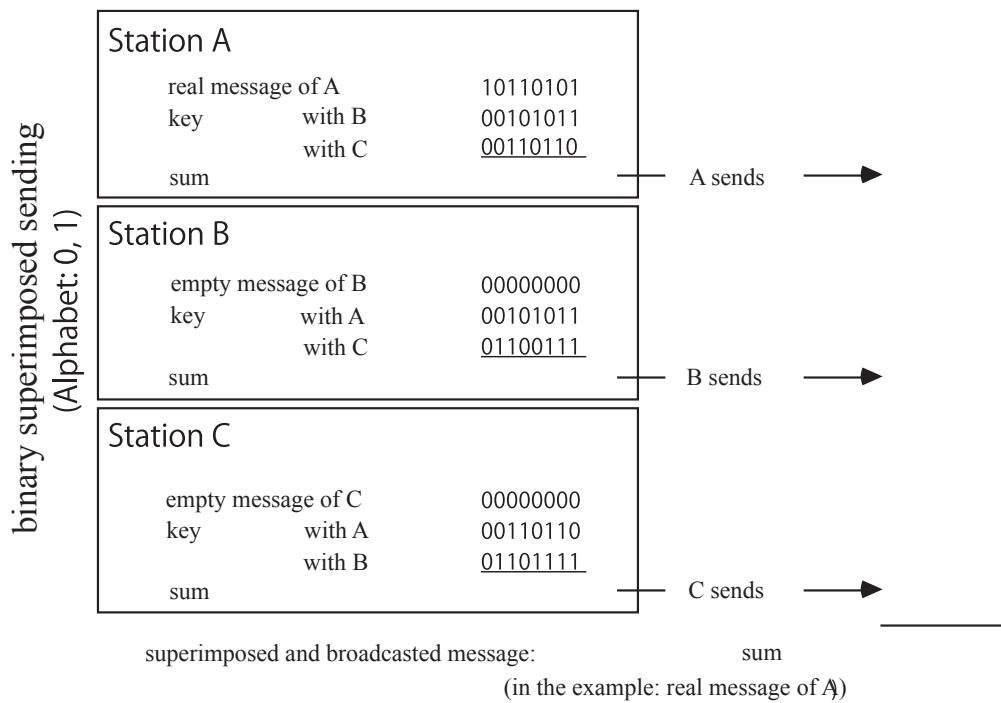
### 5-7 Comparison of “Distribution” and “Requesting and Overlaying”

- a) Compare “Distribution” and “Requesting and Overlaying”. Consider especially how opened implicit addresses can be implemented as efficiently as possible in case of “Distribution” and “Requesting and Overlaying”.
  
- b) Can you imagine communication services where “Requesting and Overlaying” are necessary because “Distribution” is not applicable?

[Solution at page 530.](#)

### 5-8 Overlayed Sending: an example

Given the example for overlayed sending shown in the following figure. Inscribe the missing values in the upper part of the figure in which it is calculated modulo 2. After that, calculate the same example for modulo 16 in the bottom part of the figure. Which values would you be allowed/supposed to copy from the upper part of the figure? Why?



Solution at page 530.

## A Exercises

### 5-9 Overlayed sending: Determining a fitting key combination for a alternative message combination

How must the keys look like in the example of the previous exercise to have the station A have sent the empty message 00000000 and station B have sent the real message 01000101 in the case of same local sum and output (and of course therewith with same global sum)? Which real message must station C have sent then? Is the key combination unique? Will it be if it is given additionally that the key between A and B stays the same because the attacker knows it? (A has given the key to B on a floppy, which B, after having copied its content to a disk, has not physically destroyed or at least not written on multiple times but he has deleted it, i.e., the operating system has signed the key as deleted. After that B has recorded the almost new floppy with very few data and gave it to a friendly man, which acted unexpectedly as an attacker.) What are the other keys in this case?

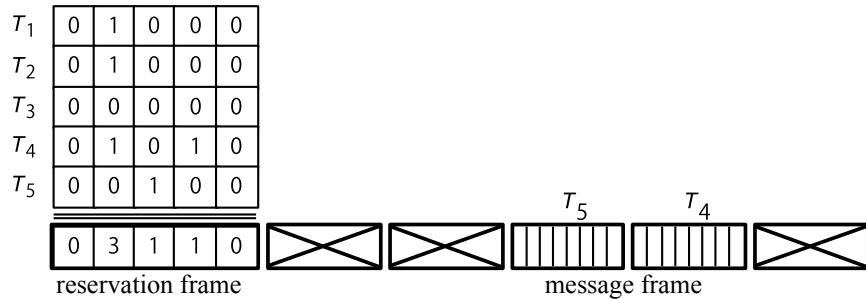
Hint: If you calculate this exercise only in regards to addition modulo 2 you should differentiate between + and - too. This of course only plays an important role with bigger modulus, thus in the exercise only with modulus 16.

[Solution at page 530.](#)

### 5-10 Several-access-methods for additive channels, e. g., overlayed sending

- a) Reservation method

It is shown in the following figure how reservation in a message frame avoids collisions of message frames if after reservation message frames are only used by such stations that reservation resulted in 1 as sum. What would happen in this example if the reservation frames did not overlay modulo a greater number but modulo 2?



- b) pairwise overlayed receiving

In a DC-net in which pairwise overlayed receiving is not supported, two friends, Scrimpy and Scrappy, are agreeing to literally double the bandwidth of their communication by pairwise overlayed receiving. What do they have to find out first? Perhaps the following example will help you: Scrimpy has sent 01101100 and receives 10100001 as sum. He asks himself: What has Scrappy sent to me? What is

the answer if the overlayed sending takes place modulo 2, as with modulo 256? (We assume that the several-access-method does not make difficulties for Scrimpy and Scrappy. For instance one may allow that one of both reserves a (simplex-)channel which both are using for pairwise overlayed sending.)

- c) Global overlayed receiving: collision detection algorithm with creating the average value

The participant stations 1 to 5 send messages 5, 13, 9, 11, 3. How does the collision detection with creating the average value and overlayed receiving take place? Let the local decision criterion be  $informationunit \leq |\emptyset|$ . Use the notation given in Figure 5.14 of the script.

How does it happen if the participant stations send 5, 13, 11, 11, 3?

[Solution at page 530.](#)

### 5-11 Key-, overlaying-, and transmission topology in the case of overlayed sending

- Why is it possible to specify the key topology independently from overlaying- and transmission topology?
- Why should overlaying- and transmission topology be adjusted to each other?

[Solution at page 530.](#)

### 5-12 Coordination of overlaying- and transmission alphabets in case of overlayed sending

With which “trick” was it possible to achieve a huge overlaying alphabet (useful, for example, for collision detection algorithm with average value creation) and a minimal delay of time, i.e., a minimal transmission alphabet, at the same time?

[Solution at page 530.](#)

### 5-13 Comparison of “Unobservability of adjacent wires and stations as well as digital signal regeneration” and “overlayed sending”

- Compare the two concepts for sender anonymity “Unobservability of adjacent wires and stations as well as digital signal regeneration” and “overlayed sending” in regards to the strength of considered attackers, effort, flexibility. Which concept is the most elegant one? Why?
- Do these concepts cover receiver’s anonymity? If applicable, what does the cooperation of sender and receiver anonymity look like?

[Solution at page 530.](#)

## A Exercises

### 5-14 Comparison of “overlaid sending” and “MIXes that change the encoding”

Compare the concepts “overlaid sending” and “MIXes that change the encoding” in regards to a security aim, strength of considered attackers, effort, flexibility. Which of these is the more practicable concept in your opinion? In which respect and why?

[Solution at page 531.](#)

### 5-15 Order of basic function of MIXes

To what extent is the order of basic functions of a MIX in Figure 5.22 inevitably?

[Solution at page 531.](#)

### 5-16 Does the condition suffice that output messages of MIXes must be the same lengths?

What do you think about the following suggestion (according to the maxims of the retired chancellor Kohl: The deciding factor is the result): MIXes can also mix input messages of different length together. The main point, is that all appropriate output messages have the same length.

[Solution at page 531.](#)

### 5-17 No random numbers → MIXes are bridgeable

Explain how an attacker could bridge a MIX if the direct scheme of changing the encoding to guarantee sender’s anonymity (cp. §5.4.6.2) is used without random numbers.

[Solution at page 531.](#)

### 5-18 Individual choice of a symmetric encryption system at MIXes?

MIXes publish their public cipher keys and therefore define the appropriate asymmetric encryption system. In §5.4.6.1 (symmetric secrecy at the first and/or at the last MIX) and in §5.4.6.3 (indirect scheme of changing the encoding) two applications of symmetric secrecy in the case of MIXes are introduced. Should each user of the MIX-network be able to choose the used symmetric encryption system or should it be determined by the MIX?

[Solution at page 531.](#)

### 5-19 Less memory effort for the generator of untraceable return addresses

It is described in §5.4.6.3 how untraceable return addresses can be created. Here it is costly for the generator that it has to memorize all symmetric keys under the unique name  $e$  of the return address until the return address is used. This can last a while and is therefore circumstantial. Consider how the generator of return addresses could avoid memorizing symmetric keys and even unique names.

[Solution at page 531.](#)

### 5-20 Why changing the encoding without changing the length of a message

Explain how an attacker could bridge the MIXes shown in the upper part of Figure 5.26.  
(It is trivial, but just try to explain it precisely).

[Solution at page 531.](#)

### 5-21 Minimal expanding of length in a scheme for changing the encoding without changing length of a message for MIXes

- a) Describe shortly the effort for communication and encryption of MIX nets.
- b) You have an asymmetric encryption system, which is secure against adaptive active attacks, and an efficient symmetric encryption system at your disposal. Both are working (after having chosen keys) on blocks of the same length. Design a scheme for changing the encoding for MIXes which expands the message minimally at the sender and which does not change the length of the message during the steps of changing the encoding.

Derive for simplicity this minimal expanding scheme for changing the encoding from symmetric encryption system with the properties  $k^{-1}(k(x)) = x$  and  $k(k^{-1}(x)) = x$ , i.e., not only en- and decryption but also de- and encryption are inverse towards each other, given in § 5.4.6.5, cp. Figure 5.31.

[Solution at page 531.](#)

### 5-22 Breaking the direct RSA-implementation of MIXes

What is the main idea of the described attack on the direct RSA-implementation of MIXes in § 5.4.6.8?

[Solution at page 531.](#)

### 5-23 Use of MIX-channels?

- a) Why and for what are MIX-channels necessary?
- b) What is limiting their use resp. is leading to the “development” of time-slice-channels?

[Solution at page 531.](#)

### 5-24 Free order of MIXes in case of fixed number of used MIXes?

Ronny Freeroute and Andi Cascadius are arguing again if free choice of MIXes and their order should be preferred- or so-called MIX-cascades should be used instead, i.e., MIXes and their order are firmly stated. Against the usual scheme of discussion

*Ronny Freeroute:* Free choice of the MIX-order would be better because everyone is free in his choice that way. (Confidence, usage of MIXes on the way) and, if only a few MIXes attack, greater groups of anonymity would come up.

## A Exercises

*Andi Cascadius:* MIX cascades must be preferred, because the largest anonymity set possible would come up in case of maximal number of attacking MIXes because the intersection of anonymity sets would be empty or the intersection would contain the whole anonymity set.

a pale Ronny says someday: “What if the attacker controls a maximal number of MIXes and knows that everyone uses a fixed number of MIXes in case of free choice of the order of MIXes. Then the attacker has good a chance to bridge the MIX that is not attacking, too.” What does Ronny mean, i.e., what could a simple attack on a MIX-net with free choice of route and participants with a fixed chosen route-length look like? And what would you do against it?

[Solution at page 531.](#)

### 5-25 How to ensure that MIXes receive messages from enough different senders?

As it was explained at the beginning of §5.4.6.1 MIXes have to work on information units from sufficient senders in one batch – are they from too few the danger is high that one of them can be observed from the few others over the MIXes too.

How can the compliance of the claim “information units from sufficient senders” be ensured

- a) at the first MIX of a MIX cascade,
- b) at a later MIX of a cascade, i.e., not the first one?

Otherwise a MIX could substitute, for instance, in its output batch all information units except for one information unit produced by itself and trace the non-substituted information unit despite all following MIXes this way.

[Solution at page 531.](#)

### 5-26 Coordination protocol for MIXes: Wherfore and where?

For which schemes of encryption is a coordination protocol between MIXes necessary? Give reasons for your answer.

[Solution at page 532.](#)

### 5-27 Limitation of responsibility areas – Functionality of network terminals

Why should the necessary areas of confidence between participant and network provider overlay as little as possible? Explain this for all introduced security measures in communication networks. (Figure 5.48 might be a help for you).

[Solution at page 532.](#)

### 5-28 Multilaterally secure Web access

Consider multilaterally secure web access: Which security problems exist? Which participant has which protection interests? Then apply all procured concepts to this new problem. (Your curiosity is supposed to be created and the procured is supposed to be practiced with this larger scaled exercise. Therefore it is okay to have a look in the web for existing concepts after having analyzed the problem. And even I will allow it myself to link to literature when giving the solution.)

[Solution at page 532.](#)

### 5-29 Firewalls

Many companies and authorities control the traffic between public networks, e.g., the *Internet*, and their in-house network, e.g., LAN or *Intranet*, by so-called Firewalls (the name comes from the fire protection in buildings and means there is a wall which prevents the spreading of fire or at least delays the spreading). Do you think this is useful? What does it exactly bring? Where are the limits? Is this able to substitute admission and access controls for a workplace computer?

[Solution at page 532.](#)

## 6 Exercises for “value exchange and payment systems”

### 6-1 Electronic Banking – comfort version

- a) What do you think about the following paper-banking comfort version: The customer receives pre-filled forms from his bank in which only the amount must be filled in. It especially saves time for the customer because he does not have to sign the forms anymore.
- b) Does this paper-banking comfort version differ from the electronic banking version which is usual today?
- c) Will this be changed if customers receive a chip card from their bank with pre-loaded keys for MACs or digital signatures?

[Solution at page 532.](#)

### 6-2 Personal relation and linkability of pseudonyms

Why do role-pseudonyms offer more anonymity than personal-pseudonyms and transaction-pseudonyms more anonymity than business-relation-pseudonyms?

[Solution at page 532.](#)

## A Exercises

### 6-3 Self-authentication vs. foreign-authentication

Explain self-authentication and foreign-authentication and the difference between both. Give typical examples. Why is the distinction between self-authentication and foreign-authentication important for configuration of several-sided IT-Systems?

[Solution at page 533.](#)

### 6-4 Discussion about security properties of digital payment systems

The security properties described in §6.4 do only consider availability and integrity of payment procedures, thus only the most essential ones. Create a list of necessary protection goals that you consider as necessary. Order them by confidentiality, integrity, and availability.

[Solution at page 533.](#)

### 6-5 How far do concrete digital payment systems fulfill your protection goals?

Prove known digital payment systems against your protection goals that you elaborated in exercise 6-4. Examples could be

- tele-banking
- homebanking computer interface (HBCI); <http://www.bdb.de/>, search for HBCI

[Solution at page 534.](#)

## 9 Exercises for “Regulation of security technologies”

### 9-1 Digital signatures at “Symmetric authentication allows concealment”

Is it possible to use digital signatures instead of symmetric authentication codes in the procedure of “Symmetric authentication allows concealment” (§9.4)? If no, why not?

[Solution at page 534.](#)

### 9-2 “Symmetric authentication allows concealment” vs. Steganography

What do “Symmetric authentication allows concealment” (§9.4) and Steganography (§4) have in common? What is the difference?

[Solution at page 535.](#)

### 9-3 Crypto-regulation by prohibition of freely programmable computers?

Due to the procedures in §9.1 - §9.6 it is clear that crypto-regulation is not applicable as long as freely programmable computers are available. We assume that *all* states agree to prohibit the production of freely programmable computers and they could eventually succeed in enforcing this prohibition. We assume the decades in which freely programmable computers already sold still worked have elapsed – thus there are no freely

*9 Exercises for “Regulation of security technologies”*

programmable computers anymore. Is it then possible to enforce a crypto-regulation? Asked more exactly: Is it then possible to enforce a crypto-regulation which allows to decrypt email traffic of a suspect after a permission for observation has been given – but not before?

[Solution at page 535.](#)

*A Exercises*

# B Answers

## B.1 Answers for “Introduction”

### 1-1 Link between spatial distribution and distribution in terms of control- and implementation structure.

Spatially distributed IT-systems usually allow inputs at more than one place. Those inputs (if not ignored) have effect on the system state. If you want to implement a central control structure through an instance with a global view, every incoming input needs to be communicated to it. Only after this may the state transition take place. Since information can only be transported at the speed of light and since today's computers work at more than  $10^7$  instructions per second you have:

$$\frac{300000 \text{ km/s}}{10^7 \text{ instruction/s}} = 30 \text{ m/instruction}$$

You would use more time for transmission than for processing<sup>1</sup>. While the computation performance is steadily increasing (and the structural size is decreasing) the speed of light remains constant, so this argument becomes an issue more and more. Physical maintenance in spatial distributed IT-Systems is quite costly especially right after a failure. But IT-Systems that have a distributed control- and implementation structure allow the postponement of certain maintenance tasks (even after a serious failure) since a failure will not bring down the whole system. Further arguments can be found in the next answer.

[Exercise at page 409.](#)

### 1-2 Advantages and disadvantages of a distributed system in terms of security

---

<sup>1</sup>You could argue that you have to consider the instructions of the users but not the computer which would result in a completely different picture (since a human being acts much slower than any computer). But from my point of view this is not right: Since it is not about whether it is a distributed system from the user's point of view but the implementation structure (and therefore at least one level deeper) machine operations must be regarded as units.

Another objection could be, that not machine instructions but hard disc accesses must be regarded as operations since they make the state transition persistent and they can be set back for fault tolerance reasons anyway. This is surely the right view for system shaping - by implicitly introducing a distributed control- and implementation structure for computers and systems.

## B Answers

### a) Advantages:

*Confidentiality* can be supported by everybody can have their data under his physical control → End of external determination on information: It is unnecessary to give away your own data and hope for the best.

*Integrity* can be supported by everybody being able to see his own data before anyone else accesses them and confirming if they are valid or not. Of course this process can be automated to make a PC validate the data. Furthermore the logging can be distributed to complicate faking or suppressing of logging.

*Availability* can be supported by not having a single point of failure for explosive attacks or logical bombs.

Disadvantages:

*Confidentiality* can be endangered by the fact that data can be moved freely which makes controls difficult. It is doubtful that there is always a legally responsible entity, named “speichernde Stelle” (storing place) in privacy laws. *Integrity* and *availability* can be endangered by the mere existence of more possible points of attack. Just think of the many physical connections (especially wireless ones) that can hardly be secured or the many computers staying around unsupervised.

- b) Spatial distribution supports availability against physical threats. Distribution as well as control and implementation structures support confidentiality and integrity.

[Exercise at page 409.](#)

### 1-3 Advantages and disadvantages of an open system in terms of security

#### a) Advantages:

Open systems can be connected more easily to cooperating distributed systems. Therefore the advantages of distributed systems mentioned above can be achieved more easily. The compatible interfaces of open (sub)systems allow an easier creation of over-all systems using different designers, producers, and maintainers. As seen in §1.2.2 diversity can heavily increase security:

Disadvantages:

Open systems can be easily connected to cooperating and distributed systems. So the disadvantages of distributed systems mentioned above can be achieved more easily. In particular a user of an open system usually does not know who has which rights and under which role name.

- b) Therefore they amplify each other so anything said about distributed or open systems holds true for distributed open systems.

[Exercise at page 409.](#)

### 1-4 Advantages and disadvantages of a service-integrating system in terms of security

a) Advantages:

Service integrating systems can be more expensive since all efforts are shared by several services. Relatively seen this allows more effort regarding security. On the one hand this allows a more careful design and its validation (like testing for Trojan Horses). On the other hand you can employ more redundancy for increasing the availability, like additional alternative physical connections. Then maybe even the more expensive measures for protecting the anonymity of sender and recipient (cp. §5) are more likely to be achieved.

Disadvantages:

If a security property is violated this would surely have an impact on several concurrent services. This turns out to be a disaster if no spare systems are at hand (this would be a totally service integrated system [Pfitz\_89, pg. 213ff][RWHP\_89]). Access must be granted even to those users that only make use of parts of the services offered. For the unused services those participants are an unnecessary security risk.

- b) Service integration amplifies anything told here (independent from the type of the system) but this applies especially to distributed open systems.

[Exercise at page 409.](#)

### 1-5 Advantages and disadvantages of a digital system in terms of security

- a) Digitally transmitted or stored information can be altered or copied perfectly (=without traces) if the attacker is willing to take a certain minimal amount of effort. This amplifies the problems regarding confidentiality and integrity as long as the encryption techniques from §3 are not used (which are available by actually using digital transmission). Availability is supported by digital transmission and storage in common.

Computer aided relaying and processing are vulnerable to problems of confidentiality, integrity, and availability. On the other hand its enormous capabilities allow protection measures that would not be feasible elsewhere. With an appropriate system configuration this disadvantage can be more than balanced out. If you are confident about the correctness and completeness of the protection measures they should rather be implemented in hardware than in software. Mark that: Protections implemented in software often turn out to be *soft*!

- b) The arguments from 1-1 show that the property *digital* supports the construction of distributed systems.

As long as an open system is not complete (especially being incompletely specified) computer aided relaying and processing (both freely programmable) are necessary to make the systems adaptable for future developments.

Today it is inevitable that service integrated systems are digital.

[Exercise at page 410.](#)

## B Answers

### 1-6 Definition of requirements for four exemplary applications in the medical domain

- a) *Availability*: Medium critical, as long as there are local originals printed on paper. In case of failure the situation remains the same.

*Integrity*: Critical since nobody wants to make additional checks with the paper documents. Nor does anybody want to update them by hand or print them out regularly.

*Confidentiality* against external doctors: critical; against third persons: highly critical. The patient may request admission or deletion rights that are to be granted. International harmonization would be difficult.

- b) *Availability*: Uncritical, as long as the surgery team on site does not rely on the help. Otherwise: Highly critical.

*Integrity*: Small integrity violations (i.e., some broken pixels) are uncritical. Larger violations (bad images) are critical. To make the patient (or his relatives!) able to prove professional blunder requires the reproduction of who saw what and who gave the advice (even after many years). In particular, a clear identification (cp. §2.2.1) of the external advisor is needed for court.

*Confidentiality*: Confidentiality of the images: less critical; if transmitted at the beginning of the case history the content of course needs to be protected. The protection of the sender and recipient identity is superfluous. ([PfPf\_92, §4.2]).

- c) *Availability*: Medium critical, since failure “only” leads to the current situation. As long as patients remain at home who would be in a hospital otherwise: critical.

*Integrity*: False alarm is uncritical but should not be triggered too often. The opposite: see availability.

*Confidentiality* of an alarm is less critical, all other message contents as well as the patients as sender and recipient need to be protected ([PfPf\_92, §4.3]).

- d) *Availability*: Uncritical.

*Integrity*: Not very critical since for critical cases a doctor is compulsory. Anyway, while accessing a medical database it must be clear who is responsible for its content. If doctors become dependent on such a database, requirements are heavily increased towards availability and integrity.

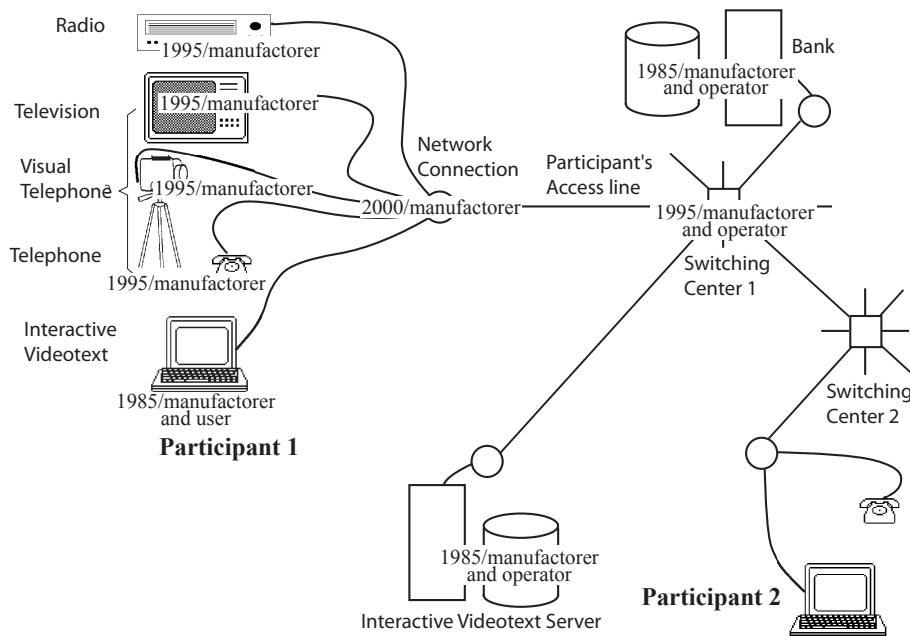
*Confidentiality* of the message content and of the request itself must be secured by anonymity ([PfPf\_92, §4.1]).

Of course I am not sure about the completeness of my protection goals.

[Exercise at page 410.](#)

### 1-7 Where and when can computers be expected?

The numbers show the year where a noteworthy spread of computer aided components (within its component class) took place or is predicted. In the near future you must consider switching stations, “harmless” end user devices or even network connection units as potential attackers.



[Exercise at page 410.](#)

### 1-8 Deficits of legal provisions

The answer consists of a combination from the arguments of §1.2.3 and §1.2.5.

A prohibition is only useful if its adherence can be *verified* (in principle impossible with observing attacks) with a reasonable effort, if it is secured by criminal prosecution and if it is possible to *restore* the original state. All that is not true for IT-Systems.

“Data theft” in general, specifically the direct wiretapping of connections or the copying of data from computers, is hardly detectable since the original data will not change. Nor is the installing of Trojan Horses or the unallowed processing of data you legally (or illegally) obtained.

The reconstruction of the original state would be to delete all data created. But you can never be sure that there are not any further copies. Besides data can remain inside the human’s brain where deleting is not a worthwhile option.

Parallel to the problem of “data theft” is the problem of data loss: Maybe lost data can not be restored or reproduced so that the original state can not be brought back.

So more effective measures are needed.

## B Answers

*Annotation* For IT-Systems crossing the national borders an international agreement (expressed by laws) and its enforcement (by international courts) are needed. While there are many such IT-Systems, they are not covered by many international agreements.

Exercise at page 411.

### 1-9 Alternative definitions/characterizations of multilateral security

“The large-scale automated transaction systems of the near future can be designed to protect the privacy and maintain the security of both individuals and organizations. . . . Individuals stand to gain in increased convenience and reliability; improved protection against abuses by other individuals and by organizations, a kind of equal parity with organizations, and, of course, monitorability and control over how information about themselves is used. . . . the advantages to individuals considered above apply in part to organizations as well. . . . Since mutually trusted and tamper-resistant equipment is not required with the systems described here, any entry point to a system can be freely used; users can even supply their own terminal equipment and take advantage of the latest technology.” from [Cha8\_85]

“Meanwhile, in a system based on representatives and observers, organizations stand to gain competitive and political advantages from increased public confidence (in addition to the lower costs of pseudonymous record-keeping). And individuals, by maintaining their own cryptographically guaranteed records and making only necessary disclosures, will be able to protect their privacy without infringing on the legitimate needs of those with whom they do business.” from [Chau\_92]

“Considering the growing importance of legal affairs using open digital systems, the need arises for making usage of those systems unobservable for non participants and anonymous for participants by keeping up the necessary legal certainty” from [PWP\_90]

“The term multilateral security is therefore used here to describe an approach aiming at a balance between the different security requirements of different parties. In particular, respecting the different security requirements of the parties involved implies renouncing the commonly used precondition, that the parties have to trust each other, and, especially, renouncing the precondition that the subscribers have to place complete trust into the service providers. Consequently, each party must be viewed as a potential attacker towards the other and the safeguards have to be designed accordingly.” from [RDFR\_97].

“Multilateral security means to respect all security needs of all participating parties.” from [Rann\_97].

“Multilateral security means to include the security interests of *all* participants as well as to deal with the resulting conflicts during the creation of a communication connection.” from [FePf\_97]

“Multilateral security means to include the security interests of *all* participants as well as to deal with the resulting conflicts during the creation of a communication connection.

The creation of multilateral security does not inevitably lead to the fulfilling of all the participant’s interests. It possibly reveals incompatible interests the participant were not aware of since protection goals are formulated explicitly. But it inevitably leads to the interacting of the participants using multilateral communication systems to achieve a certain balance of powers.” from [Fede\_98]

Multilateral security is *security with minimal assumptions about others*:

- Every participant has security interests.
- Every participant may express his interests.
- Conflicts are to be recognized and solutions to them are to be negotiated.
- Every participant may enforce his security interests in negotiated solutions.

from [FePf\_99]

“... new developments, e. g., the rising need and demand for multilateral security, which cover the requirements of users and uses as well as those of IT system owners and operators. ... The TCSEC had a strong bias on the protection of system owners and operators only. This bias is slowly losing strength, but can still be seen in all criteria published afterwards. Security of users and uses, especially of users of telecommunication systems is not considered. Therefore techniques, providing bi- or multilateral security, e. g., those protecting users in a way privacy regulations demand it, cannot be described properly using the current criteria.” from [Rann\_94]

“Multilateral security emphasizes the equal protection of all participants. The protection goals of privacy and liability are considered as important priorities with regard to personal freedom of man and obliging social behavior” from [PoSc\_97]

“While for classic security models the operator’s view stays in front, multilateral security respects the participating user’s view. Besides the protection the networks offer to their users, the user needs mechanisms for self-protection for reducing his dependency on others. Not only are they now protected from attacks from the network but they also keep up their mutual interests like buyer and seller exchange money and receipts sometimes even with notary supervision.” from [GGPS\_97]

“Multilateral security is a user-centric feature. The protection demanded is finally for the people and organizations that do employ telecommunication systems to do their tasks. However security serves not only the communication partners but also all others that are somehow connected to the partners or the content transmitted...” from [RoSc\_96]

“Multilateral security: Whereas the introduction of new communication media and services primarily targeted the *security of the provider* (like against unpermitted usage) now the *security of the participant* is to be emphasized.” from [BDFK\_95]

“It seems quite legitimate to demand that users be treated equally in terms of their security rights. If you promise every user that their security does not depend on the goodwill

## B Answers

of others, but only on their own behavior, such a system would be truly multilateral.” from [PSWW\_96]

“Multilateral security means the upholding of morale in an information technical sense.” a student in winter term 99/00.

[Exercise at page 411.](#)

## B.2 Answers for “Security of single computers and its limits”

### 2-1 Attackers beyond the wisdom of textbooks

To begin with, an *attacker with unlimited capacity* in terms of computation power, time, and memory comes into mind. Fortunately we will see that there are corresponding information-theoretical techniques to achieve security (confidentiality and integrity) against such a complexly, theoretically unlimited attacker.

However it is challenging to fight against attackers that are owners of unknown technical, physical, or chemical innovations (like new measurement tools, explosives, . . . ).

And it becomes highly critical if they posses the ability to control new physical phenomena: new kinds of radiation, fields, etc.; Or applied to humans: Supernatural senses (X-Ray-viewing capability, soothsaying) or abilities (passing-through-the-wall, invulnerability, telekinesis). This is not meant to start a parapsychological discussion, but gives warning that perfect security can only be achieved in a *closed world* with no new perceptions. Luckily, as we are all curious students, we live in an open world. Here are some examples of closed world attacks:

- *Unpermitted information gaining*: Against an attacker with soothsaying abilities neither shielding (cp. §2.1) nor cryptographics (cp. §3) help to protect confidentiality.
- *Unpermitted modification of information*: Against an attacker with telekinetic abilities shielding (cp. §2.1) does not protect confidentiality. Against an attacker with telekinetic and soothsaying abilities cryptographics (cp. §3) will not help either.
- *Unpermitted affecting of functionality*: Against an attacker with telekinetic abilities neither shielding nor (finite) fault tolerance will help to maintain availability.

[Exercise at page 411.](#)

### 2-2 (In)Dependence of confidentiality, integrity, availability with identification as example

You can not expect any “secure” IT systems if you have at least one of the three protection goals without any requirements at all:

- If there are no requirements regarding *availability* then no IT systems at all needs to be realized. With that, all its (existing) pieces of information are good in terms of integrity and confidentiality.
- If there are no requirements regarding *integrity* an IT system can not distinguish between different persons since the information needed for that could have been modified without permission. The same applies for distinguishing different IT systems.
- If there are no requirements regarding *confidentiality* an IT systems can not identify itself towards others because an attacker could gather all necessary information from the IT system itself. The other way round is not that simple: “Since an IT-Systems can not keep any information confidential it can not recognize other instances since an attacker could obtain all data required to do so from the IT system to distinguish other instances.” This argument seems consistent but the IT system could apply a collision resistant hash function (cp. §3.6.4 §3.8.3) to the information expected from the other instance and save only the result of it. Later on the result is compared with the input applied on that hash function.

[Exercise at page 412.](#)

### 2-3 Send random number, expect encryption – does this work conversely, too?

The difference is subtle:

With the normal version you only have to assume that random numbers will not repeat (with a relevant probability) during random number generation.

With the opposite version you *additionally* have to assume that the random number generation is unpredictable for the attacker. Actually this should be true for any random number generator (just like what the name says) but it is always safety first.

The difference becomes clear with an extreme example: The normal version is secure if the random number generator is a counter that is incremented after every “random number”. With this implementation of a random number generator the opposite version would be completely insecure.

For further interest: In [NeKe\_99] you will find, next to a detailed explanation of this example, a worthwhile introduction to a formalism (BAN-Logic) with which you can examine such protocols.

[Exercise at page 412.](#)

### 2-4 Necessity of logging in case of access control

- a) The confidentiality problem is reduced with every level of the “recursion” since the amount of personal related data is (hopefully) getting fewer and less sensitive as well. Or the other way round: The problem is not really recursive since the access to the secondary data can be more restricted than for the primary data.

## B Answers

For example you could print out the secondary data and prevent the manipulation afterwards by spatially remote people. Furthermore it is not useful to let the secondary data be created too thoroughly; it should really get fewer. Finally you can reduce the circle of people that *may* access the secondary data to a few certain trustworthy ones. Hopefully this reduces the circle that *may* access. If you introduce tertiary data you can have them evaluated by other privacy commissioners as the secondary data. Then the controllers control each other. Assuming the use of nonalterable protocols you could realize a time staggered  $2n$  eye principle (corresponding to the known - not time staggered - 4 eye principle) for  $n$  levels.

- b) You should strive for a strict and detailed regulation concerning who may do what under which circumstances. Then the amount of logging can be reduced: As much logging as needed but no more.
- c) Here the “sensitivity” does not decrease with every level but it increases: The regulation for who may grant rights is at least as security critical as the enforcing of the rights and their limits. The same as for a) is true regarding that the granting of rights may be subject to stronger security regulations and other security regulations than for the usage of rights, the granting of rights for rights granting to other and stronger security regulations and so and so on.

[Exercise at page 412.](#)

### 2-5 Limiting the success of modifying attacks

Modifying attacks should be a) *instantly detected* and all unpermitted modifications should be b) *localized as precisely as possible*. After that c) *the state* should be recovered as fully as possible compared to the original state *without the modifying attack*.

- a) Besides the commonly applicable information technical techniques such as
  - **Error detecting encoding** for transmission and storage
  - **Validation of the authenticity of all messages** before their processing (authentication codes or digital signatures, cp. §3) so that faked commands in the messages will not be executed at all. The same applies for storage content that can be read from unprotected areas (like floppy discs, CDs, USB media devices).

special tests of plausibility exist for every single application. For example, a person can travel a maximum of 1000 km/h. This can be used to discover if an attacker pretends to be somebody else who reported earlier from somewhere else.

- b) Here it helps to *log* who had accessed what and when (cp. §2.2.3).
- c) Here you must differentiate between hardware and data (incl. programs). **Hardware** has to be repaired or replaced very quickly. Or you have alternatives like

## B.2 Answers for “Security of single computers and its limits”

backup computer centers, alternative communication networks. The usage of hardware is mostly a financial problem.

For **data** (incl. programs) you will need *backups*. Here the following dilemma exists: On the one hand the last current state possible should be recovered so that a minimum of work is lost. On the other hand if unpermitted modifications took place the current state should be rolled back to the latest version that was unmodified. What now? The backup system must allow to set back the distance of recovery dynamically. This can be done by conducting a daily **full backup**, and weekly, monthly, and per year. The daily backup can be overwritten weekly, the weekly every month, and the year backup never, please - your company should be able to afford that many streamer tapes! An alternative would be to make a backup from time to time but also to create a **logfile** where all changes (incl. the exact values) are recorded. Using the current state and the logfile you can rollback to any previous state desired - as well go forward with an old backup and the logfile. Please remember to conduct a backup of the logfile as well. And of course you can combine that too.

[Exercise at page 412.](#)

### 2-6 No hidden channel anymore?

Unfortunately not! If our military really needs its computers there is the following hidden channel: Every time a *computer fails* the military will order a new one. If they do so within the next 24 hours and if you assume that computers fail every 10 years (=3652.5 days with 2.5 intercalary days) so there is a hidden channel with almost (assuming a perfect computer that will not fail except for the usage of the hidden channel: exactly)

*ld 3652, 5 bit*

and therefore more than 11 bit in 10 years.

Of course you can object that *illoyal staff* represents a far more bigger hidden channel. But I am only considering the information technical point of view as well as the usually overlooked aspects.

By the way, the *loyal staff* may be a hidden channel of a bigger bandwidth too: Consider a Trojan horse that is placed in a computer where it can influence the video output of the terminal by simply reducing the image quality. Then the attacker only has to observe if the staff leaves the hermetically shielded rooms with reddish eyes.

[Exercise at page 413.](#)

### 2-7 Change of perspective: the diabolical terminal

The properties for a) are first followed by the motives of the participants to use such a device. The consequences are stated behind the implication arrow. Keywords found in the PC handbook are marked in *cursive*.

## B Answers

Using the device simultaneously for as many purposes as possible to make its inner workings as inscrutable as possible. The motivation for the user to tolerate that, is that he can use his data for many purposes which makes the device look like it has a good price performance relation → service integrated terminal device.

The device should count as useful, nifty, and advanced or even as inevitable → service integrated device, useful, nifty, advanced, small, colorful, communicative ... as requested by basic agreements for big companies or public institutions.

The device should be an alterable → programmable device; dynamical loading of software: *PC, PDA, ActiveX, BHO, Plugins*

The responsibility for changes should belong to someone who does not completely understand what he is doing. But he does not realize that and he has good reasons for doing the things. The user itself would be a perfect match. Then others could say: Blame yourself ... (A subtle variation would be a preinstallation by the seller according to the buyers' wishes)

As much computation power and storage as possible that are used for inscrutable purposes so that the resource usage for (universal) Trojan horses will not attract any attention.

Since Trojan horses will not work perfectly at first, system crashes should be seen as normal.

Inscrutable and complex closed source programs.

Proprietary, publicly undocumented data formats instead of openly documented or standardized ones, e.g., *Word* instead of *RTF*.

No access control (so that every program can influence every other). Not to mention access control employing the Least Privilege principle. → *Windows 98* and *Mac OS 9* do not have any access control. *Windows 2000* and *Linux* do not have access control following the Least Privilege principle.

The device should have universal, bidirectional interfaces so the user can connect to everything fairly easily. Especially good are public open networks → *Internet* On the one hand for dynamically reloading of software, on the other to enable attackers to access their Trojan horses as often as possible without being noticed.

Giving away user data (content, user ID, or the ID of the device) of which the user is not aware of → Transferring plain data over open networks like the Internet. Adding the creator device of every object into the digital representation of the object (DCE UUID of OLE or the Microsoft *GUID*=Global Unique Identifier). Device IDs like Ethernet addresses *mac address* or processor IDs (*Pentium III*) undermine the anonymity of the user since they can be transferred without being noticed.

[Exercise at page 413.](#)

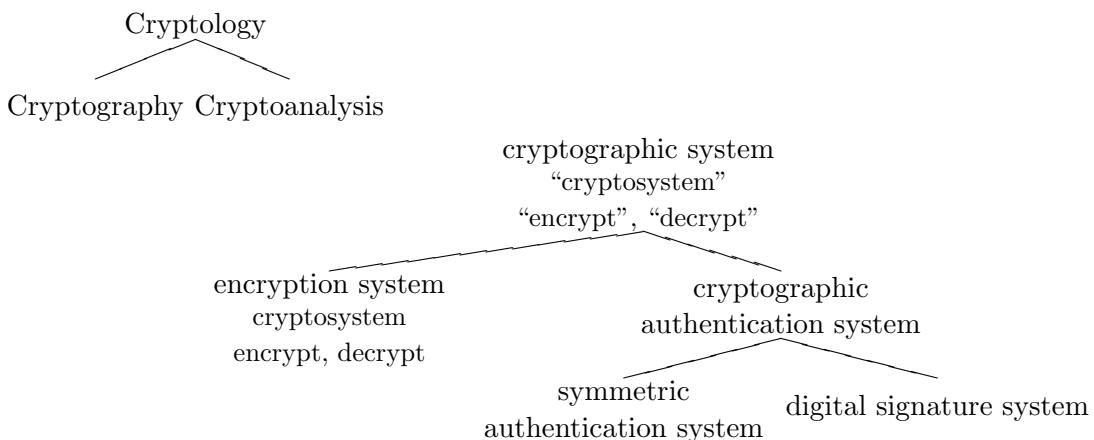
## B.3 Answers for “Cryptography”

### 3-1 Terms

Cryptology encompasses cryptography (the science of algorithms of the “good” users) as well as cryptoanalysis (the science of algorithms used by attackers on cryptographic systems). Knowledge in the area of cryptoanalysis is necessary to evaluate the security of cryptographic systems and its applications.

Cryptographic systems are either encryption systems (protection goal: confidentiality) or authentication systems (protection goal: integrity). Authentication systems are either symmetric (and therefore not suitable as legal evidence) or asymmetric and therefore suitable as legal evidence towards a third person. Because of this important property asymmetric authentication systems are also called digital signature systems.

The advantage to place terms like *cryptosystem*, *encrypting*, and *decrypting* at the top of the tree of terms is to use short common terms almost everywhere. The disadvantage is that the distinction between encryption and authentication is pushed into the background so that I will use these terms only for cryptographic encryption systems – if used in a more common meaning I will use quotation marks. My tree of terms looks like this:

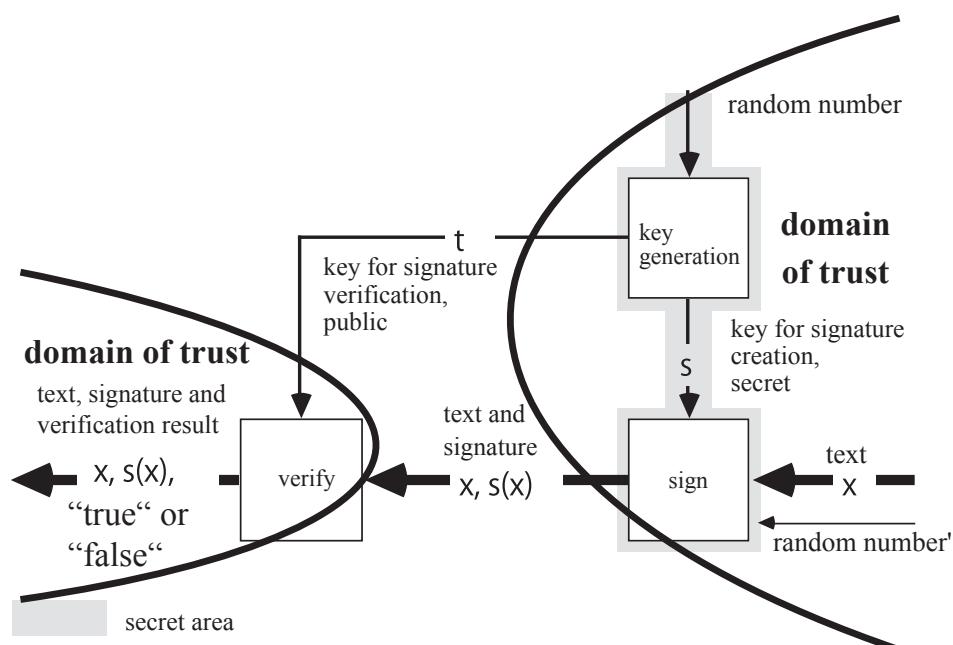
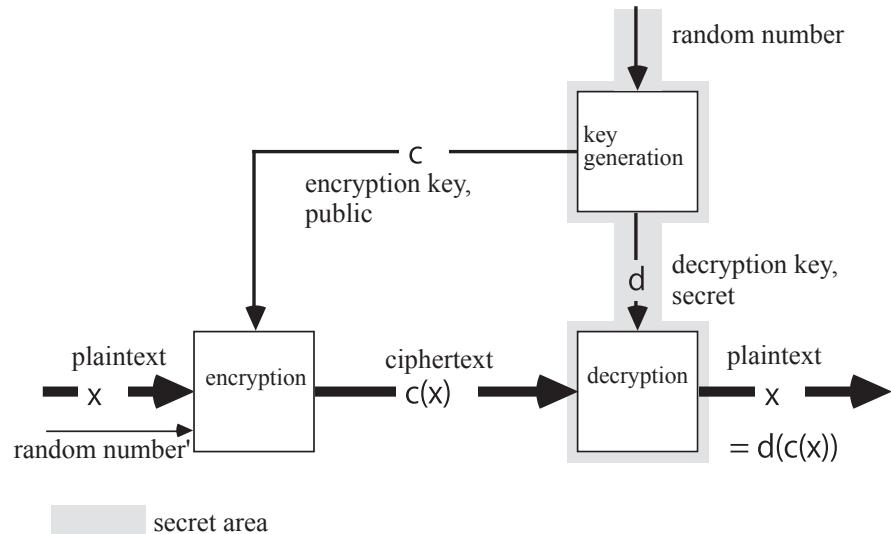


[Exercise at page 413.](#)

### 3-2 Domains of trust for asymmetric encryption systems and digital signatures

The secret key can be protected (in terms of confidentiality and integrity) by generating it inside the device (so the region of trust) in which the key is used. The following pictures arise:

## B Answers



A backup copy of the signing key is not necessary since signatures remain testable and therefore valid even if the key is destroyed. The signing key should never leave the signing device. If the key is not available anymore you will just create another key pair, get the public part certified - and there you go. The same applies to communication with asymmetric concealment: After generating and exchanging the keys the message is just encrypted once again. If using asymmetric concealment for storing data, you may either need plaintext backups of the data or the encrypted data and the decryption key.

[Exercise at page 414.](#)

### 3-3 Why keys?

- a) You can do without keys too: For symmetric cryptographics two users need to exchange a secret and exclusive cryptographic algorithm. For asymmetric cryptographics you will have to publish a public algorithm from which it is impossible to derive the secret one.
- b) The disadvantages are numerous and fatal:
  - Where do users get good cryptographic algorithms from? Eventually non-cryptographers want to communicate confidentially with integrity without anybody else (this includes the cryptographer as well) breaking its security. Finally, the cryptographer that created the algorithm would needed to be “removed” since he knows the secret. While this was done in the ancient times, today all people (even cryptographers) are subject to the Human Rights.
  - How to analyze the security of cryptographic algorithms? Anybody analyzing a symmetric algorithm is just as much a security risk as the inventor. For asymmetric algorithms anybody who knows the secret algorithm for validating purposes is a security risk.
  - Implementing the cryptographic systems in software, hardware, or firmware is possible only if the user is able to do this on his own. Otherwise the same applies for the implementor as for the algorithm inventor.
  - In public systems where potentially everybody wants to communicate securely with everybody else, only software implementations with algorithm exchange instead of key exchange would be possible - firmware depends on the machine, and hardware exchange would require physical transportation. But even software implementations have the disadvantage of requiring (for symmetric systems with confidentiality and integrity and asymmetric systems full of integrity) transmissions of huge amounts of bits: Algorithms are usually larger than programs.
  - A standardization of cryptographic algorithms would be impossible due to technical reasons.
- c) The main advantage of keys is: The cryptographic algorithms can be published. This allows:
  - + a broad and public validation of its security
  - + its standardization
  - + any implementation form desired
  - + mass production
  - + a validation of the implementation’s security as well

[Exercise at page 414.](#)

### 3-4 Adjustment of keys in case of autonomous key generation by participants?

The objections against autonomous decentral key generation by participants are wrong (cp. below). Thus this proposal is unnecessary and in terms of the confidentiality of the secret keys - as you know - very dangerous.

Of course, you can not totally rule out that the random number used for initializing the key generation occurs more than once with the consequence that the secret part of the key can be generated multiple times. But the conclusion that therefore the key generation needs to be coordinated is pretty obscure. If the same random number occurs more than once either the random number generator is bad or the number is too short. In both cases it will not help to use the random number once and discard it if it occurs again. Because nothing would prevent an attacker from creating equally long random numbers with the same generator. Since this can not be prevented, only one thing helps: Carefully picking and validating the random number generator as well as making random numbers long enough. Anything else is decoration, which distracts from the real problem.

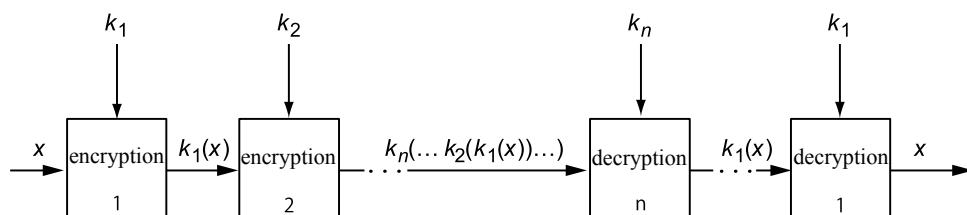
Besides: If keys are compared it is sufficient to do so with the *public keys*. So the key generation can stay autonomously - the key comparison becomes part of the key certification.

[Exercise at page 415.](#)

### 3-5 Increasing security by using multiple cryptographic systems

For the protection goal **confidentiality** you may use cryptographic systems sequentially. Then the plaintext is encrypted with encryption system 1, the result is encrypted with encryption system 2 and so on (cp. Figure 3.2 where  $k_i$  is the key of the  $i$ -th encryption system:  $S := k_n(\dots k_2(k_1(x))\dots)$ ). Of course every system needs its own unique key and the applying order must be the same.

Decrypting is the other way round; therefore as last with encryption systems 1 which reveals the plaintext:  $x = k_1^{-1}(k_2^{-1}(\dots k_n^{-1}(S)\dots))$ .



Of course asymmetric systems can be used that way too. Even a mixed usage of symmetric and asymmetric encryption systems is possible. In the example you could replace every  $i$ ,  $1 \leq i \leq n$  the  $k_i$  on the left with  $c_i$  and on the right with  $d_i$ .

This construction of course only works if the key text space of encryption systems  $i$  is a sub space of the plaintext of encryption system  $i + 1$ . In practice this can be

easily achieved (cp. §3.8.1,§3.8.2). If plaintext and key text do have the same length on all  $n$  systems this measure will not affect memory and transmission complexity (but at the most time it will increase due to the arrangement into blocks). The *computation complexity* is the *sum of all computation complexities* of every single encryption system.

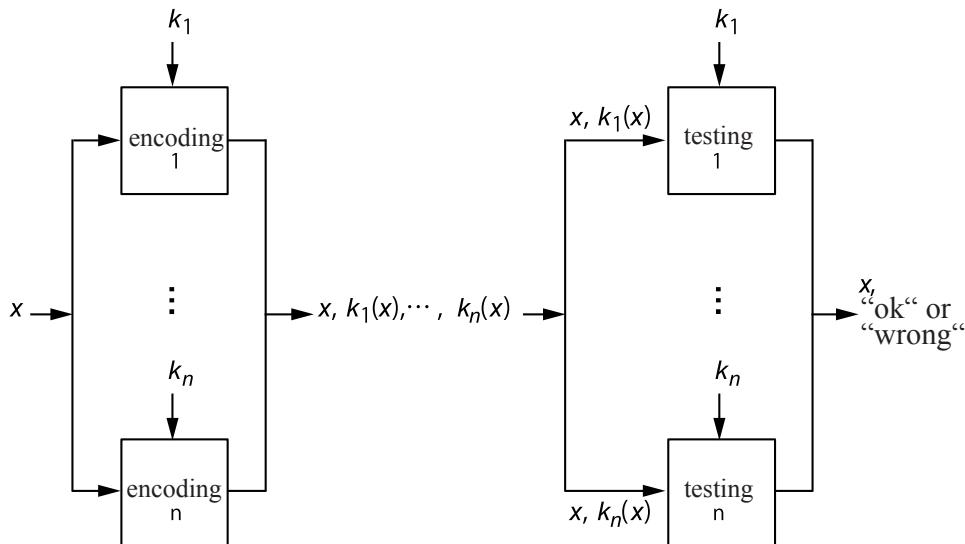
Besides: The encryption system created by using several sequential encryption systems is called **product cipher** of the encryption systems used.

For the protection goal authentication you can use the cryptographic systems in **parallel**: For every system the authenticator of the original message  $x$  is created and all of them are consecutively appended on  $x$  (with separators; cp. Figure 3.6:  $x, k_1(x), \dots, k_n(x)$ ). The recipient checks every authenticator separately: They all have to be valid.

Of course you may use digital signature systems that way too, even a mixed usage is possible. In the example you could replace every  $i$ ,  $1 \leq i \leq n$  the  $k_i$  on the left with  $s_i$  and on the right with  $t_i$ .

Here the keys used must be picked independently, too.

*Computation complexity* and *memory and transmission complexity* for the authenticator are exactly the sum of each single system. The computation can be done in parallel so that the whole system (properly implemented) is as fast as the slowest authentication system used.



For the protection goal concealment as well as for the protection goal authentication, you will need additional memory to house the software that implements the cryptographic systems.

About the *provability* of both constructions:

*Systems parallel to the authentication* can be trivially proved since nothing of the usage of the single systems would change, and especially not the distribution of messages and key texts/MACs.

## B Answers

*Sequential Systems for concealment* can not be proved at all. The distribution of the inputs of “encryption 2” up to “encryption  $n$ ” is changed by the construction - instead of the plaintext the key text is encrypted. This may happen in such an unfortunate way that the encryptions 2 up to  $n$  will not support security at all so that Sequential Systems would be only as secure as “encryption 1”. In practice this would surely only occur if “encryption 2” up to “encryption  $n$ ” are explicitly designed to be “unfortunate” (in Exercise 3-31 you will find a provably secure (in a certain sense), but more costly construction for using several encryption systems).

Reversely, the proof for *systems parallel to the authentication* also shows that the construction is *only* as secure as it is hard to break all the systems used. Also reversely, the proof for *sequential systems for concealment* fails, mainly because the attacker does not have the interim results (the key texts between the encryption systems used) - this is what makes his task unequally harder than to *merely* break all encryption systems used.

Discussion of *wrong* solutions:

To encrypt the plaintext with several encryption systems, where some of them are unsecure, the message is split into as many parts as you have encryption systems. Then every part is encrypted with another system. *Disadvantage:* The probability that an attacker can decrypt at least some parts of the message is usually increased and not decreased.

To authenticate the message with several authentication systems you create the 1. validation record with the first authentication system. The second system is used to authenticate the 1. validation record; so the 2nd validation record is created, ... *Disadvantage:* If an attacker can break the first by finding another message for the 1. validation record all other authentication systems become ineffective (because the 1. validation record for the other matching part is not altered, the remainder can stay untouched). Besides, the computation of the validation record would not be that easy to parallelize.

To save key exchange efforts the same key is used for several cryptographic systems (if possible at all). *Disadvantage:* In many cases the resulting system is not more secure, sometimes even weaker than the weakest of the systems used. This can be easily shown for authentication: Consider, all systems require the same key. Then it is clear that breaking one system results in the breaking of all systems. If some systems allow effective gaining of information about the key, the sum of all these information may be sufficient for a successful attack.

Discussion of *clumsy* solutions:

You use authentication system  $i$  not only to create a validation record for message  $x$  but also for the validation records 1 to  $i - 1$ . Of course this solution is secure but unnecessarily expensive:

- a) The calculation effort of the validation records is increasing proportionally with the length of the string to be authenticated.

- b) The calculation of the validation records is no longer parallelizable since validation record  $i$  depends on part  $i - 1$ .

[Exercise at page 415.](#)

### 3-6 Protocols for key exchange

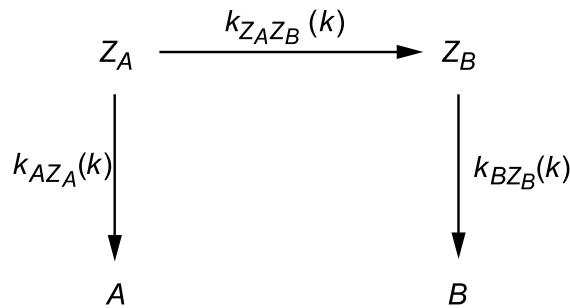
- a) The functionality of *one* key exchange center has to be provided by *several* others. For this either:

- all key exchange centers know each other’s keys directly or,
- the key exchange centers require a meta key exchange center.

In the latter case two key exchange centers need to exchange two keys like two normal participants do.

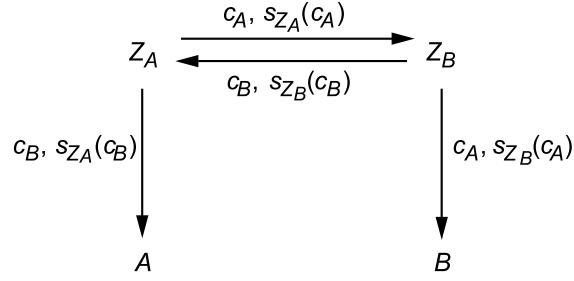
From now on we will assume that both the key exchange centers  $Z_A$  and  $Z_B$ , with which the participants  $A$  and  $B$  have exchanged keys, now have a common key.

Symmetric system: Now one of the two key exchange centers may generate a key and transmit it confidentially to the other (i.e.,  $Z_A$  generates key  $k$  and sends it to  $Z_B$  encrypted with  $k_{Z_A Z_B}$  (their common key)). Now every key distribution center securely tells “their” participants the key (i.e.,  $Z_A$  sends  $k_{AZ_A}(k)$  to  $A$ , and  $Z_B$  sends  $k_{BZ_B}(k)$  to  $B$ ).



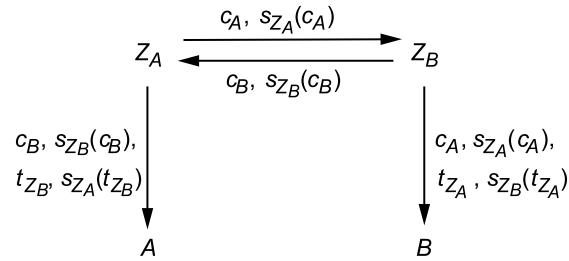
*Annotation* Authentication of public keys: The public key of every participant is authenticated by “its own” center and sent to the others center ( $Z_A$  sends the key  $c_A$  to  $Z_B$ , signed with  $s_{Z_A}$ ). The center of the other one checks the authentication and authenticates the key itself if necessary so that the recipient can check this second authentication ( $Z_B$  checks with  $t_{Z_A}$  and signs  $c_A$  with  $s_{Z_B}$ ).

## B Answers



The resulting key exchange is of course only as confidential as the weakest key exchange sub protocol.

Annotation: In the case that the authentication is being done by digital signature systems, an alternative would be that the center of the other participant is not authenticating the public key of the first participant but the public key of his center. Then the participant can validate the authentication of the public key of the other participant on his own.



The advantages of this alternative are

- + Centers can store less (in the previous alternative  $Z_B$  has to store  $c_A, S_{Z_A}(c_a)$  since only then  $Z_B$  can justify the issuing of the  $c_A, S_{Z_B}(c_a)$  certificate).
- + It is clear who has checked what (and who is legally responsible).
- + It is transparent to the participant how many steps the authentication of the public key took
- + This alternative can be applied more generally (cp. exercise 3-6 part g).

The disadvantages are:

- The participants have to check several signatures to prove the authenticity of the key.

From my point of view the advantages are significantly prevailing - and they will increase with the growth of the participant's computation power.

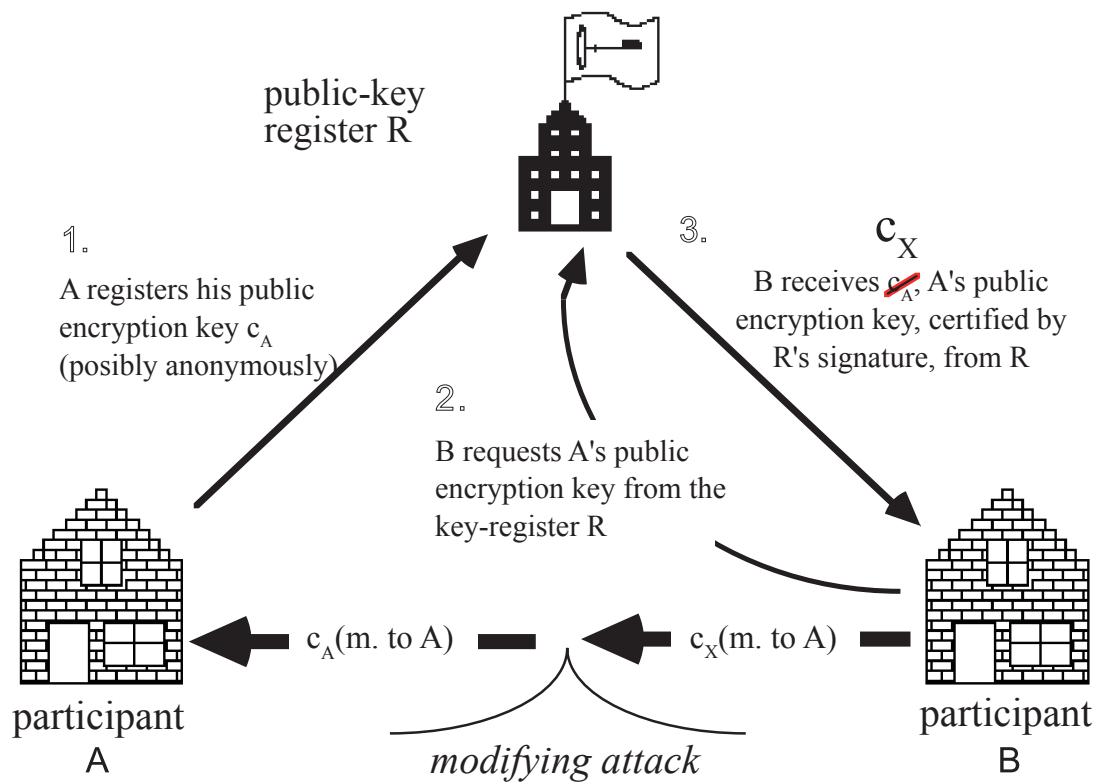
- b) *Optional:* Likely your solution is made of several key parts that are exchanged with two different key distribution centers (one per country), and the participants are added modulo 2. This is better than to exchange the key as a whole. However not all key exchange centers per country need to cooperate to get the key. It is sufficient, if one of the two cooperates. This means that the decision of the participants which are the centers in their country that will not cooperate at all is circumvented. When we suppose that all key distribution centers have exchanged keys, then there is a better solution that respects the local decision of both participants:

In the first phase one of the participants exchanges key parts over all his key exchange centers with all the others centers. Then the participant and the others participants key distribution centers add each of them modulo 2 to gain a secret key. If the participant has  $a$  key exchange centers and the other one has  $b$  then  $b$ -time  $a$  partial keys (so  $a \cdot b$ ) are exchanged to gain  $b$  keys.

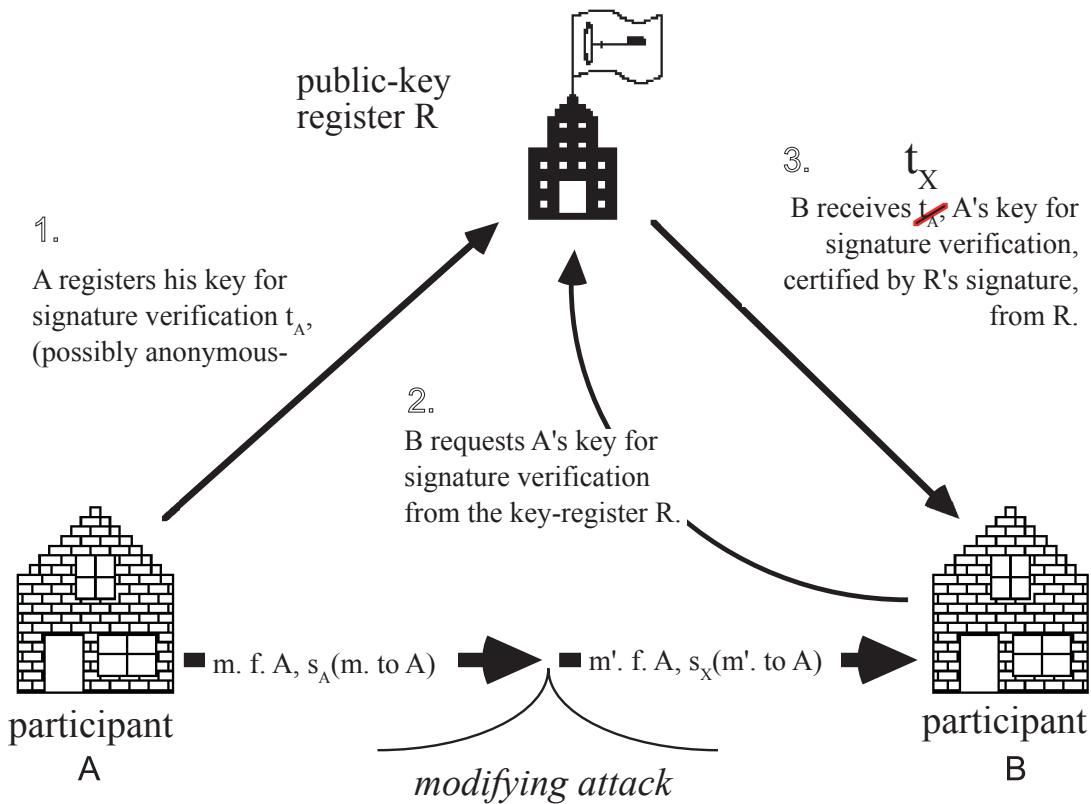
In the second phase those  $b$  secret keys (exchanged with the key distribution centers of the other participant) along with the keys exchanged between the other participant and his key distribution centers are used to exchange  $b$  partial keys (which of course have nothing in common with partial keys from phase 1) between the participants (cp. Figure 3.3).

- c) Yes, parallelizing is useful for asymmetric systems too. There are methods existing for an **altering attack** on a single (or a few) key registers  $R$ : It could hand out wrong keys to which the key register knows the secret keys. As seen on the following figures it could on the one hand break confidentiality of the messages unnoticed by reencrypting them. On the other hand it could fake them by resigning or newly signing.

### **Altering attack on concealment by reencryption of messages**



**Altering attack on authentication by resigning of messages**

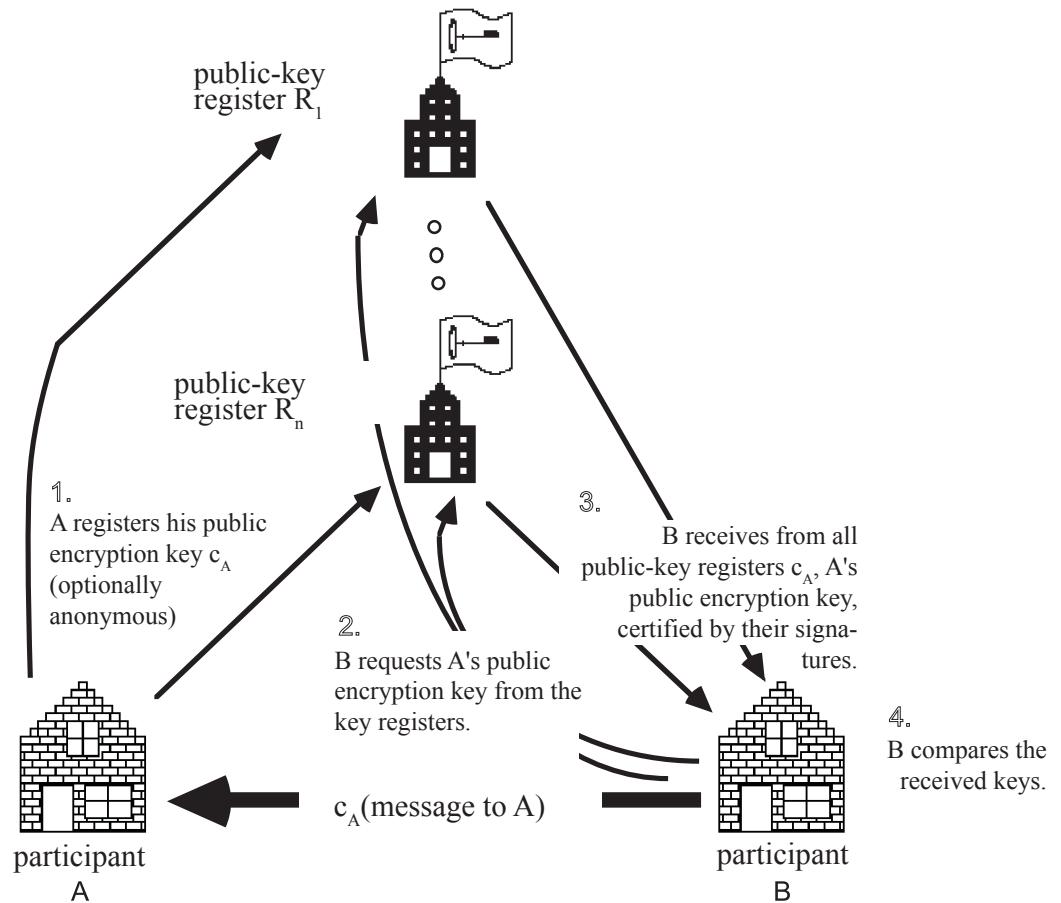


The parallelizing is done as follows: Every participant  $A^2$  publishes his own public key to several public key registers. Every participant  $B$  requests  $A$ 's public key from these registers and compares them. If some of them are not equal either the signature system was broken (modification on its way to participant  $B$ ) or at least one of the key registers is cheating, or participant  $A$  did not tell every register the same key. The latter case can be excluded by “honest” key registers by synchronizing the key entries. Again you can not distinguish all cases clearly so availability is not guaranteed.

---

<sup>2</sup>This is pretty simple but not precise. It should be read as: Every participant in a role that has been used by participant  $A$  before ...

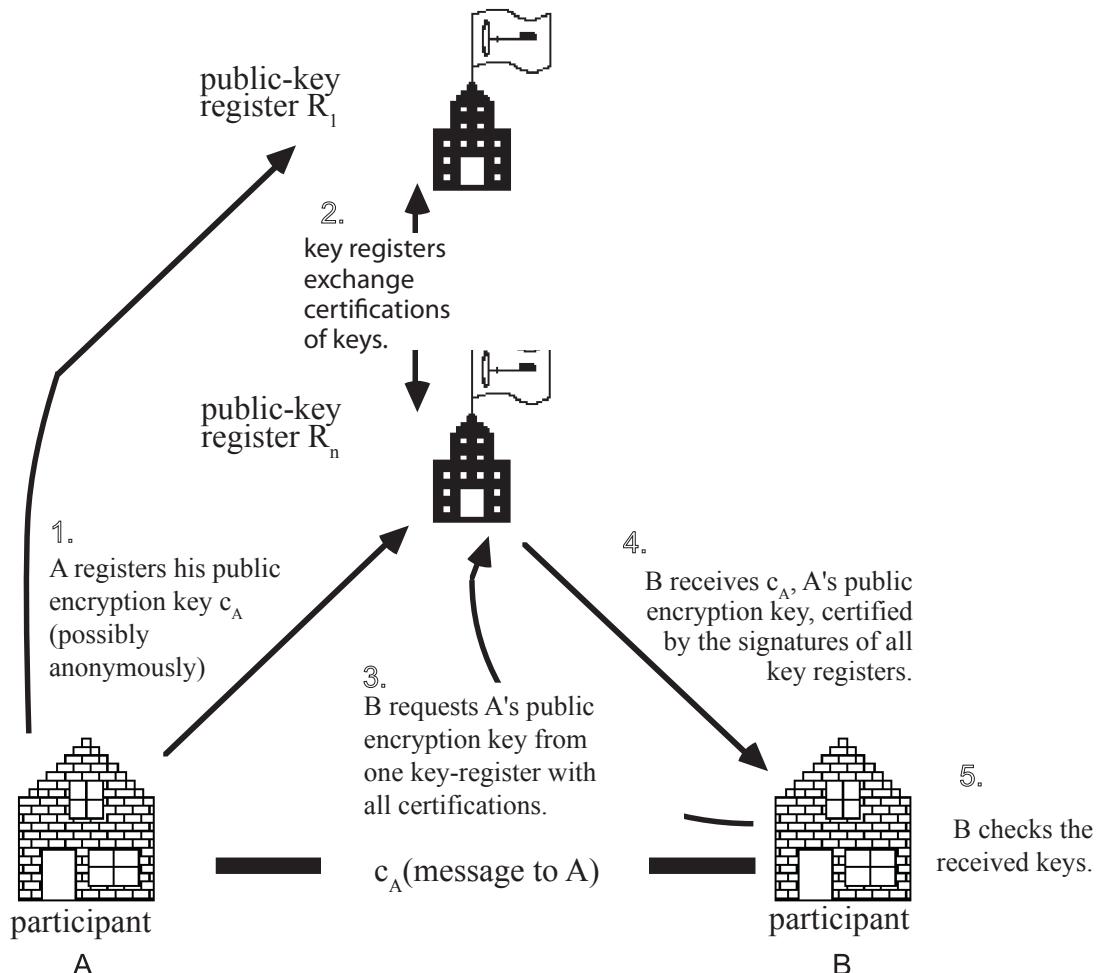
## B Answers



To make at least some cases distinguishable the participant who passes a public key to a key register should provide a written “statement” showing which key is his public key. Alternatively the participant should receive a statement from the register which key they have put into their database. This second signature can also be a digital signature since the register’s public test key is known. After this exchange any disputes between both parties are as easy to solve as the weakest of the two systems is secure.

There are at least two variations to parallelize asymmetric cryptographic systems:

- a) If public keys are to stay valid forever you do not have to check for their expiration. Hence every key register may save the other’s key registers signatures with the keys. Then a participant may request the key and all the signatures associated with it of another participant from *one* key register. This saves additional communication. Nothing will change for the signature validation.



- b) If you have doubts about the security of the asymmetric cryptographic system you may use several of them instead of one (cp. Exercise 3-5). And instead of using a single multi-authenticated key you would use multiple keys correlating to the numbers of systems used. For the concealment system  $KS_i$  the key text of  $KS_j$  is encrypted with  $KS_{j+1}$ . For the digital signature system the plaintext is signed and the signatures are concatenated.
- d) Yes. As to be seen in the next figure you conduct the known key distribution protocol, each for symmetric and asymmetric concealment, in *parallel*:<sup>3</sup>

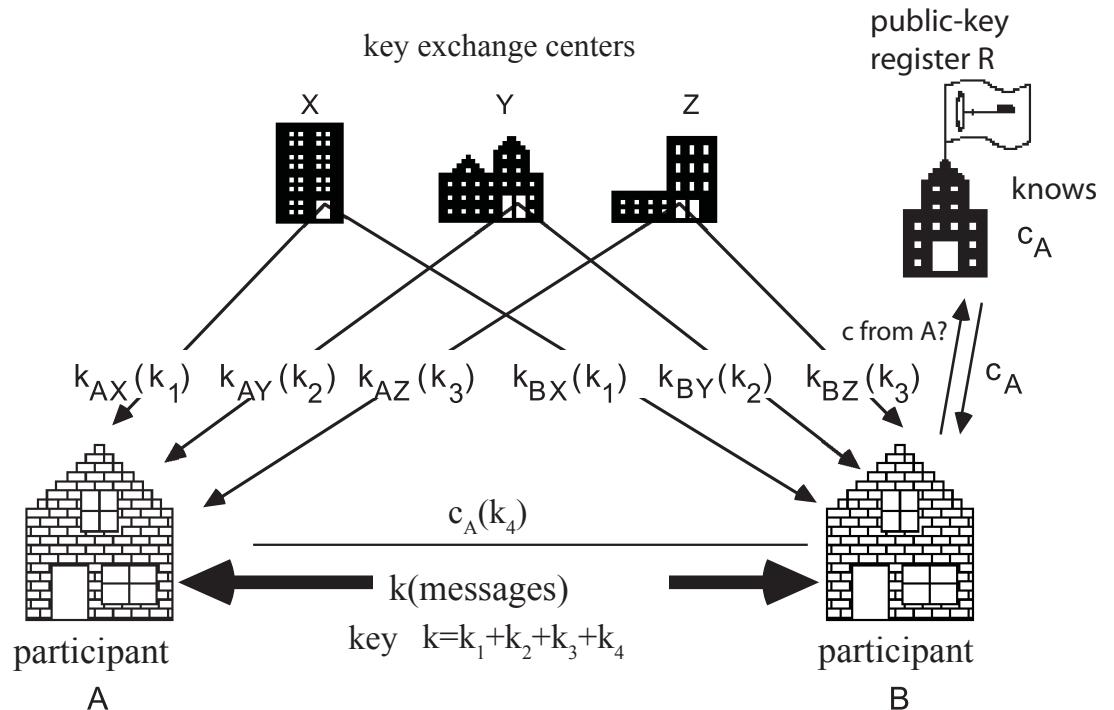
A participant  $B$  then sends a key  $k_4$  of the symmetric encryption system to the other one. The key is then concealed with the public key  $c_A$  (that is known to

---

<sup>3</sup>If you want to resist modifying attacks on the public key register you can and should use the parallelizing (which is not shown in the figure to keep it simple) for the public key registers (cp. answer to Exercise 3-6c)

## B Answers

*B)* of his partner *A*. Both partners use the sum  $k$  of all the symmetric keys ( $k_1, k_2, k_3$ ) they received from the key distribution protocol and the additional key  $k_4$ . To acquire that sum an attacker would have to gather all summands. And this would require the passive help of all key distribution authorities *and* to break the asymmetric encryption system.



After the key exchange you have to use a *Vernam cipher* (one-time pad) for concealment and an *information theoretical safe authentication code* for authentication. If you suspect *active* attacks from key registers the single key register should be extended by further ones (cp. Part c).

*Annotation :* The following variation of the question can be relevant *in practice*:

We do not assume that the attacker (while not receiving passive assistance from the key distribution authorities) is complexly theoretically unlimited but “only” able to break asymmetric cryptographic systems. Then you will not need a Vernam cipher nor information-theoretically secure authentication codes.

- If the key is used within a symmetric encryption system the partner can not decrypt correctly anymore. If the key is used in a symmetric authentication system the messages that were “correctly” authenticated would be classified as “fake” too.

To prevent any hidden alteration of the keys, they should also be authenticated instead of merely being concealed (in which order this is to be done is subject to

Exercise 3-17). If the authentication for the messages used for the key distribution is valid then the circle of suspects is limited to the key distribution authorities and both participants. The participants should make sure they have the same keys by using a *testing protocol for key equality* right after the key exchange. This can be done by each one sending encrypted random numbers and returning them in plaintext to the partner (This little crypto protocol is of course only secure if the symmetric concealment used can withstand at least one chosen ciphertext-plaintext attack (cp. §3.1.3.2) and if an altering attacker with knowledge of the inconsistent keys positioned between the participants can be excluded. Additionally you need to prevent the “mirror attack” from Exercise 3-22 part a) by having one participant sending the random numbers and waiting for the response before the other one does so. Finally you can not use the Vernam cipher with this protocol! Because then the attacker would gain the exact region of the key that is being tested for equality. In short: We need a better testing protocol or a very good block cipher. The first one is shown right next).

First of all we will look at a *wrong* solution. However this will lead us in the right direction:

Within the network it is known how to create checksums over strings (namely parity bits, Cyclic Redundancy Check (CRC), ...). The participants create such a checksum and one of them sends the result to the other one. If the checksums are equal so both participants may think that (with a very high probability) they have the same key. So where is the error in the reasoning? Simply said, checksums are made to prevent stupid errors but not for preventing intelligent attacks. For the usual checksums it is easy to find many different values that have the same checksums. So this can be done by the attacker to distribute inconsistent keys. Then the two participants would recognize this only if they are using the key. A clever creation of a checksum would be a process with an outcome the attacker can not predict!

In [BeBE\_92, pg. 30] (with reference to [BeBR\_88]) you will find the following nice idea for an *information theoretically secure testing protocol for key equality*: The participants pick randomly (and independently from the previous pick) several times one half of all key bits and tell each other the sum mod 2 (If the keys do not match it is discovered with a probability of  $\frac{1}{2}$ ). To prevent the wiretapping attacker from knowing anything about the forthcoming keys one bit of the sum mod 2 is removed (for example the last one). The key is shortened one bit for every iteration. This is no problem because you only need a few iterations: After  $n$  steps the probability that the result is  $1 - 2^{-n}$  and for  $n > 50$  sufficiently high (For this testing protocol for key equality too you have to exclude an altering attacker with knowledge of the inconsistent keys between the two participants: He could modify the sums sent to make them look consistent to the other participant).

*One key distribution authority:* If only symmetric authentication is used for the key exchange (so neither digital signatures nor unconditional secure pseudo-signatures;

## B Answers

cp. §3.9.3) then from each participant's view the other participant or the key distribution authority is lying (this has been proven for a long time within the context of the Byzantine Agreement protocols [LaSP\_82]). Either the participants are giving up or they pick, if possible, another key exchange authority.

*Several key distribution authorities in sequence:* If the two participants detect inconsistencies in their keys, then the key distribution authorities should check the consistency with the same method as well and, if possible, circumvent attacking key authorities. This is equal to the changing of the key exchange authority for "one key exchange authority".

*Several key exchange authorities in parallel:* If the participants detect inconsistencies in their keys' sum they should repeat the testing protocol for equality for every single one of the added keys. All inconsistent ones are omitted from the sum. If there are not enough keys left to be added the participants should use additional key distribution authorities or even abstain from communication under these circumstances. Otherwise an altering attacker between the participants can prevent the inclusion of keys into the sum that are unknown to him.

f) Ideas (maybe incomplete; further ones are welcomed):

- a) Keys contain a part representing their expiration time. Then the participants only need some correctly working clocks for their security. Disadvantage: If a key is compromised before its expiration, the usage of it can not be stopped.

A related solution would be to limit the amount of usages of the key. This only makes sense for symmetric cryptographic systems where both participants can count. Of course they have to keep permanently track of the keys expired.

- b) Explicit revoking of keys - either by the key owner or the key's certificate authority. This can be realized by a digitally signed message that contains, along with the revoked public key or certificate, the exact point of time of revoke and the revoker's identity (If the key owner is the revoker this authentication is the last action undertaken with the associated secret key).

Of course, explicit revoking of keys only works with participants that can not be cut off by an altering attacker. This additional restriction in contrast to 1. is the price for higher flexibility.

However there are protocols that enable participants to detect their isolation very fast: They send each other messages after a specific number of time units previously agreed upon. All messages are numbered and authenticated (including the enumeration). If there have not been any messages for some time or if some of them are missing, the participants can conclude that somebody tried or is trying, at least temporarily, to isolate them.

- c) We combine 1. and 2. to have the advantages of both worlds.

- d) Prior to the usage of the key, the owner is asked if it is still valid. Disadvantage: If the participant is not always online, large delays may occur if the answer is being waited for without a timeout.
- e) A variation of 4. by waiting only a relatively short amount of time. Since an altering attacker can suppress the answer for a short time unnoticed, this should be combined with variant 3.
- g) What needs to be taken care of while exchanging public keys?  
 → The public keys need to be *authentic*, therefore the recipient needs to be able to verify the authenticity.

With whom can the task of a trustworthy public key register be distributed?

→ The functionality of the trustworthy public key register could be distributed among some of the other *participants*.

When does the key exchange take place?

→ Instead of letting public key registers (one or more; cp. part c) certify the association between your identity and your public key by its digital signatures the participant certifies this association by all participants he knows (and with whom he can safely communicate, for example outside the system to be secured). Those certificates are passed along with the public key. The recipient of such a publicly certified key now decides if he wants to trust its authenticity. For this you could use certificates with known and authentic testing keys. To use those other certificates you need certificates for their testing keys and so on. Usually a public key  $t$  becomes more trustworthy to participant  $T$  the more participants (called *introducer* in [Zimm\_93]) certified key  $t$  in a manner participant  $T$  can verify that. On the other hand the certificate will be more trustworthy to  $T$  the less certificates are needed by  $T$  for its authentication.

What is to be done if everyone is not trusting everybody in the same way?

→ Instead of considering all introducers equally trustworthy you can assign them individual probabilities for their trustworthiness. Consequently you can also calculate a probability for trustworthiness for every certificate chain as well as for every multiply certified public key. The participant can now decide if the probability of trustworthiness is sufficient for the application.

What is to be done if a participant discovers that his private key is known to somebody else?

→ If there is no central hierarchical key distribution then there is no central hierarchical list of all invalid keys. So the poor participant now needs to send his signed message “My key is invalid from now on” to all other participants. If legal stuff comes into play the receiving of this *key-revocation-message* needs to be acknowledged by all with date and time (An obvious but wrong solution would be: Everybody using a key must ask the owner if the key is still valid. Unfortunately

## B Answers

now everybody who knows the secret key can give the correctly signed answer: “Of course my key is still valid. I guard my secrets!!!”).

What should a participant do if his secret key was destroyed?

→ Besides the creation of a key pair and the publication of the public one he should tell everybody not to use the old key for new messages. To prevent that not everybody is able to do so, this message should be signed - optimally with the key that is just considered as destroyed. So right after the generation of a key pair the message “Please do not use my key  $t$  anymore!” should be signed as first. This message does not need to be kept as secret as the secret key of which mostly no copies exist. But this message can be stored with several copies. Often it is useful to distribute it with a threshold scheme (cp. §3.9.6) among several other friends.

What is to be done if a participant discovers that he certified a wrong association Participant  $\leftrightarrow$  Public Key?

→ The participant revokes this certificate with a signed message. Regarding the distribution of this *Certificate-Revocation-Message* the same as for the revealing of the secret key (cp. above) applies.

How to combine the technique developed in this exercise part with the one from exercise part c)?

→ This key exchange method can be easily combined with the methods from c) and d). Central hierarchical key registers and anarchical key distribution à la PGP/GPG support each other. The safety is higher than with only one system.

See p. #466

See p. #471

[Exercise at page 415.](#)

### 3-7 What kind of Trust is necessary for which function of a TrustCenter?

Blind trust for an entity is always necessary if it can make observing attacks. If an entity can only make altering attacks you can shape the procedures in order to alert the attacked participant. Conversely, if the participant does not recognize an attack, then he can justify his trust.

- *Key generation* requires blind trust since the generator can always store a copy locally.
- *Key certification* only requires checkable trust since with a wrong key certification the affected participant would see it during the first conflict. With corresponding procedures (cp. §3.1.1.2) he can even prove this attack.
- *Directory services* only require checkable trust since the providing of false keys can be proven if answers to requests to the directory service are digitally signed.

- *Locking services* only require checkable trust (cp. directory services).
- *Timestamp services* only require checkable trust if a suitably distributed log file exists, since it prevents the service from back-dating (cp. [BaHS\_92, HaSt\_91]).

These ruminations allow the conclusion that if security functions are bundled into one entity you should only bundle key certification, directory service, locking service, and timestamp services but never the key generation (cp. Exercise 3-2,3-3 part a).

[Exercise at page 416.](#)

### 3-8 Hybrid encryption

Case 1:

- Generation of a key  $k$  with a fast symmetric encryption system like AES. Encryption of  $k$  with the public cipher key  $c_E$  of the recipient  $E$ . Encryption of the file with  $k$ . Sending:  $c_E(k), k(file)$ .

Receiver decrypts the first part with decipher key  $d_E$  and gains  $k$ . He uses  $k$  to decrypt the file.

- Generation of a key  $k$  with a fast symmetric encryption system that conducts concealment as well as authentication like DES running in PCBC mode (cp. §3.8.2.5; if you like to use two pure cryptographic systems you must replace  $k$  with  $k_1, k_2$  and continue as seen in Exercise 3-17. Signing of  $k$  (maybe with additional date and time, etc..) with your own secret signing key  $s_S$  (S as in Sender) and then encryption of the signed  $k$  with the public cipher key  $c_E$  of the recipient  $E$  (to conceal first and then sign would work too). Encryption and authentication of the file with  $k$ . Transmission:  $c_E(k, s_S(k)), k(file)$  (If the symmetric encryption system is less complex and if the signature  $s_S(k)$  is not concealed with it then the message format  $c_E(k), k(s_S(k), file)$  can be better.).

Receiver  $E$  decrypts the first part with his decipher key  $d_E$  and gains the signed  $k$  and checks the signature with the testing key  $t_S$ . If the signature is valid  $E$  uses the key  $k$  to decrypt the file and to check the authenticity (For pure cryptographic systems: Let  $k_1$  be the key for a symmetric encryption system,  $k_2$  key for a symmetric authentication system. By replacing  $k$  with  $k_1, k_2$  you send:  $c_E(k_1, k_2, s_S(k_1, k_2)), k_1(file, k_2(file))$ . The authentication of  $k_1$  is not necessary:  $c_E(k_1, k_2, s_S(k_1)), k_1(file, k_2(file))$  will do fine too.)

- Like b) but the symmetric cryptographic system only needs to perform the authentication.

Case 2:

Here  $S$  can perform the same as in case 1 therefore creating a key  $k$  for every file, encrypt it asymmetrically and so on.  $E$  must proceed as with case 1, too. Alternatively  $S$  can

## B Answers

reuse the key  $k$  created for the first file transmission - and tell  $E$  about this. This can save a lot of complexity: No additional key generation, key distribution, key transmission, and key decryption - and for b) and c) no key signing, no signature transmission and no signature validation.

Case 3:

Here  $E$  can perform as with case 1, therefore creating a key  $k$  for every file, encrypt it asymmetrically for  $S$  and so on. If  $E$  wants to reuse key  $k$  for the first file transmission as  $S$  did in case 2 you must be cautious:  $E$  did not create  $k$  on its own and needs to make sure that  $k$  is only known to the intended recipient of his file. For b) and c) it is assured by the digital signature from  $S$ . For a)  $E$  can not be sure who has generated  $k$  since here  $k$  is not authenticated at all, especially not by  $S$ . If at a)  $E$  would not create a new symmetric key  $k'$  but use the old  $k$  an attacker  $A$  that has generated the original  $k$  and sent the first file to  $E$  could intercept the message  $k(\text{file from } E \text{ addressed to } S)$  and decrypt it with  $k$ .

And what do we learn? A protocol that creates a random key can not be used (at least not without further deliberation) with an *existing* key. This is according to the fundamental proposition of cryptographics: All keys and initial values should be picked randomly and independently from each other - unless they need to be picked in dependence, or the picking is thoroughly analyzed and includes a proof of security.

[Exercise at page 417.](#)

### 3-9 Situations from practice (for systematics of cryptographic systems)

- a) A *symmetrical authentication system* is sufficient to spot alterations at each other trusting partners. A key exchange between old friends is no problem. The system is neither very security critical nor time critical. To keep rental options and preferences of the crypto folk secret from curious renters that are (or their friends) employed as communication technicians you can additionally use symmetric concealment. So picking a cryptographic system for the Pfitzmann family is an agony for choice.
- b) *Symmetric encryption system*, direct key exchange, security critical (cryptographers!), time is uncritical. David will use a one-time pad or a pseudo one-time pad - if it is only about stealing ideas. If David is also interested in having his data transmitted unaltered to the trustworthy cryptographer, then an additional authentication is necessary.

Since David has probably already met all cryptographers that are trustworthy to him a direct key exchange should be no problem. But the question arises if David wants to encrypt his messages for each cryptographer separately - which would be necessary with canonical key distribution (every cryptographer with another key). If David wants to occasionally send ideas to a certain group then he could give everybody of the group the same key - this would save quite some encryption

effort. In terms of confidentiality this would not be a problem since the plaintext would be known to everybody from the group. However if David wants to exclude somebody from the group - then he must send everybody else from the group a new key. Such a group key is also problematic in terms of authentication. Because now everybody could authenticate the new idea with the symmetric authentication system and therefore tarnish David’s reputation of having new and good ideas. With pairwise key knowledge this could not happen.

- c) A *symmetric authentication* is sufficient with actually no need for a key exchange. But time and space are very critical. You could use DES (running in CBCauth mode cp. §3.8.2.2).
- d) The computer freak should use a *digital signature system*. The key exchange has to take place via a central authority or by a phone book to make the seller know about the right testing key of the buyer. And to make sure that the association buyer → testing key is legally binding. Since the cryptographic system is time uncritical but security critical the computer freak should use GMR.
- e) *encryption systems* for keeping business secrets (maybe even *authentication* cp. below), but symmetrical is sufficient because it is not a legally binding contract). The key exchange should run over a public key register. So because of the key exchange an asymmetric encryption system is needed, like RSA (or a hybrid, cp. §3.1.4, like RSA combined with DES). If the request is done via FAX, with 9600 bit/s or ISDN with 64 kbit/s, the application is not time critical. If the company uses broadband communication then the usage of a hybrid concealment system is necessary (cp. §3.1.4). The security of RSA and DES should be way sufficient for this not very security critical application.

If authentication is not used at all, the following attack can circumvent the security of business secrets even if a perfect encryption system is used: To gain knowledge about the prices of the competitors the attacker just sends them an according request (including the key for the encryption system). If only network addresses are used as “sender” and therefore as authentication, the attacker just has to intercept the reply message to make the real recipient not suspicious. If the sender is generated by the network the attacker has to adapt it to the connection of the attacked one.

For the reply you will need additional asymmetric authentication besides the concealment to make the offer legally binding (cp. d)).

[Exercise at page 417.](#)

### 3-10 Vernam cipher

- a) The key 00111001 may be exchanged. The plaintext 10001110 is supposed to be encrypted. Using addition modulo 2 the encrypter receives the ciphertext 10110111. By addition of the key and ciphertext modulo 2 the encrypter receives the plaintext. Written down as a computation it looks like this:

## B Answers

<i>encrypter:</i>		<i>decrypter:</i>	
plaintext	10001110	ciphertext	10110111
⊕ key	00111001	⊕ key	00111001
ciphertext	10110111	plaintext	10001110

- b) Let plaintext  $x$ , ciphertext  $S$ , and key  $k$  be elements of the group. By using the definition of the Vernam cipher the following is valid: The encrypter calculates  $x + k =: S$ . The decrypter calculates  $S - k =: x$  (addition of the inverse element is written as subtraction here). If an attacker finds out  $S$  then for each plaintext  $x'$  exactly one key  $k'$  exists, because in each group the equation  $x' + k' = S$  can be solved for  $k'$ :  $k' = -x' + S$ . Since all keys have the same probability from the attacker's point of view, the attacker will not get to know anything about the plaintext when he knows the ciphertext  $S$ .

Annotations:

- 1) The proof uses no commutativity, thus it is valid for *arbitrary* groups. Check if your proof possibly is valid only for Abelian groups.
- 2) The proof ignores that according to the communication protocol, the attacker gets to know something about the plaintext: usually its length, at least his maximum length. The first can be avoided by padding messages to standard length and/or by adding meaningless messages, cp. §5.4.3. The best is to transmit a permanent stream of characters to ensure that an attacker can not recognize when no messages are being transmitted, cp. §5.2.1.1.
- c) The result is unambiguous only if the coding operation is assumed as agreed. (This agreement can be considered as part of the definition of the encryption system or of the key.) If addition modulo 2 is agreed as operation, the following plaintext arises:

$$\begin{array}{rcl} \text{ciphertext} & 01011101 \\ \oplus \text{key} & 10011011 \\ \hline \text{plaintext} & 11000110 \end{array}$$

If addition modulo 16 is agreed, however, as an operation, another plaintext arises:

$$\begin{array}{rcl} \text{ciphertext} & 01011101 \\ -(16) \text{ key} & 10011011 \\ \hline \text{plaintext} & 11000010 \end{array}$$

if it is calculated character wise (4-bit-wise), otherwise the following arises:

$$\begin{array}{rcl} \text{ciphertext} & 10111010 \\ -(16) \text{ key} & 11011001 \\ \hline \text{plaintext} & 11100001 \end{array}$$

written using our convention: 10000111.

- d) Calculating modulo 2: Since each bit is en- and decrypted, it is valid: Invert the third bit from the right, i.e., transmit 010111010101110101011001.

(Modulo each number causes a change of the ciphertext from  $S$  to  $S'$  and a change of the plaintext about the same difference  $d := S' - S$ , because  $S$  is encrypted as  $x' = S' - k = (S + d) - k = (S - k) + d = x + d$ .)

The following calculations assume that it is written character-wise from the left to the right.

Calculation modulo 4: Because there are always two groups of bits being en- resp. decrypted, it has to be decided what has to happen with the third last and fourth-last bit. If 01 modulo 4 is added to them, the third last bit is inverted for sure, but the fourth-last bit too, provided that an add carry exists. If the attacker does not know the third last plaintext bit, he can not prevent the change of the fourth-last plaintext bit deterministically. Because nothing was said in the exercise about the fourth-last bit, one can accept the following as a solution: The attacker transmits 010111010101110101010001. Should the change of the fourth-last bit be avoided with unknown third last plaintext bit deterministically, the exercise is unsolvable for modulo 4.

Calculation modulo 8: Because there are always 3-groups of bits being en- resp. decrypted, it has to be decided what has to happen with the three last bits. If 100 modulo 8 is added to them, the third last bit is inverted for sure. Thus the attacker transmits 010111010101110101010001.

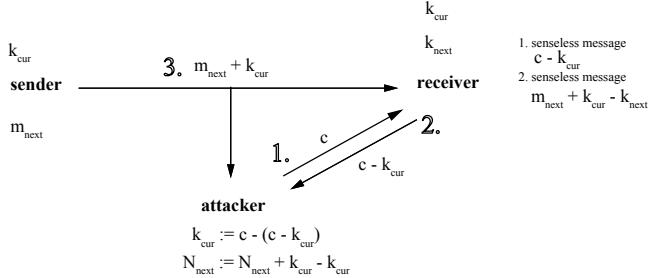
Calculation modulo 16: Because there are always 4-groups of bits being en- resp. decrypted, it has to be decided what has to happen with the four last bits. If 0100 modulo 16 is added to them, the third last bit is inverted for sure, but the fourth-last too, provided that an add carry exists. If the attacker does not know the third last plaintext bit, he can not prevent the change of the fourth-last plaintext bit deterministically. Because nothing was said in the exercise about the fourth-last bit, one can accept the following as a solution: The attacker transmits 010111010101110101010001. Should the change of the fourth-last bit be avoided with unknown third last plaintext bit deterministically, the exercise is also unsolvable for modulo 16.

- e) Because every section of the key is used by the attacked person only once and all sections are generated stochastically independent of each other, an attacker can learn nothing from earlier reactions about future reactions. (In principle such an attack exists of course, but it has just no use). The same is true for active (not adaptive) attacks.

It is has been observed, however, that both sorts of active attacks in regards to *messages* can be successful: By an active attack (chosen ciphertext-plaintext-attack) on the recipient an attacker finds out the value of the just current key section  $k_{cur}$ . Then he is able to decrypt the next messages  $N_{next}$  encrypted by the sender. What we learn is: The partner who decrypts is not allowed to output decrypted messages and make an active attack possible in this way. It is allowed

## B Answers

to deviate from this restriction, if the received message is authenticated correctly. (For obvious reasons with an information-theoretically secure authentication code.)



An additional annotation: Vernam cipher keys (or at least key sections) must be applied by one partner only in order to encrypt and from the other partner only in order to decrypt. It is in no case enough that two partners have a common key and always the one who wants to send just a message to the other, uses for this as a “key” the next key section. Otherwise it can happen that both partners send messages at the same time, so that an attacker finds out the difference of their plaintexts.

- f) Since this exercise is about properties of the en- and decryption function itself, i.e., independent of single keys, the abbreviated notation introduced in §3.1.1.1 is chosen clumsily (The functions themselves are just not present in this definition), so that we go over to the more detailed notation likewise described in the mentioned section. First the definition (1) is written down in the more detailed notation:

Definition for informational theoretically security in the case that all keys are chosen with same probability:

$$\forall S \in S \exists const \in \mathbb{N} \quad \forall x \in X : |\{k \in K | enc(k, x) = S\}| = const \quad (1)$$

We show: At least if, as with all Vernam cipher, key space = plaintext space = ciphertext space, it is *necessary* and *sufficient*: The function *enc* has to run through the whole co-domain in case of retaining one of both input parameters and varying the other.

- 1 (retaining  $k$ ) In the case of constant  $k$  each ciphertext must be encrypted uniquely, thus occurring only for one plaintext (i.e.,  $enc(k, \bullet)$  is injective). Since there are as many ciphertexts as plaintexts, it is equivalent that each ciphertext occurs at least once (i.e.,  $dec(k, \bullet)$  is surjective).
- 2a (retaining  $x$ , passing through is *necessary*) From definition (1) follows that behind each actually appearing ciphertext  $S$  each plaintext  $x$  is possible. It follows from 1. that each value  $S$  of the ciphertext space occurs actually. This is formally written in the following way:  $\forall S \forall x \exists k : enc(k, x) = S$ . This is equivalent to  $\forall x \forall S \exists k : enc(k, x) = S$ , i.e., that all  $S$  can occur in case of retained  $x$ .

2b (retaining x, passing through is *sufficient*) Do all S occur the other way around when retaining x security follows: If S is given, thus for each  $x$  exists a  $k$ , so that  $\text{enc}(k, x) = S$ . In order to show that all  $x$  have the same probability, too, we do need that again because of the same size of the spaces  $\text{enc}(k, x) = S$  is only valid for one  $k$ .

For the binary case of the Vernam cipher we search 2x2 functional boards, in which in every line and column stands a 0 and 1. There are exactly two which represent the function XOR and NOT XOR. (That there are exactly two, one can see for example in this way: If one specifies the value for one combination of both input parameters the function is clear, because the result of the function must change each time when the parameter values are varied. Since one has 2 possibilities for the first function value it follows that there must be two functions.)

$S = k(x)$	x	0	1
k	0	y	$\bar{y}$
	1	$\bar{y}$	y

$\bar{y}$  may be the inverse to y. For  $y=0$ : XOR

$S = k(x)$	x	0	1
k	0	0	1
	1	1	0

$S = k(x)$	x	0	1
k	0	1	0
	1	0	1

For  $y=1$ : NOT XOR

g) Dear younger friend!

Firstly, if it is not intuitively clear to you, you can reread in historic sources [Sha1\_49], that the natural language consists of *redundancy* of more than the half and is therefore no replacement for a randomly chosen key, i.e., a key of maximal *entropy*. Therefore, one can understand the sense of longer ciphertexts, if natural language is encrypted in this way, without determining book, side, and line. One can in fact understand even the sense of the book passage used for encryption and determine it if necessary. (An exact procedure to the entire breaking by means of a pure ciphertext-plaintext-attack is outlined in [Hors\_85, page 107 f.]. Entire breaking by means of a plaintext-ciphertext-attack is essentially easier.)

Secondly, if you have not recently done it, you should inform yourself about the current and future expected computing power and storage capacity of a PC. Then you will also fear that it will soon be possible to store all the world's literature on some optical disks and then decrypt automatically each ciphertext, that has been encrypted using your procedure, that contains mechanically recognizable redundancy by entire search in some hours. (Very defensive explanation for plausibility: World literature may have  $10^8$  volumes that average 1 MByte. Thus  $10^8/600$  disks are needed for 5,25"-optical disks with 600 MB each that are usual for 1990, thus circa 160 000 optical disks. To examine them from each bit position needs  $10^8 * 2^{20} * 8 = 8 * 10^{14}$  steps, where each step might last  $10^{-6}$  s. Thus entire rummage would last  $8 * 10^8$  s, thus 92.593 days when using a computer with today's performance. If one starts with the most renown works of the world, the search will end a lot faster.) Finally a small comfort: You are not the first one who creeps in such a mistake. Besides, you had the luck to find a friend by publishing your procedure who points out your mistake to you. Many of your colleagues who want

## B Answers

to learn either nothing more or whose contractors want to protect copyrights keep their procedures secret. You can imagine what often happens then.

- h) No, because one can think out an arbitrary different plaintext of the same length and then calculate the key so that he fits to this plaintext and the existing ciphertext. Thus the prosecution authority gets no meaningful information.
- i) One exchanged storage medium is used for 2-way communication, so that for every direction only half of the storage capacity can be used. Then arises:

	text communication (typing speed 40bit/s)	speech communication (normal ISDN coding: 64kbit/s)	high-resolution full-video communication (140Mbit/s)
3.5" floppy (1.4 MByte)	1.62 days	87.5 s	0.04 s
CD-R (700 MByte)	850 days	12.7h	21 s
DVD one-sided (4.7 GByte)	5440 days	3.4 days	134 s
Blueray ROM (50 GByte)	57870 days	36.2 days	1429 s

Exchange of a 1.4 MByte suffices for text communication for most applications “for life” (since you will only like a few so much that you type 1.62 days non-stop for them). Correspondingly, exchange of optical 700 MByte of disks suffices for many linguistic communication-applications (even if many people talk more perseveringly than typing). For uncompressed high-resolution full-video communication the current mass storages are still too small, but are absolutely useful for compressed full-video communication of low dissolution like intended for ISDN (an additional channel of 64 kbit/s for the picture): To talk and watch one’s telephone partner information-theoretically secure with the help of a DVD for 3.4 days, might also reach here “for life”. For volatile acquaintances suffices a CD-R newly – and for intense tele-relations, a Blueray ROM will suffice in the future.

Another question is whether the reading access to media goes quickly enough to be able to access the key in real time. For text communication and linguistic communication no problem. For high-resolution full-video communication with 140 Mbit/s for all media this is not possibly up to now – another reason, why lossy compression is used for full-video signals. The first obtains app. the factor 4, with the latter it depends on the fact which quality loss one would like to accept. If you have still unpacked your pocket calculator, you can calculate the numbers

for high-resolution full-video communication once more with the typical values of 34 Mbit/s for lossless and 2 Mbit/s for lossy compression. What arises? DVD is absolutely suitable for high-resolution full-video communication – for the storage of the one-time pad as well as for the storage of full-video signal. No miracle: key, ciphertext, and plaintext are of the same length – and the medium was designed for the latter.

See p. #50

See p. #74

See p. #74

[Exercise at page 418.](#)

### 3-11 Symmetrically encrypted Message Digest = MAC ?

- a) The recipe is completely insecure, as the example mentioned in the exercise shows:

From the intercepted message everybody can calculate the check digit by means of the publicly known procedure. Everybody can determine the current key section from the check digit and its intercepted encryption with the Vernam cipher. The check digit can be calculated by means of the publicly known procedure for each message (which the attacker can freely select) and then be encrypted by means of the current key section to an “authentic” MAC.

A selective break is reached by this passive attack which is not acceptable, as already mentioned in §3.1.3.1.

- b) For the usual collision-resistant hash-functions, this is a secure construction according to current knowledge. But these hash-functions are not only collision-resistant, but they additionally have a trait which is essential for the security of the construction applied with TLS:

Their functional value says *nothing* about the input.

To make clear why this is essential, we assume the contrary: The collision-resistant hash-function may output 80 bits of its input as a part of its functional value. If the functional values as well as the inputs are long enough, such a collision-resistant hash-function can be a super collision-resistant one. However, used with the construction applied with TLS, it shortens the key length by about 80 bits if the key “unfortunately” stands at the corresponding place of the input. If a key length of 128 bits is used which usually is more than enough, then an attacker has to try at most  $2^{48}$  times to determine the key. Today nearly every relevant attacker is able to do this.

- c) Yes: Hash ( $<key>, <number\ of\ byte>, <byte>$ ) and transfer the functional values (if necessary together with  $<number\ of\ byte>$ , if the recipient does not count bytes or the communication medium is unreliable). Now the recipient only has to try out the values of the byte, by hashing ( $<key>, <number\ of\ byte>, <value\ of\ byte>$

## B Answers

and to compare this with the transmitted functional value to decrypt the functional value to the plaintext byte. Except for the sender and recipient nobody can do this, because nobody knows the key. The numbering of the bytes prevents that same bytes result in same functional values, a synchronous stream cipher is produced this way, cp. §3.8.2. In the example it is supposed that a byte is a meaningful compromise between the number of trials in case of decryption by the recipient and the message-expansion by essentially more values of the hash-function (typically 160 bits).

[Exercise at page 419.](#)

### 3-12 Authentication codes

- a) The key 00111001 is exchanged. The plaintext  $HTTT$  is to be secured. Using authentication code  $H, 0 T, 1 T, 0 T, 1$  is generated as a “ciphertext” and transmitted.
- b) If an attacker intercepts  $H$  or  $T$  and the corresponding MAC, two (of four) ciphertexts remain for him indistinguishable for good, because  $H$  resp.  $T$  occur exactly twice in each column of the table in which they occur. Exactly these two values distinguish, whether the MAC has to have the value 0 or 1 for the other value of “tossing the coin”. (This claim can be easily verified for this simple authentication code by consideration of all possibilities.) Therefore the attacker has no better strategy than guessing, what will lead to the fact that the recipient will recognize the falsification with a probability of 0.5. An easier representation: If  $k = k_1 k_2$  then is  $k(H) = k_1$  and  $k(T) = k_2$ . (This is illustrated in the following picture taken from the exercise, where  $k_1$  is drawn bold.) Thus the authentication code can be pictured as shown in the figure on the right: Here the value of the MAC is represented in dependency of the key and text. Obviously one does not get to know anything about the MAC for  $T$  through the current MAC for  $H$ .

		Text with MAC						Text		
		H, <b>0</b>	H, <b>1</b>	T, 0	T, 1			0, 0	H	T
keys	0, 0	H	-	T	-	keys	0, 0	<b>0</b>	0	
	0, 1	H	-	-	T		0, 1	<b>0</b>	1	
	1, 0	-	H	T	-		1, 0	<b>1</b>	0	
	1, 1	-	H	-	T		1, 1	<b>1</b>	1	

- c) No, because the recipient uses his key sections in a fixed order.
- d) *Example:* Let the key 00111001 be exchanged. The plaintext  $HTTT$  is to be secured. Then the ciphertext 00110100 is generated and transmitted.  
In terms of authentication the argumentation above applies analogously: For both key values that are possible after being observed by an attacker he has to take different ciphertexts in case of the intended change of plaintexts. Thus again he

has no better chance for success than guessing right with a probability of 0.5. If an attacker executes a modifying attack, however, he can recognize in every case in the reaction of the attacked person (accepts faked message or not), which of both possible key values that are still possible after his first observation are present, and thereby he is able to determine exactly the key value. From this value and the observed ciphertext arises the clear text. This is explained with an example: The attacker intercepts the ciphertext 00. Then he guesses from the two possibilities the key value 00, thus the plaintext  $H$ . He changes the plaintext to  $T$  by forwarding the ciphertext 10. If the recipient accepts this, his guess was right, thus the original plaintext was really  $H$ . If the recipient does not accept, the value of the key was 01, and therewith the original plaintext is  $T$ .

- e) An additional key bit is used. If it is 0 everything stays the same. If it is 1 all  $T$  are replaced by  $H$  in the table and vice versa. Even with the optimal attack over the key regarding to acquisition of information described above the attacker does not know the value of the additional key bit. This key bit acts as a Vernam cipher so that the encryption is still perfect even after this attack.

Now, unfortunately, the combined system is not more efficient than an authentication code and Vernam cipher, provided that only the profit bit is encoded first and afterwards authenticated. If one does it the other way round, one needs instead of 3 key bits 4.

- f) It is to be shown that for each possible  $MAC'$  (i.e.,  $MAC' \in K$ ) there is exactly one value  $(a, b)$  compatible to the observation  $(m, MAC)$ . The system of equations

$$\begin{aligned} MAC &= a + b * m \\ MAC' &= a + b * m' \end{aligned}$$

with the unknown quantities  $a, b$  has exactly one solution if and only if  $m \neq m'$ , but  $m \neq m'$  is exactly the aim of the attacker.

For those who do not remember linear algebra that exactly, here is a somewhat more precise explanation: A system of equations  $C = M * X$  with the matrix  $M$  and the vectors  $C$  and  $X$  is exactly then uniquely solvable if  $M$  is regular.  $M$  is regular if and only if a triangular matrix can be created by the rows and columns of  $M$  (i.e., all diagonal elements are  $\neq 0$  and either all values beneath the diagonal or all values above the diagonal are 0). This is shown for our matrix

$$\begin{pmatrix} 1 & m \\ 1 & m' \end{pmatrix}$$

shortly: Subtraction of the first from the second row results:

$$\begin{pmatrix} 1 & m \\ 0 & m' - m \end{pmatrix}$$

## B Answers

This is exactly then a diagonal matrix if  $m \neq m'$ .

To show a subtle error, another “proof” is shown in the following:

$$MAC := a + b * m \Leftrightarrow a := MAC - b * m$$

Thus  $b$  can be chosen arbitrarily, because the equation can be fulfilled by  $a$ . Thus the attacker gets to know nothing about  $b$ . In case of authentication of another message  $m'$ , he ought to know  $b$  to be able to correct  $MAC$  to  $MAC'$  {because  $b * m$  changes its value arbitrarily and is distributed uniformly in case of variation of  $m$ }.

Without the addition of the parentheses in the “proof”, the authentication code  $MAC := a + b * m^2$  would be a nice counterexample: Because the “proof” applies here, too, but here the message  $m$  can be replaced by  $-m$  without having to change the  $MAC$ .

[Exercise at page 420.](#)

### 3-13 Mathematical secrets

- a) It is not completely clear because the prime numbers are not chosen with the same probability any more, but such that primes after long gaps in the sequence of primes are chosen with higher probability and such that if prime numbers are only two larger than another prime number, they are only chosen if they were chosen as random number at the beginning.

For specialists: At least under Cramer’s assumption the procedure is secure nevertheless: This assumption (which I know from [GoKi\_86]) implies  $\pi(x * \log^2 x) - \pi(x) > 0$ . This means that one gets the prime numbers after longest gaps only up to a factor of  $\log^2 x \approx l^2$  more frequently than the above mentioned prime twins. For the products  $n$  probability differences up to  $l^4$  arise. If there would be a successful factorization algorithm for this distribution it would be less successful only with a factor of at most  $l^4$  for a correct distribution. Therefore its success probability decreases not faster than each polynomial, which contradicts the factorization assumption. (Without Cramer’s assumption one would come artificially to the same result, if one proceeds the search by adding 2 in each step not arbitrarily long, but uses a criterion for stopping, e.g., after  $l^2$  steps.)

- b) One can expect that both prime numbers are very close together, thus both are about  $\sqrt{n}$ . Therefore, one can search the integers around  $\sqrt{n}$  (i.e., divide  $n$  by each of them.)

More exactly: In most cases e.g.,  $q \leq p + 16l$  applies (prime number theorem with some leeway, because the distance to the next prime number averages  $l * \ln(2)$ ). The particularly beautiful number 16 was chosen to alleviate the following calculations).

In these cases is  $p^2 < n < p(p + 16l) < (p + 8l)^2$ , i.e.,  $p < \sqrt{n} < p + 8l$ , thus  $\sqrt{n} - 8l < p < \sqrt{n}$ . Thus one only has to examine this  $8l$  numbers and has them factored with significant, even huge probability.

c)  $19 = (10011)_2$ . mod 101 we have:

$$3^{(1)_2} = 3; 3^{(10)_2} = 3^2 = 9; 3^{(100)_2} = 9^2 = 81 \equiv -20; 3^{(1001)_2} = (-20)^2 * 3 = 1200 \equiv -12;$$

$$3_2^{10011} = (-12)^2 * 3 = 144 * 3 \equiv 43 * 3 \equiv 129 \equiv 28.$$

d)  $77 = 7 * 11$ ;  $\phi(77) = 6 * 10 = 60$  and  $3 \in \mathbb{Z}_{77}^*$ , because  $\gcd(3, 77) = 1$ . Thus Fermat's small theorem says:  $3^{60} \equiv 1 \pmod{77}$ .  
Thus it is valid:  $3^{123} \equiv (3^{60})^2 * 3^3 \equiv 1^2 * 27 \equiv 27$ .

e) Euclid's algorithm:

$$\begin{aligned} 101 &= 4 * 24 + 5; \\ 24 &= 4 * 5 + 4; \\ 5 &= 1 * 4 + 1; \end{aligned}$$

Backwards:

$$\begin{aligned} 1 &= 5 - 1 * 4 && \text{(Now dividing 4 by 5 and substituting 24)} \\ &= 5 - 1 * (24 - 4 * 5) = -24 + 5 * 5 && \text{(now substituting 5 by 24 and 101)} \\ &= -24 + 5 * (101 - 4 * 24) = 5 * 101 - 21 * 24 && \text{(Probe: } 5 * 101 = 505, -21 * 24 = -504\text{)} \end{aligned}$$

Thus:  $(24)^{-1} \equiv -21 \equiv 80$ .

f) There is no inverse for 21 mod 77 since  $\gcd(21, 77) = 7 \neq 1$ .

g) First we apply the extended Euclidean algorithm, in order to search  $u$  and  $v$  with  $u * 7 + v * 11 = 1$ . (Even if one can see this perhaps at first sight:  $-21+22=1$ ):

$$\begin{aligned} 11 &= 1 * 7 + 4; \\ 7 &= 1 * 4 + 3; \\ 4 &= 1 * 3 + 1. \end{aligned}$$

Backwards:

$$\begin{aligned} 1 &= 4 - 1 * 3 \\ &= 4 - 1 * (7 - 1 * 4) = -7 + 2 * 4 \\ &= -7 + 2 * (11 - 1 * 7) = 2 * 11 - 3 * 7. \end{aligned}$$

Thus the “base vectors” for  $p = 7$  and  $q = 11$  are:  $up = -21 \equiv 56, v * q = 22$ . With the formula  $y = up * y_q + vq * y_p$  arises:  $y = -21 * 3 + 22 * 2 = (22 - 21) * 2 - 21 = -19 \equiv 58$ . (Test:  $58 \equiv 2 \pmod{7} \wedge 58 \equiv 3 \pmod{11}$ .)

For  $z$  we can insert into the same equation ( $up$  and  $vq$  have to be calculated only once). We simplify before:  $6 \equiv -1 \pmod{7}$  and  $10 \equiv -1 \pmod{11}$ . Thus:  $z = -21 * (-1) + 22 * (-1) = -1 \equiv 76 \pmod{77}$ .

We could have had it simpler: After formula (\*) from §3.4.1.3 is  $z \equiv -1 \pmod{7} \wedge z \equiv -1 \pmod{11} \Leftrightarrow z \equiv -1 \pmod{77}$ .

## B Answers

h) Since everything is modulo, prime numbers one can apply Euclid-criteria:

$$\left(\frac{13}{19}\right) \equiv 13^{\frac{19-1}{2}} \equiv (-6)^9 \pmod{19}; (-6)^2 \equiv 36 \equiv -2 \Rightarrow (-6)^4 \equiv 4; (-6)^8 \equiv 16 \equiv -3$$

$$(-6)^9 \equiv -3 * (-6) \equiv 18 \equiv -1 : \text{no square.}$$

$$\left(\frac{2}{23}\right) \equiv 2^{11} \equiv 2048 \equiv 1 \pmod{23} : \text{square}$$

$$\left(\frac{-1}{89}\right) \equiv (-1)^{44} \equiv 1 \pmod{89} : \text{square}$$

The simple case  $p \equiv 3 \pmod{4}$  propounds only with  $p=23$ . Thus one is able to extract the root just with our knowledge about the algorithm: A root is

$$w = 2^{\frac{23+1}{4}} = 2^6 = 64 \equiv -5 \equiv 18 \pmod{23} \quad (\text{Probe: } (-5)^2 \equiv 25 \equiv 2)$$

(Thus the other root is +5.)

i) It is  $58 \equiv 2 \pmod{7} \wedge 58 \equiv 3 \pmod{11}$ . For  $p = 7, 11$   $p \equiv 3 \pmod{4}$  is valid, and it is

$$2^{\frac{7-1}{2}} \equiv 8 \equiv 1 \pmod{7} \text{ and } 3^{\frac{11-1}{2}} \equiv 9 * 9 * 3 \equiv (-2) * (-2) * 3 \equiv 1 \pmod{11}.$$

Thus 2 and 3 are squares mod 7 resp. mod 11. Thus we can use the simple algorithm for extracting roots to receive the roots  $w_7 \pmod{7}$  and  $w_{11} \pmod{11}$ :

$$w_7 = 2^{\frac{7+1}{4}} = 2^2 = 4 \pmod{7}.$$

$$w_{11} = 3^{\frac{11+1}{4}} = 3^3 = 27 \equiv 5 \pmod{11}.$$

With the Chinese Remainder Theorem and the already calculated “base vectors” we receive:

$$w = -21 * w_{11} + 22 * w_7 = -21 * 5 + 22 * 4 = (22 - 21) * 4 - 21 = -17 \equiv 60 \pmod{77}.$$

$$(\text{Probe: } (-17)^2 = 289 = 3 * 77 + 58.)$$

If we insert each the “other” root mod 7 or mod 11, we receive the three different roots:

$$\text{e. g.: } w' = -21 * 5 + 22 * (-4) = -105 - 88 = -193 \equiv 38 \pmod{77}.$$

$$(\text{Probe: } 38^2 = 1444 = 18 * 77 + 58.)$$

Instead of proceeding with such calculations, we now take the inverse of both already known roots:

$$w'' = 17$$

$$w''' = 39.$$

j) One knows another, essentially different root of 4, viz 2.

Thus one can receive a factor  $p$  as  $\gcd(2035 + 2, 10379)$ . Euclidean algorithm:

$$10379 = 5 * 2037 + 194$$

$$2037 = 10 * 194 + 97$$

$$194 = 2 * 97 + 0$$

Thus  $p = 97$ . One receives the complete factorization as  $q = 10379 \text{div} 97 = 107$ ; and  $p$  and  $q$  are prime.

(For fans: One starts with the factors, here  $P=97$  and  $q=107$ . The roots of  $4 \bmod p$  and  $q$  are each  $\pm 2$ . To receive one of two essentially different roots mod  $p*q$ , for which  $+2 \bmod p$  and  $-2 \bmod q$  or  $-2 \bmod p$  and  $+2 \bmod q$  is valid, one has to create  $\text{CRA}(+2,-2)$  or  $\text{CRA}(-2,+2)$ .)

- k) Let  $w_p := x^{\frac{p+1}{4}}$  and  $w_q := x^{\frac{q+1}{4}}$ .

Following §3.4.1.8 it is clear that  $x \bmod n$  has 4 roots  $w, w', w'', w'''$ , that can be calculated using  $w_p, w_q$  of  $x$  as follows:

$$\begin{aligned} w &= \text{CRA}(w_p, w_q) \\ w' &= \text{CRA}(w_p, -w_q) \\ w'' &= \text{CRA}(-w_p, w_q) \\ w''' &= \text{CRA}(-w_p, -w_q) \end{aligned}$$

Following §3.4.1.6 is  $w_p \in QR_p$  and  $w_q \in QR_q$ , but  $-w_p \notin QR_p$  and  $-w_q \notin QR_q$ . Following §3.4.1.7 it is valid:

$$x \in QR_n \Leftrightarrow x \in QR_p \wedge x \in QR_q.$$

Thus it is valid, that  $w \in QR_n$  and  $w', w'', w''' \notin QR_n$ .

- l) If the factorization assumption is wrong an attacker would be able to factorize a not negligible part of the modules  $n = p * q$ . For such modulus he can apply Euler's criteria and calculate

$$\left(\frac{x}{p}\right) \equiv x^{\frac{p-1}{2}} \bmod p \text{ and } \left(\frac{x}{q}\right) \equiv x^{\frac{q-1}{2}} \bmod q$$

for  $x$ , for which must be proofed if it is a square or not.  $x$  is a square if  $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = +1$ . Since evaluating of the formulas causes at most cubical effort (in the length of  $n$ ), thus is especially polynomial in the length of  $n$ , an attacker is able to decide for a not eligible part of all modulus if  $x$  is a square or not. This contradicts with the quadratic residuosity assumption.

To practice dealing with the formal definitions of the factorization - and quadratic residuosity assumption, the *not negligible part* is now written down formally:

There is a (probabilistic) polynomial algorithm  $F$ , so that it applies for a polynomial  $Q$ : For each  $L$  exists a  $l \geq L$ , so that it applies: If  $p, q$  are chosen as independent, random prime numbers of length  $l$  and  $n := p * q$

$$W(F(n) = (p; q)) > \frac{1}{Q(l)}.$$

From this algorithm  $F$  we construct another likewise (probabilistic) polynomial algorithm  $R$ .  $R$  calls  $F$ . If  $F$  factorizes, then  $R$  calculates, as just described,  $\left(\frac{x}{p}\right)$

## B Answers

and  $\left(\frac{x}{q}\right)$ . If both are +1,  $R$  puts out “true”, otherwise “false”.  $R$  is successful if and only if  $F$  is successful.

Thus it applies: There is a (probabilistic) polynomial algorithm  $R$ , so that it applies for the above mentioned polynomial  $Q$ : For each  $L$  exists a  $l \geq L$ , so that it applies: If  $p, q$  are chosen as random prime numbers of length  $l$ ,  $N := p * q$ , and  $x$  is coincidentally beneath the cosets with Jacobi-Symbol +1, then it applies:

$$W(R(n, x) = qr(n, x)) > \frac{1}{2} + \frac{1}{Q(l)}.$$

This contradicts with the quadratic residuosity assumption.

[Exercise at page 421.](#)

### 3-14 Which attacks on which cryptosystems are to be expected?

- a) The notary must use a digital signature system. Because he signs arbitrary messages submitted to him, excepting the time stamp (e.g., to appending of date and time to the message), the digital signature system must resist adaptive active attacks. The message format could be as follows:  
|submitted message|, date, time,  $s_{notary}$ (|submitted message|, date, time).
- b) Either symmetric or asymmetric *encryption systems* can be used.

The first must resist at least a known-plaintext-attack, because the attacker can read the plaintext of the observed article in the newspaper soon. If the attacker does sensational things, he can reach at least particularly a plaintext-ciphertext attack, e.g., he could spread the text that he wants to get encrypted, in the city hall. (Both attacks are irrelevant with the asymmetric system, because the attacker can encrypt arbitrary texts on his own.)

Even if the newspaper printed arbitrary senseless articles (perhaps the correspondent accepts box number notifications, too), the encryption system (symmetric or asymmetric) must resist even a chosen-ciphertext-attack, because the attacker can read the plaintext of his sent ciphertext in the newspaper soon.

[Exercise at page 422.](#)

### 3-15 How long must files be protected? Dimensioning of cryptographic systems; reactions to false dimensioning; Publishing public keys of a signature system or of an asymmetric encryption system?

- a) The probability that the cryptographic system will be broken this time must be sufficiently low. If an information-theoretical secure system (with the one-time pad or suitable authentication codes) is used, the following 3 points are irrelevant. Otherwise attention should be paid to:

- The time which is available to the attacker for a cryptoanalysis trivially grows with the length of the time within which the protection should be effective.
- Furthermore, the computation power/EUR probably increases exponentially with a duplication time ca. every year, until technological borders are reached sometimes. This increasing computer power will also be available to the attacker.
- Besides, more efficient algorithms can be discovered in this time. (Because for all information-theoretical secure systems there are no useful lower barriers for the complexity of breaking.)

In both examples one has to oversize the systems so that they are hopefully safe for at least 80 years.

To have a feeling what this means, an example: we assume, that the growing of the computer power would last that long. (However, this might become limited with physical borders already, cp. [Paul\_78, DaPr\_89, page 236 f., page 41] Then it is so, as if the attacker has approximately  $2^{81}$  years with current computer power at a disposal. If one could not break the system faster than by an entire search in the key space, this would have to be chosen too largely around the factor  $2^{81}$ , thus the keys about 81 bits are too long, if they are coded optimally. This would not even be a lot.

It is already different with systems based on factorization. In order to explain the progress on the one hand with the algorithms of factorization and with the computer power on the other hand, two exemplary calculations are shown:

We assume the complexity is and will be

$$L_{1990}(n) = \exp(\sqrt{\ln(n) \ln \ln(n)}), \\ L_{1995}(n) = \exp(1.923 * \sqrt[3]{\ln(n) \ln \ln(n)}),$$

this is the complexity of the best factorization algorithm known in 1990 and 1995, cp. [Schn\_96, page 256].

In 1995 we used  $n$  of length 512 bit and this was justified with the fact that those numbers can not be factorized within one year (it is different today, cp. [Riel\_99]). Thus we should, from the perspective of 1995, suppose  $n^*$  for the future with

$$L(n^*) = 2^{81} * L(n)$$

From the algorithmic point of view of 1990 this means

$$L_{1990}(n^*) = 2^{81} * L_{1990}(n) \\ \Leftrightarrow \exp(\sqrt{\ln(n^* \ln \ln(n^*))}) = 2^{81} * \exp(\sqrt{\ln(n) \ln \ln(n)}) \\ \Leftrightarrow \sqrt{\ln(n^* \ln \ln(n^*))} = 81 * \ln(2) + \sqrt{\ln(n) \ln \ln(n)}.$$

## B Answers

Furthermore we know that  $\ln(n) = 512 * \ln(2) \approx 360$  and  $\ln \ln(n) \approx 6$ , as well as  $\sqrt[3]{360 * 6} = 44$ , thus circa

$$\begin{aligned}\Leftrightarrow \sqrt{\ln(n^* \ln \ln(n^*))} &= 56 + 44 \\ \Leftrightarrow \ln(n^* \ln \ln(n^*)) &= 100^2\end{aligned}$$

A little calculations leads to the statement, that this inequality is complied for  $\ln(n^*) \approx 1440$ . Thus  $\log_2(n^*) = 1400/\ln(2) \approx 2020$ .

One would have to choose numbers approximately 3.95 times longer than before. The expenditure of calculation increases by the factor  $3.95^3 \approx 61$ .

From the algorithmic point of view of 1995 this means:

$$\begin{aligned}L_{1995}(n^*) &= 2^{81} * L_{1995}(n) \\ \Leftrightarrow \exp(1.923 * \sqrt[3]{\ln(n^*) (\ln \ln(n^*))^2}) &= 2^{81} * \exp(1.923 * \sqrt[3]{\ln(n) (\ln \ln(n^*))^2}) \\ \Leftrightarrow 1.923 * \sqrt[3]{\ln(n^*) (\ln \ln(n^*))^2} &= 81 * \ln(2) + 1.923 * \sqrt[3]{\ln(n) (\ln \ln(n^*))^2} \\ \Leftrightarrow \sqrt[3]{\ln(n^*) (\ln \ln(n^*))^2} &= 42,12 * \ln(2) + \sqrt[3]{\ln(n) (\ln \ln(n^*))^2} \\ \Leftrightarrow \sqrt[3]{\ln(n^*) (\ln \ln(n^*))^2} &= 29 + \sqrt[3]{\ln(n) (\ln \ln(n^*))^2}\end{aligned}$$

Furthermore we know that  $\ln(n) = 512 * \ln(2) \approx 360$  and  $(\ln \ln(n))^2 \approx 36$ , as well as  $\sqrt[3]{360 * 36} \approx 23$ , thus circa:

$$\begin{aligned}\Leftrightarrow \sqrt[3]{\ln(n^*) (\ln \ln(n^*))^2} &= 52 \\ \Leftrightarrow \ln(n^*) (\ln \ln(n^*))^2 &= 52^3\end{aligned}$$

A little calculator acrobatics results that this inequality is complied for  $\ln(n^*) \approx 2400$ . Thus  $\log_2(n^*) \approx 2400/\ln(2) \approx 3462$ . One would have to choose numbers approximately 6.76 times longer than before. The expenditure of calculation increases by the factor  $6.76^3 \approx 309$ .

We see: the algorithmical progress within 5 years has “as much” influence on the necessary length of the numbers as the increase of computer power expected within 80 years. Both force an extension of approximately 1.500 bits for the considered period.

(However, fortunately there are many applications where signatures must be valid only shortly, e.g., in a digital payment system if one gets a completion of an account every quarter and against this one has a right of objection for only a few months.)

- b) It is clear that one uses only stronger dimensioned systems with both systems types for all data that are to be newly encrypted or to be signed, and that new

keys must be exchanged additionally, etc. If one trusts in the fact that the soon breakable digital signature system is not yet broken, one can use it for the exchange of public keys of stronger asymmetric cryptographic systems. Otherwise a new original-key-exchange outside of the computer network, that is to be protected, is necessary - as well as with usage of symmetric cryptographic systems for key exchange. If one wants to regulate something legally obliging with the digital signature system, the public test key should be newly registered. In every case it is beneficial to let at least key registers confirm the exchange of the *public* keys. To avoid that key registers can explain arbitrarily keys for invalid, they should possess a digital signature (if necessary in the old cryptographic system) of the owner for a relevant message (key invalid from: date, time). The proceeding is different for messages that were encrypted / signed using the old, and too weak system in the future: *Secret data* must be encrypted with the stronger cryptographic system. This can happen according to the application, i.e., there is a safe environment for this new decryption with knowledge of keys needed for encryption of the present ciphertexts or not. It can be done by decrypting the present ciphertexts with the encryption system that is soon too weak and then encrypting the plaintext with the stronger dimensioned encryption system or by an additional encryption of the present ciphertexts. If asymmetric systems are considered, the latter might be the most frequent case. Independent from the use of symmetric or asymmetric encryption systems, proceeding after the latter case means that the implementation, of the encryption system which is soon too weak and the applied keys, must remain available. Avoiding the risk of changing the encryption (decrypting the weak, followed by encrypting with a strong system) (the plaintext is present for a short time) has its price. Unfortunately, one can never be sure whether all copies of the ciphertexts, as described, are encrypted stronger. Thus the outlined proceeding helps only against attackers that appear after the stronger encryption and also have, if necessary, difficulties in finding another copy of the only weakly encrypted ciphertext. Thus the outlined proceeding is no substitute for correct dimensioning starting from the beginning, but it is better, of course, than just waiting and wishing to keep the accruing insecurity of the too weakly dimensioned encryption system a secret.

*Signed data* must be provided with an additional signature in the stronger dimensioned signature system, best from the original signer. Theoretically one could oblige anybody in a transition period: everybody publishes stronger dimensioned test keys (signed for at least occasional authentication in each case with the not yet broken signature system). Then everybody is obliged to resign a message, that was signed by him with the old system, but now using the stronger system. This transition period must end of course, before the breaking of the old signature system is cheap enough. If the original Signer is not able to or not willing to do an additional signing, a time stamp service can be enabled: everyone who has a message signed by another one and wants to prevent the decay of this evidence, submits the message together with the signature (in the old signature system)

## B Answers

to a time stamp service. This signs in the new, stronger dimensioned signature system “message, signature in the old signature system, date, time”. If this is submitted later, this time-stamped signature is as authentic as it was at “date, time” in the old signature system, as the time stamp service is protected against corruption and as cryptographically secure as the new signature is to the time of submission. The protection of the time stamp service against corruption can be increased by the usual technology: instead of a signature in the new signature system a signature (independent from the others) is appropriated by independently implemented and pursued computers in each case in the new signature system. A message is considered only as correctly time-stamped if enough (independent) signatures are submitted in the new signature system. Because everyone can protect autonomously his own interests in case of resigning, the proceeding outlined here is much more satisfying. This applies particularly if the end of the cryptographic security of digital signatures (and therewith also the end of every transition period) can be determined in certain terms by the use of Fail stop signatures.

- c) I transmit the test key of an asymmetric signature system. Because then my partner can prove later with its help whether my public key of an asymmetric encryption system transmitted to him is authentic, by signing this key therefore by myself.
- d) Yes, because an “upgrade” of signature systems is securely realizable in an easier way than an upgrade of encryption systems. Furthermore the test key must be registered for legal liability.

[Exercise at page 423.](#)

### 3-16 crypto-policy by choice of a convenient length of key?

This proceeding is hopeless in principle:

Even if governments had clearly more money than big affiliated groups (what is far from clear), affiliated groups (as well as curious neighbors) can focus their cryptanalytic efforts very strongly on separate broadcasting stations and recipients. At the same time the fight against crime (and also espionage) requires of cryptanalysis a certain width, i.e., from a lot of broadcasting stations and recipients. Per key it means that compared to big affiliate groups, broken governments might have rather less than more computer power.

Beside this fundamental argument, from my point of view, there is naturally a row of other arguments, why such a “key length compromise” is unreasonable:

With the increase of available computer power in general, the key length would have to be adapted continually (cp. exercise 3-15a)), which is not possible to be managed meaningfully, however, in terms of secrecy for dated back messages (cp. exercise 3-15b)).

Criminals will not only trust in such weak encryption systems, but encode if necessary with stronger ones and additionally use steganographic systems if needed, cp. §4 and §9.

[Exercise at page 423.](#)

### 3-17 Authentication and encryption – in which order?

Let  $x$  be the message:

If asymmetric authentication is desire, so a digital signature, one should sign first and then encrypt, so that the recipient has a signed plaintext message to store after decrypting. The signature must be co-encrypted, because it contains information about the message in general. The sender creates  $k(x, s(x))$  or  $c(x, s(x))$ , depending on a symmetric or an asymmetric encryption system.

If symmetric authentication suffices, the order according to usefulness does not matter: one can do it as above, thus creating  $k(x, MAC)$  (In this case one will not have a asymmetric encryption system). Or one can authenticate the encrypted message by creating a MAC for  $k(x)$ . The latter has 3 advantages according to effort and power:

- It “uses”, through application of one-time pad for secrecy, less keys as well as less arithmetic expenditure, because secrecy is not supposed to increase the length of the message, while authentication does - and the expenditure of the second operation increases with the length of its input, i.e., the output of the first operation.
- If one checks authentication first and this test fails, one can save decrypting (and therewith expenditure) for him.
- Proving authentication and decrypting can take place parallel, so that the answer time can be shortened.

See p. #473

See p. #477

[Exercise at page 424.](#)

### 3-18 $s^2 \text{ mod } n$ – generator

- a) For  $n = 7 * 11 = 77, s = 2$  the following sequences result:

sequence  $s_i$ : 4 16 25 9 4 ...

sequence  $b_i$ : 0 0 1 1 0 ...

thus for the plaintext 0101 in case of addition modulo 2 the ciphertext 0110 results. The decrypter creates the sequence  $b_i$  in the same way and calculates using addition modulo 2 from the ciphertext 0110 the plaintext 0101.

And because it was so much fun, the second example:

For  $n = 7 * 11 = 77, s = 13$  the following sequences results:

sequence  $s_i$ : 15 71 36 64 15 ...

sequence  $b_i$ : 1 1 0 0 1 ...

thus for the plaintext 0101 in case of addition modulo 2 results in the ciphertext 1001.

- b) First we have to calculate back the sequence of the other  $s_i$  using  $s_4 = 25$ . We need the CRA, so we calculate the base vectors first:

Euclidean algorithm:

$$11 = 3 * 3 + 2; 3 = 1 * 2 + 1;$$

## B Answers

Backwards:

$$1 = 3 - 1 * 2 = 3 - 1 * (11 - 3 * 3) = -11 + 4 * 3$$

Thus  $up = 12, vq = -11$  and we have the formula  $s = 12 * s_q - 11 * s_p = 11(s_q - s_p) + s_q$ .

Furthermore we can pre-calculate  $(p+1)/4 = 1$  and  $(q+1)/4 = 3$ .

$s_3$ : (Remark: Here we can not just take 5 (the root that is known from calculating with integers), because we need the root of the 4 roots from  $25 \bmod 33$  that itself is a quadratic residue.)

Modulo 3:  $s_4 \equiv 1 \Rightarrow s_3 \equiv 1^1 \equiv 1$

Modulo 11:  $s_4 \equiv 3 \Rightarrow s_3 \equiv 3^3 \equiv 5$

$$\Rightarrow s_3 = 11(5 - 1) + 5 = 49 \equiv 16 \bmod 33.$$

$s_2$ :

Modulo 3:  $s_3 \equiv 1 \Rightarrow s_2 \equiv 1^1 \equiv 1$  (one can see that it will stay the same)

Modulo 11:  $s_3 \equiv 5 \Rightarrow s_2 \equiv 5^3 \equiv 4$

$$\Rightarrow s_2 = 11(4 - 1) + 4 \equiv 4 \bmod 33.$$

$s_1$ :

Modulo 11:  $s_2 \equiv 4 \Rightarrow s_1 \equiv 4^3 \equiv 16 * 4 \equiv 9$

$$\Rightarrow s_1 = 11(9 - 1) + 9 \equiv 31 \bmod 33.$$

$s_0$ :

Modulo 11:  $s_1 \equiv 9 \equiv -2 \Rightarrow s_0 \equiv (-2)^3 \equiv -8 \equiv 3$

$$\Rightarrow s_0 = 11(3 - 1) + 3 \equiv 3 \bmod 33.$$

The last bits of those are  $b_0 = 1, b_1 = 1, b_2 = 0, b_3 = 0$ .

Thus the plaintext is  $1001 \oplus 1100 = 0101$ .

- c) What happens here does not depend just on the  $s^2$ -mod- $n$ -Generator but happens always when one adds bitwise a one-time pad (cp exercise 3-10d)) or a more or less secure pseudo-one-time pad (Stream-cipher in deeper meaning, cp. §3.8.1):

If a bit of the ciphertext flips over then just this bit of the plaintext is false after decryption using addition modulo 2. But if a bit gets lost then the whole plaintext from this position is wrong :

Let the  $i$ -th bit of the plaintext be  $x_i$ . The  $i$ -th ciphertext bit is then  $S_i := x_i \oplus b_i$ . If  $S_i^* = S_i \oplus 1$  instead of  $S_i$  arrives at the recipient he will decrypt it as  $S_i^* \oplus b_i = x_i \oplus 1$ .

If  $S_i$  falls out, however, he will decrypt each bit from the i-th position with the wrong  $b_i$ : He reutes  $S_{i+1}$  for  $S_i$  and decrypts it with  $b_i$  and so on.

- d) The  $s^2$  - mod -  $n$ -Generator guarantees in its normal use as symmetric (and more than ever as asymmetric) encryption system as little authentication as Vernam cipher. (cp. part c)).

Nevertheless the answer for the question is: Yes. One may take an information-theoretical secure authentication code and use the sequence generated by the  $s^2$  -

$mod - n$ -Generator as key of the authentication system (the key that must be exchanged actually is only the seed for the  $s^2 - mod - n - generator$ ).

The resulting authentication system could be broken if the attacker is able to break the authentication code or the  $s^2 - mod - n$ -Generator. But the authentication code is information-theoretical secure, thus the  $s^2 - mod - n$ -Generator remains.

For specialists: One ought to prove that the attacker, in order to break the system, has to break the QRA (quadratic residuosity assumption) actually, i.e., that the resulting system is cryptographically secure. Sketch: If one can break the authentication code, if the keys of the  $s^2 - mod - n$ -Generator can be chosen, while it is unbreakable with random keys, this is a measurable difference between pseudorandom sequences and real random sequences. But it is proven that in the case of the  $s^2 - mod - n$ -Generator the sequences can not be differed by any statistical test from real random sequences below the QRA.

- e) The key exchange remains the same. Though both have to memorize not the seed  $s$  anymore but the current  $s_i$  after the first message. This requires that the recipient decrypts the ciphertexts just once and in the same order as they were encrypted. Otherwise the recipient would have to memorize more. Compare §3.8.2.
- f) Following §3.4.1.7 and §3.4.1.8 as well as exercise 3-13k) each square has exactly 4 roots mod  $n$ . (with  $n = p*q$ ,  $q \equiv 3 \pmod{4}$ ,  $p \not\equiv q$ ), from which exactly one is a square again. The root of 1 which itself is a root again is 1. Thus it is valid: if  $s_0 \neq 1$  all following inner states are  $\neq 1$ , since 1 results only from the square 1. The probability of reaching the inner state 1 after  $i$  steps is as high as the probability of choosing randomly one of the four roots of 1 as seed  $s$ , thus  $4/|\mathbb{Z}_n^*|$ .

[Exercise at page 424.](#)

### 3-19 GMR

- a)  $R = 28$  is not within the domain of the permutation pair, because  $\gcd(R, n) = \gcd(28, 77) = 7$ . Thus one can not sign anything using this. Therefore the example for  $R = 17$  follows. (There is  $17 \equiv 28 \pmod{11}$ , so somebody who calculates with 28 can find his calculations modulo 11 again.) Since  $\gcd(17, 77) = 1$  and  $(\frac{17}{77}) = (\frac{17}{11}) * (\frac{17}{7}) = (\frac{6}{11}) * (\frac{6}{3}) = (6^{(11-1)/2} \pmod{11}) * (3^{7-1}/2 \pmod{7}) = (-1) * (-1) = 1$   $R = 17$  lies in the domain of the permutation pair.

Because the messages are always 2 bit long, one does not need a prefix-free transformation.  $S := f_m^{-1}(R)$  can be created directly. This is (cp. §3.5.3):

$$S = f_{01}^{-1}(17) = f_1^{-1}(f_0^{-1}(17)).$$

Step 1: First  $x := f_0^{-1}(17)$  is needed. Following §3.5.2 one has to test at the beginning whether 17 or -17 is a square. (cp. §3.4.1.7)

Modulo  $p=11$ :  $17 \equiv 6$ ; Euler criteria:  $6^{(11-1)/2} = 6^5$ ;  
 $6^2 \equiv 3$ ;  $6^4 \equiv 3^2 \equiv 9$ ;  $6^5 \equiv 9 * 6 \equiv -1 \Rightarrow 17 \notin QR_p \Rightarrow 17 \notin QR_n$ .

## B Answers

The root  $w$  of  $-17$  is to be extracted. Modulo  $p=11$ :  $60 \equiv 5; 5^{(11+1)/4} = 5^3 \equiv 4$ ;  
 Modulo  $q=7$ :  $60 \equiv 4; 4^{(7+1)/4} = 4^2 \equiv 2$ ;

(One can shorten the following, but it is mentioned for completeness: Applying QRA and the “base vectors” from exercise 3-13g) results in  $w = -21 * 4 + 22 * 2 = -40 \equiv 37$ . This  $w$  is the root from which to square again.  $x$  is supposed to be the one with Jacobi symbol  $+1$  (thus  $\pm w$ ), which is  $< 77/2$ . This applies for  $x = 37$ .)

Step 2: Now  $S = f_1^{-1}(x) = f_1^{-1}(37)$  is calculated, thus  $S \in D_n$  with  $(2S)^2 \equiv \pm 37$ .

(First we would have to test whether  $x$  or  $-x$  is a square. But we know this from the first step: We calculated the root  $w$  which was a square in any case. One can proceed immediately with  $w$ . Furthermore we have to now calculate again with mod  $p$  and  $q$  solely for extracting a root of  $w$ . We did not need to compound  $w$  using QRA, i.e., we could leave out all parenthesized parts.)

Thus: Root  $w^*$  from the result of step 1:

Modulo 11:  $4^{(11+1)/4} = 4^3 \equiv 9$ ;

Modulo 7:  $2^{(7+1)/4} = 2^2 \equiv$ ;

Next one has to divide by 2. It can happen solely with  $p$  and  $q$  as well: (Extended Euclidean Algorithm for determination of the multiplicative inverse of 2 and subsequent multiplication with it, or halving in  $\mathbb{Z}$  of  $w^*$ ,  $w^* + p$  or  $w^* + q$ .)

Modulo 11:  $w^*/2 \equiv 10$ ;

Modulo 7:  $w^*/2 \equiv 2$ ;

The demanded signature  $S$  is one of the 4 numbers  $CRA(\pm 10, \pm 2)$ . Before QRA we have to ensure Jacobi symbol  $+1$ . Instead of the Euler criteria we can use this one (§3.5.1)

$$\left(\frac{2}{p}\right) = -1 \text{ and } \left(\frac{2}{q}\right) \equiv +1.$$

According to that it follows, because the Jacobi Symbols of  $w^* \bmod p$  and  $q$  were 1 each, that

$$\left(\frac{w^*/2}{p}\right) = -1 \text{ and } \left(\frac{w^*/2}{q}\right) \equiv +1.$$

We create  $CRA(-10, 2)$  (One could also take  $CRA(10, -2)$ ).

$$CRA(-10, 2) = CRA(1, 2) = -21 * 1 + 22 * 2 = 23.$$

And again we have just caught that of the numbers  $\pm S$  that is  $< 77/2 \Rightarrow S = 23$ .

Annotations:

1. It does not matter, whether one divides by 2 first and then applies CRA, or the other way round.

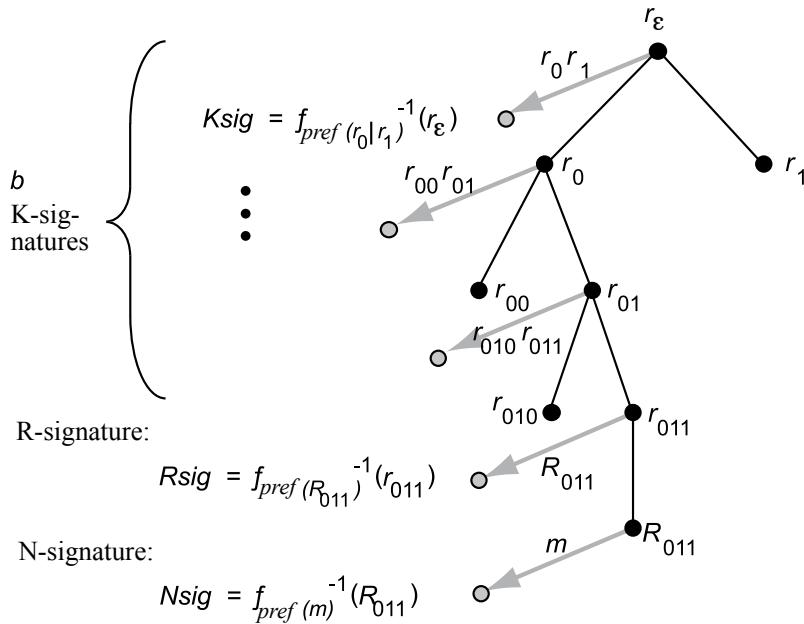
2. If you receive 12 as signature, then you have forgotten to pay attention to the Jacobi symbols. 12 has the Jacobi symbol

$$\begin{aligned} \left(\frac{12}{77}\right) &= \left(\frac{12}{11}\right) * \left(\frac{12}{7}\right) \\ &= 1^5 \bmod 11 * 12^3 \bmod 7 = 1^5 \bmod 11 * (-2)^3 \bmod 7 = -1. \end{aligned}$$

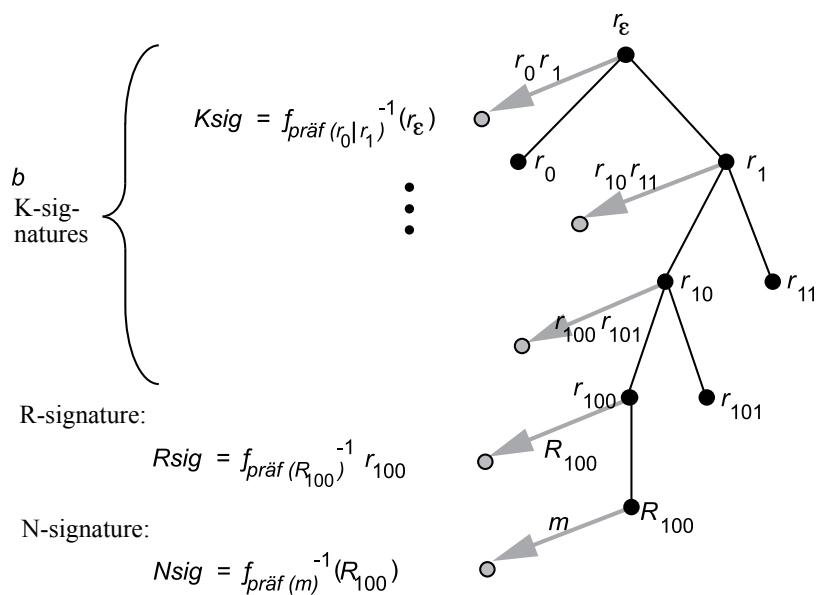
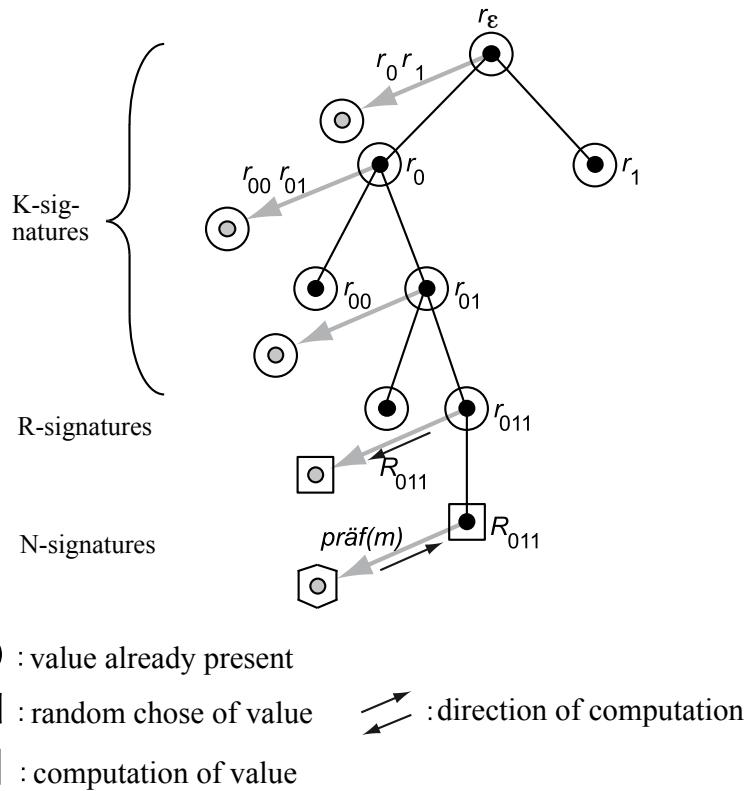
It means that 12 is not in the domain of the permutations. The tricky thing is that the rest of the signature appears to work though:  $f_0(f_1(12)) = 17$ , because  $f_1(12) = 4 * 12^2 \bmod 77 = 4 * 144 \bmod 77 = 4 * -10 \bmod 77 = 37$ .  $f_0(37) = -(37^2) \bmod 77 = 17$ . The test for whether the signature lies in the domain must be a part of the signature test.

- b) Because 8 messages are to be signed the reference tree has a depth about 3, as in the figures in §3.5.4.

Analogous to Figure 3.26, the fourth figure, i.e., the one attached to  $R_{011}$ , consists of the following parts: (“parts” are those parts here: The black references and the gray signatures.)

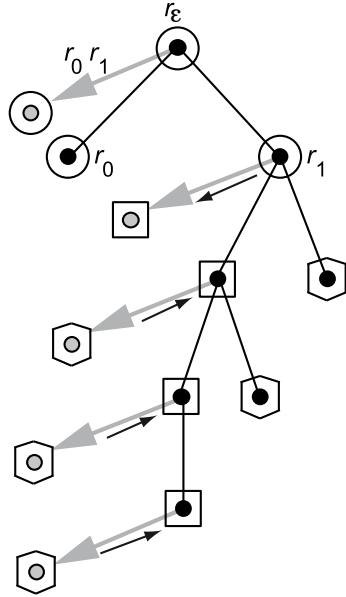


Through a comparison with Figure 3.26, one can see what must be newly chosen or calculated: the only new reference is  $R_{011}$ . New signatures are those of  $R_{011}$  in terms of  $r_{011}$  and (of course) of the message  $m$  in terms of  $R_{011}$ . If one considers, as in Figure 3.27, in which order one calculates at best, it results:



By comparing it with the fourth signature above one can see that only the toggling three references  $(r_\epsilon, r_0, r_1)$  are already there as well as the toggling K-signature.

(This is just the change from the left to the right half of the tree, after the first signature it is the most time-consuming operation.) For the order for calculation it results:



Of course one can send everything that was signed, but this is only necessary for parallel testing. If testing is proceeded sequentially one does not need to send the inner nodes of the signatures: The recipient of the signature has to recalculate all grey arrows in the opposite direction and only compare whether the root of the tree  $r_\epsilon$  that is known to him results.

- c) It is to show: if there would be an algorithm that finds collisions of functions  $f_{pref(m)}$  efficiently, then there is an algorithm that finds collisions for pairs  $(f_0, f_1)$ . Therefore it is shown how to directly calculate from each collision  $f_{pref(m)}$  a collision of  $(f_0, f_1)$ . The following collision is given  $m, m^*, S, S^*$  with  $m \neq m^*$  and  $f_{pref(m)}(S) = f_{pref(m^*)}(S^*)$ . Set  $v := pref(m)$  and  $w := pref(m^*)$ . Let us call the bits of  $v = v_0v_1\dots v_k$  or  $w = w_0w_1\dots w_{k'}$ .  $v_i$  and  $w_i$  for the first bits from the left where  $v$  and  $w$  differ. Since  $v$  and  $w$  are prefix free codings of different messages  $m, m^*$  applies  $i \leq \min(k, k')$  and of course  $0 \leq i$ . Then:

$$\begin{aligned} f_v(S) &= f_{v_0} \cdot f_{v_1} \cdots \cdots f_{v_i} \cdots \cdots f_{v_k}(S) = \\ &f_{w_0} \cdot f_{w_1} \cdots \cdots f_{w_i} \cdots \cdots f_{w_{k'}}(S^*) = f_w(S^*) \end{aligned}$$

Since in both lines the same permutations are shown (for all  $j$  with  $0 \leq j \leq i$  it applies  $V_j = w_j$ ), from the equality of the figures follows the equality of the original figures. Thus it is

$$f_{v_i} \cdots \cdots f_{v_k}(S) = f_{w_i} \cdots \cdots f_{w_{k'}}(S^*).$$

## B Answers

Consequently

$$f_{v_i}(f_{v_{i+1}} \cdots f_{v_k}(S)) = f_{w_i}(f_{w_{i+1}} \cdots f_{w_{k'}}(S^*))$$

is a collision of  $f_0$  and  $f_1$ .

Prefix free coding is needed to ensure that  $v$  and  $w$  differ on each bit position, which *exists* for both (otherwise there would be either no  $v_i$  or no  $w_i$ .)

[Exercise at page 425.](#)

### 3-20 RSA

- a) Key generation:

Let  $p = 3, q = 11$  therefore  $n = 33$

Therefore:

$$\Phi(n) = (p - 1) \bullet (q - 1)$$

Let  $c = 3$ . Test if  $\gcd(3, 20) = 1$ . (Euclidean Algorithm)  $\Rightarrow$  Yes!

In order to find the exponent for decoding, one can either use the conventional algorithm or calculate it for  $\text{mod } p$  and  $q$  separately:

**Conventional Algorithm:** We are looking for

$$d = c^{-1} \text{ mod } \Phi(n)$$

here

$$d = 3^{-1} \text{ mod } 20$$

The extended Euclidean Algorithm gives us  $d = 7$ .

**More efficient Algorithm:**

$$\begin{aligned} d_p &:= c^{-1} \text{ mod } (p - 1) \quad \text{in our case } d_p := 3^{-1} \equiv 1 \text{ mod } 2 \\ d_q &:= c^{-1} \text{ mod } (q - 1) \quad — \quad d_q := 3^{-1} \text{ mod } 10 \end{aligned}$$

From the extended Euclidean Algorithm we get  $d_q = 7$ .

**Encryption:**

Take  $m = 31$  for the plaintext. The corresponding ciphertext  $S$  is  
 $S = 31^3 \equiv (-2)^3 \equiv -8 \equiv 25 \text{ mod } 33$

**Decryption:**

Conventional algorithm:

$$S^d = 25^7 \equiv (-8)^7 \equiv 64^3 \bullet (-8) \equiv (-2)^3 \bullet (-8) \equiv 64 \equiv 31 \text{ mod } 33$$

33 is indeed the plaintext.

More efficient algorithm:

$$\text{Mod 3: } S^{d_p} = 1^1 = 1$$

$$\text{Mod 11: } S^{d_q} \equiv 3^7 = 9^3 \bullet 3 \equiv (-2)^3 \bullet 3 \equiv 3 \bullet 3 = 9$$

Using the CRA (as usual) we get  $m = 31$ .

- b) To make things simple we can just modify the example used in a): Let key generation be identical (only c is now called the test key and d the signing key).

If the plaintext just so happens to be  $m = 25$ , then the signature will be the  $3^{rd}$  root of m which gives us  $Sig = 31$  (see above) and testing will result in  $Sig^3 = 31^3 \equiv 25$ .

Note:

1. If you want to use encryption for some messages and authentication for others, you should choose different key pairs. Otherwise, in order to get the plaintext for an encrypted RSA block, one could just get the owner of the key to sign that block. This risk can be circumvented, though, using a one-way hash function before signing a message or by designing the cryptographic protocol in a way that prevents signing of arbitrary messages. But why take this risk in the first place just to save that little bit of effort in key generation and storage in RSA?

2. Neither redundancy nor hash functions were used in our example; this system is still weak against Davida attacks and should not be used unmodified.

- c) To 0 and 1 respectively – independently of the chosen key pairs. We see that these plaintexts should be avoided by using an appropriate encoding as input to RSA.

If blanks (the characters are by far most likely to result in long sequences) are not encoded as 0...0, this will be the case automatically with reasonable probability. You can also take care of this by means of redundancy and hash functions which should be used anyway.

- d) One approach would be to insert 100 random bits at fixed positions in each RSA block.

We need enough random bits so it becomes impossible for an attacker who sees the ciphertext S to just guess a likely plaintext  $m$ , and do a complete search to find out if S contains m. In order to do this, he would just encrypt m using the public key c and iterating through all combinations of random bits. With just 8 random bits this would only take 256 iterations.

Timestamps, even if encoded in more than 8 bits, are not random enough. An attacker might just try out all probable timestamp values.

- e) If the randomly chosen bits are part of the output of the encryption system, the proposed in-deterministic flavor of RSA is not safe against active attacks: an attacker could just use the same approach as in deterministic RSA.

The answer becomes more complex if the random bits are not part of the output. But, as was pointed out in §5.4.6.8 and more elaborately in [PfPf\_90], it will still be “no” — at least if they are the first or the last 100 bits.

Therefore, in a concrete implementation of RSA, for example, 100 random bits should be chosen in combination with a redundancy rating. The message format could then be  $m^* = (m, z, h(m, z))$  (where ordering is irrelevant), m being the

## B Answers

message itself,  $z$  being the random bits and  $h$  being a one-way hash function. The encrypted message will be  $m^*c$ .

If the range of  $h$  is 160 bits (see §3.8.3), we need to subtract 260 bits off the overall RSA block size in order to get the net RSA block size. 740 bits remain if the module is 1000 bits long which is 74 %.

- f) Random bits can either be inserted into the to-be-signed message itself or into the block that is exponentiated for RSA (or both).

If they are inserted into the to-be-signed message, they will be hashed with it. It is crucial for the security of the signature that the hash function is collision resistant for every choice of random bits. This should be the case for a good hash function — but why take the additional risk and the extra effort of hashing more bits?

If random bits are inserted into the block signed by RSA, the risk of multiplicative attacks is increased. Why take that risk?

All in all, I would always sign deterministically using RSA.

- g) According to §3.4.1.1 the complexity of signing an RSA block of length  $2l$  with an exponent of length  $|c|$  is proportional to  $(2l)^2 \bullet |c|$ . The expense per bit when signing extremely short messages does not differ, because only one block has to be encrypted.

When encrypting very long messages it is noted that the number of bits that can be put into a single RSA block is increased with  $l$ . Neglecting the random number and the hash value that take up space also, we can encrypt  $2l$  bits in each RSA block. Therefore the complexity for very long messages is proportional to  $\frac{(2l)^2 \bullet |c|}{2l} = 2l \bullet |c|$ . Under the best of circumstances the complexity only grows proportionally to  $l$ . (In reality it is even less, at least for very large values of  $l$ , since the fast algorithms mentioned in §3.4.1.1 really pay off. On the other hand, for extremely long messages most of the time hybrid encryption (§3.1.4) is used instead of plain RSA.)

- h) Let  $l$  be the security parameter, i. e., the length of  $p, q$ .

**Mod n:** What we need is  $x^d \bmod n$  ( $d = c^{-1} \bmod \Phi(n)$  has already been calculated while generating the key). The length of  $x$  and  $d$  each are about  $2l$ .

Exponentiation using “square-and-multiply” will therefore be broken down to  $\frac{3}{2} \bullet 2l = 3l$  modular multiplications of numbers of length  $2l$  (counting squaring as a multiplication).

**Mod p,q:** We need  $x^{d_p} \bmod p$ ,  $x^{d_q} \bmod q$  and the CRA for those values ( $d_p$ ,  $d_q$ , and the “base vectors” for the CRA have already been calculated during key generation). So we are left with two more exponentiations, the arguments of which only have a length of  $l$  ( $x$ , because it can be reduced  $\bmod p$  and  $q$  resp., and  $d_p$  and  $d_q$  because they were calculated  $\bmod(p-1)$  and  $\bmod(q-1)$ , respectively) and the

CRA. The latter operation is made up of only two multiplications and is therefore negligible compared to the exponentiations.

That makes about  $2 \bullet \frac{3}{2} \bullet l = 3l$  modular multiplications of numbers of length  $l$ .

So the ratio between the overall complexity of the approaches is just the same as between modular multiplications of numbers of size  $l$  and  $2l$  respectively. (Standard) modular multiplication is made up of multiplication and division. These operations both have a complexity of  $O(l^2)$ . Therefore, the expense quadruples when doubling  $l$ .

- i) A random exponent will have roughly size  $2l - 1$ . Instead of  $2l - 2$  squaring operations and  $\frac{2l-2}{2}$  multiplications when using a random exponent, only 16 squaring operations and one multiplication will have to be performed. Therefore we will only have to do 17 operations instead of  $3l - 3$ . With this, the public operation will be sped up by  $\frac{3l-3}{17}$ . If  $l = 512$ , which is more realistic, the factor will be  $\frac{3 \bullet 512 - 3}{17} \sim 90$ .
- j) Great idea: since  $scd(p-1, q-1) < \Phi(n) = (p-1)(q-1)$ , because  $p-1$  and  $q-1$  are at least both divisible by 2, we can substitute  $\Phi(n)$  by  $scd((p-1), (q-1))$  and get:

$c$  will be chosen from a slightly smaller interval in step 3 this way, although the same numbers in that interval can be chosen. In step 4 the multiplicative inverse to  $c, d$ , will be smaller on average. We now get

$$c \bullet d \equiv 1 \pmod{scd(p-1, q-1)}.$$

This yields

$$c \bullet d \equiv 1 \pmod{p-1}$$

as well as

$$c \bullet d \equiv 1 \pmod{q-1}.$$

The validation of encryption and decryption in §3.6.5.1 still holds.

Does this modification interfere with the security of RSA? I believe that is very, very improbable, because only the choice of  $c$ , which has to be published anyway, changes slightly — and in practice  $c$  is very often chosen to be small anyway, to make encryption less costly (see §3.6.1 and also part i) of these exercises).

Those who wish to be absolutely puristic about it may want to use this hybrid approach:

Step 3 is done as in §3.6.1 using  $\Phi(n)$ , and in step 4 first  $\Phi(n)$  is used and then  $gcd(p-1, q-1)$  where the smaller result will be selected.

This way nothing changes as seen from the outside — the security of RSA will stay completely unchanged. We only save a bit of effort while encrypting since  $d$  is a little smaller. Not much though, because  $p-1$  and  $q-1$  will not have too many common prime factors if  $p$  and  $q$  are chosen randomly and stochastically independently.

## B Answers

k) We recycle the example from a), so  $n_1 = 33$

We choose

$$\begin{aligned} n_2 &= 5 \bullet 17 = 85 \\ n_3 &= 23 \bullet 29 = 667 \end{aligned}$$

We get

$$\begin{aligned} \Phi(n_2) &= 4 \bullet 16 = 64 \\ \Phi(n_3) &= 22 \bullet 28 = 616 \\ \gcd(3, 64) &= 1 \\ \gcd(3, 616) &= 1 \end{aligned}$$

(In case you chose 7 or 13 as prime factors of  $n$ , you probably found  $\Phi(n)$  to be divisible by 3)

For the plaintext  $m = 31$  we get

$$\begin{aligned} S_1 &= 31^3 \equiv (-2)^3 \equiv -8 \equiv 25 \pmod{33} \\ S_2 &= 31^3 \equiv 29791 \equiv 41 \pmod{85} \\ S_3 &= 31^3 \equiv 29791 \equiv 443 \pmod{667} \end{aligned}$$

The extended Euclidean Algorithm gives us

$$-18 \bullet 33 + 7 \bullet 85 = 1$$

From the CRA we get

$$-18 \bullet 33 \bullet 41 + 7 \bullet 85 \bullet 25 = -9479 \pmod{33 \bullet 85} = 1741$$

(Check:  $1741 \pmod{33} = 25, 1741 \pmod{85} = 41$ )

Once more, this time with  $33 \bullet 85 = 2805$  and  $667$ :

The extended Euclidean Algorithm<sup>4</sup> yields:

$$778 \bullet 667 - 185 \bullet 2805 = 1$$

The CRA gives us:

$$778 \bullet 667 \bullet 1741 - 185 \bullet 2805 \bullet 443 = -673566391 \pmod{(667 \bullet 2805)} = 29791$$

(Check:  $29791 \pmod{667} = 443, 29791 \pmod{2805} = 1748$ )

The third root of  $29791$  is  $31$  — which is exactly the plaintext  $m$ .

<sup>4</sup>Since computation gets difficult with these numbers, I include it here.

*Euclidean Algorithm:*

$$\begin{aligned} 2805 &= 4 \bullet 667 + 137 \\ 667 &= 4 \bullet 137 + 119 \\ 137 &= 1 \bullet 119 + 18 \\ 119 &= 6 \bullet 18 + 11 \\ 18 &= 1 \bullet 11 + 7 \\ 11 &= 1 \bullet 7 + 4 \\ 7 &= 1 \bullet 4 + 3 \\ 4 &= 1 \bullet 3 + 1 \end{aligned}$$

- l) Breaking RSA completely means being able to calculate d for any given n and c. The proof mentioned in the exercise shows that knowing n, c, and d you can factor n — breaking RSA completely therefore is just as hard as factoring. Unfortunately this proof does not even prove the impossibility of breaking RSA universally without knowledge of n — i. e., to find a procedure that is equivalent to the key without being able to factorize n.

[Exercise at page 425.](#)

### 3-21 DES

- a) **Encryption:**

$$\begin{aligned}
 K_1 &= 011, & K_2 &= 010 \\
 L_0 &= 000, & R_0 &= 101 \\
 L_1 &= R_0 = 101, & R_1 &= L_0 \oplus f(R_0, K_1) = 000 \oplus (R_0 \bullet K_1 \bmod 8) \\
 &&&= 101 \bullet 011 \bmod 8 = 5 \bullet 3 \bmod 8 = 111 \\
 L_2 &= R_1 = 111, & R_2 &= L_1 \oplus f(R_1, K_2) = 101 \oplus (111 \bullet 010 \bmod 8) \\
 &&&= 101 \oplus (7 \bullet 2 \bmod 8) = 101 \oplus 110 = 011
 \end{aligned}$$

Exchanging both halves yields the ciphertext 011111.

#### Decryption

We are using the variable names from the pictures in §3.7.3. Go there for explanation.

$$\begin{aligned}
 R_2 &= 011, & L_2 &= 111 \text{ are given} \\
 R_1 &= L_2 = 111, & L_1 &= R_2 \oplus f(R_1, K_2) = 011 \oplus (111 \bullet 010 \bmod 8) \\
 &&&= 011 \oplus (7 \bullet 2 \bmod 8) = 011 \oplus 110 = 101 \\
 R_0 &= L_1 = 101, & L_0 &= R_1 \oplus f(R_0, K_1) = 111 \oplus (R_0 \bullet K_1 \bmod 8) \\
 &&&= 111 \oplus (101 \bullet 011 \bmod 8) = 111 \oplus (5 \bullet 3 \bmod 8) \\
 &&&= 111 \oplus 111 = 000
 \end{aligned}$$

Since our choice of variable names includes an implicit exchange of values from left to right, we do not need to exchange both halves explicitly. The plaintext therefore reads 000101, which is exactly the one that was encrypted.

---

*Going back:*

$$\begin{aligned}
 1 &= 4 - 3 \\
 &= 4 - (7 - 4) \\
 &= -7 + 2(11 - 7) \\
 &= 2 \bullet 11 - 3(18 - 11) \\
 &= -3 \bullet 18 + 5(119 - 6 \bullet 18) \\
 &= 5 \bullet 119 - 33(137 - 119) \\
 &= -33 \bullet 137 + 38(667 - 4 \bullet 137) \\
 &= 38 \bullet 667 - 185(2805 - 4 \bullet 667) \\
 &= 778 \bullet 667 - 185 \bullet 2805
 \end{aligned}$$

## B Answers

- b) Tables are created explicitly so every result can be looked up immediately. Where possible, multiple s-boxes are integrated into one (in DES at least two: array of  $2^{12}$  entries).
- c) The trick in fast software implementations of these systems is to store and look up permutations that would take up many bit operations otherwise in tables. A table with  $2^{32}$  entries would be too large though. As an example we will demonstrate, how to get along with only 4 tables of  $2^8$  entries each:

Entries will be 32 bits long and the  $i^{th}$  table tells us where to permute the  $i^{th}$  set of 8 bits.

Example:  $2^{nd}$  table – e. g., bits 9-16. Let  $W(9), \dots, W(16) = 28, 13, 1, 5, 30, 14, 2, 19..$ . In every entry bit no. 9 will be at position 28, bit no. ten will be at position 13, ..., bit no. 16 will be at position 19 and all other bits will be 0.

The final result of the permutation will be achieved by indexing the  $i^{th}$  table with the  $i^{th}$  set of 8 bits from the input and get the binary OR or XOR of all 4 results.

- d) For one pair of plaintext blocks and corresponding ciphertext blocks we try out different keys until we find one that matches. This will, on average, take  $2^{55}$  attempts. After that, we check whether the key we found also is correct for other pairs of plaintext and ciphertext blocks, which is very likely. Thus, the average expense will be  $2^{55}$  DES-encryptions.
- e) As was laid out in §3.7.6 we choose the complement of one of the given plaintext block / ciphertext block – pairs, i. e., we take the compliment of the plaintext block and ask for the corresponding ciphertext. With this approach it takes  $2^{54}$  attempts on average to obtain the key, which is half as many.

More formally:

The plaintext block/ciphertext block – pair  $(x, \text{DES}(k,x))$  is given.

We get  $\bar{x}$  and ask for  $\text{DES}(k,\bar{x})$ .

Because of the complementarity property we have  $\text{DES}(k,\bar{x}) = \overline{\text{DES}(\bar{k},x)}$ . We calculate  $\text{DES}(\bar{k},x)$  and have the ciphertexts for  $x$  encrypted with  $k$  as well as with  $\bar{k}$  at our hands for the attack.

In order to find the key  $k$  we try out different keys until encryption yields either  $\text{DES}(k,\bar{x})$  or  $\text{DES}(k, x)$ .  $k$  will be  $k'$  or  $\bar{k}'$  respectively.

If we make sure that each of the possible  $2^{56}$  values of  $k'$  and  $\bar{k}'$  is tried out at most once, the search terminates after at most  $2^{55}$ , on average after  $2^{54}$  steps, where every step is made up of one DES-encryption and two comparison operations between the plaintext and the ciphertexts. Since the comparison operations are much less costly than the encryption, the expense for this chosen plaintext/ciphertext attack is only half as big as for the plaintext/ciphertext attack described earlier.

Same way as above we have to test whether  $k$  is the right key for other plaintexts blocks as well.

- f) There is a linear equation for every one of the 64 ciphertext bits that is applied to plaintext bits according to the permutations and to key bits according to permutations and partial key generation — with known coefficients because the permutations and the partial key generation are public and fixed. Due to the fact that plaintext and ciphertext are known, we get 64 linear equations with 56 coefficients. Every software package for equation solving can figure those out.

It obviously does not matter if the block length is 64 or 128 bits or whether the key length is 56 or 256 bits. The algorithm evidently breaks every Feistel cipher with known, fixed permutations. It probably can be modified to work on arbitrary linear block ciphers.

[Exercise at page 426.](#)

### 3-22 Modes

- a) There is no redundancy. More precisely: A given ciphertext can be decrypted into any arbitrary plaintext. When re-encrypted, the resulting ciphertext will of course be identical to the input — in particular its last block. If the messages do not contain any internal redundancy, e.g., in files that only contain numbers, the attacker might just send arbitrary ciphertexts.
- b) (This scheme is a bit better: In general the last plaintext block to some random ciphertext will not be all zeros, so the attack will be spotted. But: )

If the ciphertext is at least three blocks long, the attacker can modify all but the last two. Since the mode CBC synchronizes every two blocks, the last block will still be decrypted correctly to a block of zeros and the attack will not be detected automatically.

- c) Now, if some ciphertext block is altered, encryption will result in something completely different. Thus there will be a wrong block in the plaintext block buffer when encrypting the next block. In general, the resulting error will not be eliminated by synchronization because a function of the last block will be added to the next block after decryption, so decryption will fail again and result in a wrong plaintext buffer all over again.

{Arguing PCBC that is a synchronous cipher which would render the attack ineffective, would be wrong though. For this the definition of “synchronous” in §3.8.1 is too weak, because even dependency on the bit position in the message satisfies it. Regard the following example for clarification: OFB is a synchronous stream cipher, but the ciphertext can be modified at will, as long as no ciphertext bits of the authenticator are changed and no ciphertext is omitted.}

- d) **ECB:** The attacker can not achieve much, because the block cipher is assumed to be secure and it is used for nothing more than a block cipher. So there are only the disadvantages left that were mentioned along the definition of block ciphers

## B Answers

and their respective attacks: Deterministic block ciphers can be attacked actively by asking the victim for the encryption of likely plaintexts. The resulting ciphertexts can be compared then to intercepted ciphertexts. With a bit of luck some of the intercepted ciphertexts can be “decrypted” and even without luck certain plaintexts can be ruled out.

**CBC for encryption:** The attacker can attack the system actively by inserting the values in the buffers. If these values are simply recycled, i. e., unless every message begins with a value that is chosen by the sender, after an active attack on the first block the same attack as on ECB can be used with the same probability of success. If the values in the buffers are reset after the active attacks, the attacker can still scan the ciphertext for blocks he knows (i. e., from an earlier attack on ECB or CBC). For these blocks he can then proceed to calculate the respective plaintext in CBC: he just needs to subtract the last key. But this kind of search is harder, because the attacker can not predict the encrypted values as well (and have them encrypted during his attack). His chances to succeed are slimmer than in ECB if the buffers are reset appropriately.

**CBC for authentication:** If single blocks are authenticated, everything that was said for encryption holds. If many blocks are authenticated at a time even if the buffers are not reset he has to guess their precise values, which is unfeasible if reasonable block lengths are chosen. This is said assuming that the attacker can not just “ask for” the authenticated values already during his attack, because no cryptographic system is secure against that. It is the embedding protocol’s responsibility to take care of this. The simplest approach is to have the partner create part of the message that is to be authenticated.

**CFB for encryption:** The attacker succeeds for the next output block precisely if he knows the encryption of the value in the shift register. If the shift register is not reset for every message, he can in an active attack ask for the encryption for a chosen plaintext (works for ECB, CBC and CFB), leave that plaintext in the shift register, have a cup of coffee, and wait for the next output unit to subtract the known value and celebrate the successful attack...

**CFB for authentication:** ditto.

**OFB:** The attacker either needs to predict the value of the shift register or insert it himself. The latter can not be done by manipulating the plaintext stream, because it is not fed back. Success of an active attack depends solely on the initialization of the shift register.

**PCBC:** For the advantages mentioned for CBC to be achieved, i. e., same probability of success as for ECB, the attacker can try to set the values in the two buffers actively to insert a value of  $h$  that suits his purposes. Apart from initializations he faces more obstacles than with CBC, which is illustrated by the following attempt to set the buffer values in PCBC: Send plaintext  $P_1$ . Now we know the contents of the plaintext block buffer ( $P_1$ ) and we can find out the contents of the ciphertext block buffer by examining the output ( $C_1$ ). Thus  $h(P_1, C_1)$  can be calculated.

Still it is hard to set both values. A plaintext block  $P_2$  could be chosen to set  $P_2 + h(P_1, C_1)$  to some arbitrary value and this way produce the ciphertext  $C_2$ . But now  $P_2$  has to be modified and the plaintext block buffer changes. The same is true for choosing the plaintext block  $P_2$ , because adding  $h(P_1, C_1)$  produces unwanted results in  $C_2$ .

For a known function  $h$  it proves difficult to produce a certain output value from a small set of input values. Thus even a known and irregular distribution of plaintexts probably can not be used in favor of the attacker, provided initialization of the buffers is done in a wise manner. See “CBC for authentication”.

**OCFB:** As opposed to PCBC, in OCFB the shift register value can be set by the attacker even without initialization as long as:

- $h$  is not a one-way function
- he has the chance to see the result  $Res$  of the encryption before the corresponding unit in the plaintext stream has to be handed over.

With  $Res$  the attacker has the value of the first input parameter of  $h$  and, unless  $h$  is a one-way function, is able to calculate an appropriate second value for the second input parameter for many output values of  $h$ . From this value he subtracts the desired plaintext and receives the plaintext unit that he has to insert to get his choice of result of  $h$ . If this scheme is repeated  $\frac{b}{r}$  times, there will be a chosen value in the shift register. Now what was said in “CFB for encryption” applies.

- e) Coding will be done like this: append a bit of given value (say 1) to the plaintext. If the text length is not a multiple of the block length, pad with bits of another value (say 0).
1. is satisfied, because we pad up to the next multiple of the block length.
  2. is satisfied, because all bits from the last 1 to the end of the text can be cut off to decode the text.
  3. needs a definition for “minimal”. Sensible definitions might be:
    - Expansion over all plaintexts is minimal.
    - Expansion over all plaintexts weighted their respective probability is minimal. This will be the same as above in case of standard distribution.
    - Worst case expansion (expansion of the worst-case plaintext) is minimal.

The given encoding satisfies the last definition, because every encoding has to add at least one bit to at least one text (in order to distinguish between “padded” and “not padded”, and because this encoding adds exactly one bit if possible. The encoding probably also satisfies the first definition (why not try to prove that?). You can find a distribution of plaintexts for every given encoding that makes length expansion not minimal according to the second definition.

- f) Shift registers and feedback are substitute by a counter that starts at some initialization value and is increased for every output unit. In order to access the  $i^{th}$  unit,

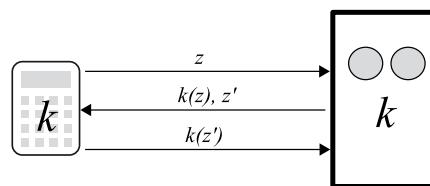
## B Answers

$i - 1$  is added to the initialization value — off we go. This mode is called **Counter Mode** for obvious reasons. [Schn\_96, page 205].

See p. #473  
 Exercise at page 426.

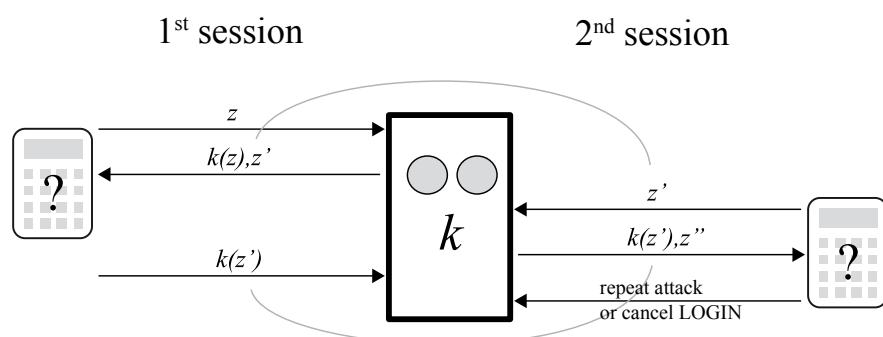
### 3-23 Mirror attack

Let the main frame implement the same secure block cipher (amongst other helpful features). Also assume that independently chosen keys have been exchanged between the main frame and each user (more accurately: between the main frame and the “calculator”). Login will now be done according to the following protocol: send random number, expect encryption, decrypt, compare. This will be done in both directions, i. e., client authenticates to main frame and main frame authenticates to client (for simplicity the user id, which has to be submitted, too, so the main frame knows which key to use, is not included in the following picture).



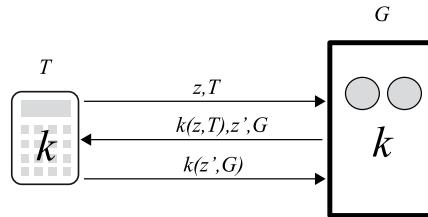
If main frame and “calculator” behave completely symmetrically (as suggested by the picture and by our discussion so far) the protocol is insecure regardless of the strength of the block cipher. If the main frame supports more than one session at a time, the second random number it generated in the first session may be used as the first random number in the second session. If the main frame does not detect this (for example by enforcing one login per user at any one time) and responds correctly in session two, his correct response may be fed back to it in session one. The main frame does not authenticate the “calculator”, but its own “reflection”. We are out of luck!

Minor mistake — major security hole (same goes for the “calculator”, if it accepts the role of the second partner during authorization, so the roles are symmetrical, too. This could be the case, if clients can log into many main frames at a time).

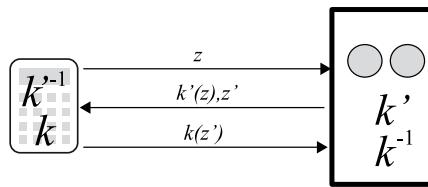


Correct processes can be designed by following one out of two schemes: either instances have some memory and accept only random numbers that satisfy certain conditions (which is kind of contrary to the idea of a random number), or they behave asymmetrically. This could be done by either

- attaching their identity (e.g., ‘M’ for ‘main frame’ and ‘C’ for ‘calculator’) to the random number.



- having only one of the parties decrypt and one encrypt.
- having two keys instead of one for one pair of main frame and client; one for the main frame to encrypt and the client to decrypt and vice versa.



It would also be secure – though not very elegant – to only react to a random number after having received replies to all random numbers that were sent, as in 3-6 part e. This way the main frame could not handle logins in parallel.

With precise enough clocks in place in both main frame and “calculator”, copying from the terminal to the calculator can be omitted once by using time instead of the random number (note the remark in 2-3 to make weakest possible assumptions on random number generation).

This will still not put up with attackers hijacking sessions after login. To account for this scenario identity must be authenticated continuously.

(It is noted that we need a block cipher with strong security properties in this exercise, because it is used more in the way of an authentication system than a encryption system and encryption systems do not generally make good authentication systems. See exercise 3-10 part d. If time stamps are used, for example, an attacker should not be able to guess from an encrypted message that he has seen the encryption of similar, forged messages.)

[Exercise at page 427.](#)

### 3-24 End-to-end encryption

No matter if symmetric or asymmetric cryptography is used, every user needs at least one secret key. Thus, wherever a computer is used by multiple users, the main problem is where to keep secret keys out of reach of other legitimate users of the system.

If the system is trusted enough, so that we can safely assume that it is able to protect one user's data from other users and if its hardware security shields off physical attacks, then the problem is solved inherently.

If the system is completely untrusted, then not only storing the key, but also encryption, decryption, testing signatures and most of all signing needs to take place on a different, personal computer. It should have at least some kind of a small key pad and display to communicate with its owner directly. Incarnations of this concept include so-called 'super smart cards' 'smart floppy disks' (computers resembling floppy disks that connect to others via the floppy disk drive, which do not need an extra reader and provide more space than a wallet-sized chip card with it is standardized magnetic-strip-dimensions) up to notebooks.

As a compromise between effort and security every user can enter a password of length (actually entropy) great enough to create a key for a symmetrical encryption system. Using this key the actual keys for end-to-end encryption are decrypted at the beginning of the session and encrypted at its end. This procedure may of course be broken either if the passwords do not contain enough entropy after all or the symmetric encryption system can be broken, or if other users can get their hands on the password by simulating a password prompt, so the user thinks he is communicating with the operating system itself. The latter could be prevented by always checking the computer for hardware manipulations and booting the authentic operating system before logging in. The two things that are needed for this approach are intrusion detecting casings, see §2.1, and authentic booting see [[Cles\\_88](#), [ClPf\\_91](#), [Groß\\_91](#)].

[Exercise at page 427.](#)

### 3-25 End-to-end encryption without exchanging keys before

- a) To send out a message to others, a set of one-time pads is generated. Those will be sent out separately on separate routes (e.g., via a friend and, where possible, from separate places like home, office, public phones, cellular phones) and media (e.g., phone, fax, e-mail) and possibly at different times (provided the service supports this, i.e., it does not have real-time requirements). The sum of all those pads is used to en-/decrypt and/or authenticate the message. An attacker would have to be at many different places at many different times in order to eavesdrop on all the pads and read or manipulate the encrypted/authenticated message. This method trades off security in availability for security in confidentiality and integrity: only one pad (or, of course, the message itself) needs to be intercepted in order to prevent the message from being read.

The one-time pad and the information-theoretically secure authentication codes may serve as cryptographic systems, the former being as simple as can be and the latter certainly feasible with reasonable effort without a computer.

- b) Once they get back their crypto devices they can use asymmetric crypto systems. For this the crypto devices that have deleted all their secret keys need to create new key pairs. In addition to a) Bob tells Alice his public key (or vice versa), for example via telephone — Alice’s crypto device of course has also deleted Bob’s public keys because it could not tell whether only its secret keys were going to be extracted, or whether the public keys were going to be manipulated, too. (These days telephone conversations are authenticated by the speaker’s voice. As soon as speech synthesizers work sufficiently well, that will be history. We should not rely on this authentication alone.). After that the public keys are used to exchange one or more pads or authenticate the message directly.

If Bob and Alice took precautions in case they lost their keys by agreeing on a common secret they memorized (and thus always carry around with them) they can, of course, use that as a starting value for symmetric key generation. This applies most to b). Using a mnemonic pass phrase<sup>5</sup> both will be able to remember and a cryptographic hash function<sup>6</sup> a shorter value can be generated, that contains more entropy per character. This value can now be used as a starting value for key generation or even directly as a symmetric key.

[Exercise at page 428.](#)

### 3-26 Identification and authentication by an asymmetric encryption system

- a)  $U$  creates a random number and encrypts it, so only  $T$  can read it:  $c_T(z)$ .  $U$  sends  $c_T(z)$  to the instance that is supposed to be  $T$  and expects  $T$ , depending on the protocol, to either send back  $z$ , or to use  $z$  in some other cryptographic operation and send back the result. If that happens, that instance is in fact  $T$  (with a certain probability that is determined by the length of  $z$  and given that  $c_T$  is authentic, that the asymmetric encryption system is secure and  $z$  really was random). Now  $U$  has authenticated  $T$ . But beware: this naïve approach is prone to the changing attack laid out in 1. If some attacker  $A$  wants to be authenticated as some user  $T$  he just sends on  $c_T(z)$  to  $T$  and relays the reply to  $U$ . We see that this protocol does not authenticate  $T$  as a direct communication partner, but only makes sure  $T$  is involved in communication somewhere and, more specifically,  $T$  receives  $z$ . So, if  $U$  inserts its message to  $T$  into  $z - z = c_T(\text{message})$ , this protocol nevertheless does its job. When receiving  $z$ ,  $U$  knows that the message has reached  $T$ .

(Warning: If you came up with an essentially different solution, check whether you made  $T$  use his private key  $d_T$  on something that had not been encrypted using  $c_T$ . This kind of operation — using  $d_T$  first, for signing, then  $c_T$ , for testing the signature — is only possible in certain encryption systems, like RSA, but not in every encryption system. Where this is possible, the encryption system can be used as an authentication system as well, and the solution of the exercise is trivial.)

---

<sup>5</sup>a fairly long string with little entropy per character

<sup>6</sup>that may be publicly known

## B Answers

- b) The asymmetric encryption system needs to be secure against adaptive chosen-cipher-text attacks. Otherwise an entity's responses during authorization may be used to break their encryption system.
- c) Construction of a symmetric authentication protocol from an asymmetric concealment protocol:
  - 1)  $T$  sends message  $M_i$  to  $U$ .
  - 2)  $U$  generates a random number  $z$ , encrypts  $N_i$  and  $z$ , and sends  $c_T(N_i, z)$  to  $T$ .
  - 3)  $T$  decrypts using  $d_T$  and checks whether  $M_i$  from step 1 is identical to the one from step 2. If so,  $T$  sends  $z$  to  $U$  - otherwise  $T$  never returns  $z$ .
  - 4) If  $U$  receives the same value for  $z$  that he created,  $M_i$  is an authentic message from  $T$ .

This authentication system is only symmetric, because  $z$  is useless for everyone else and therefore does not prove anything, since it was not created by  $T$ , but by  $U$ . Therefore  $U$  could send  $z$  to himself arbitrarily.

The disadvantage of this protocol compared to other authentication protocols is that it needs three messages instead of one. It therefore consumes more time and network resources.

Tabular representation of a symmetric authentication system on top of an asymmetric encryption system.

Given:  $U$  knows  $c_t$ ,  $T$ 's public key.

$$\begin{array}{ccc} T & \xrightarrow{M_i} & U \\ T & \xleftarrow{c_T(M_i, z)} & U \\ T & \xrightarrow{z} & U \end{array}$$

Result: If — in this asymmetric authentication system — every effectively possible modification of  $c_T(M_i, z)$  changes  $z$ , which in general should be the case in a block cipher, then  $U$  knows after the third message that  $M_i$  was really sent by  $T$ .

Correspondingly, the tabular representation of a symmetric authentication system through a symmetric authentication system could look like this:

Given:  $U$  and  $T$  both know  $k$ ,  $k$  being a key for a symmetric authentication system.

$$T \xrightarrow{M_i, k(N_i)} U$$

In the first approach  $T$  produces  $N_i$  and  $U$  only supplies  $z$ . Also,  $T$  does not react to  $U$ 's reply to his message if  $N_i$  is not part of it. Thus, the asymmetric encryption system does not have to withstand an adaptive active attack in its nastiest form.

[Exercise at page 428.](#)

### 3-27 Secrecy by symmetric authentication systems

Instead of every single bit of the message, only an appropriate MAC is sent. The recipient then figures out which bit corresponds to the MAC.

Sender	recipient
<p>Knows <math>k_{AB}</math></p> <p>Let the message M be <math>b_1, \dots, b_n, b_i \in \{0, 1\}</math></p> <p>Calculates  <math>MAC_1 := code(k_{AB}, b_1),</math>  <math>\dots,</math>  <math>MAC_n := code(k_{AB}, b_n)</math></p> <p>Sends  <math>MAC_1, \dots, MAC_n</math></p>	<p>Knows <math>k_{AB}</math></p> <p>Checks whether  <math>MAC_1 = code(k_{AB}, 1)</math> or  <math>MAC_1 = code(k_{AB}, 0)</math>  Thus receives the matching value for <math>b_1</math>  <math>\dots</math>  Checks whether  <math>MAC_n = code(k_{AB}, 1)</math> or  <math>MAC_n = code(k_{AB}, 0)</math>  Thus receives the matching value for <math>b_n</math></p>

The  $MAC$ s need to be long enough, so the recipient can check which bit matches the  $MAC$ . The small authentication code in Figure 3.15, for example, is insufficient: with a key value of 00,  $H$  and  $T$  are authenticated with 0. Therefore the recipient is unable to decide whether  $H$  or  $T$  are supposed to be transmitted. The same holds for a key value of 11 and a  $MAC$  of 1. Designing the  $MAC$  long enough makes the probability for this to be insignificant.

The message is concealed perfectly using the above protocol, since no one but the sender and recipient can test which bits match the  $MAC$ s. If someone could guess those bits with a probability greater than 0.5, this would be an approach for breaking the authentication system.

With long enough  $MAC$ s the message is actually authenticated: after changing a  $MAC$  it will, with overwhelming probability, either not match 0 or 1, or it will match the same value as before. Otherwise the authentication system could be broken with a probability too great to neglect: the attacker could change the  $MAC$  and flip the corresponding message bit that is sent in the regular authentication system.

Instead of authenticating a single bit it can be more efficient to authenticate more than one bit using a single  $MAC$ . This saves bandwidth (more bits per  $MAC$ ), and also makes it necessary for the recipient with a given number  $l$  of authenticated bits per

## B Answers

*MAC* to do a maximum of  $2^l$  and an average of  $2^{l-1}$  steps to decrypt. If  $l$  bits are sent separately, then  $2 \times l$  steps are necessary in a complete search, and only  $l$  steps if 0 is assumed whenever the test fails for 1. Comparing the difference in expense between  $l$  bits being sent together and separately we find that for  $l = 2$  sending  $l$  bits together has only advantages, because  $2^l = l \times 2$ , but the expense for greater  $l$  grows very fast.  $l = 8$  might be a sensible compromise for many applications.

See p. #369

[Exercise at page 428.](#)

### 3-28 Diffie–Hellman key exchange

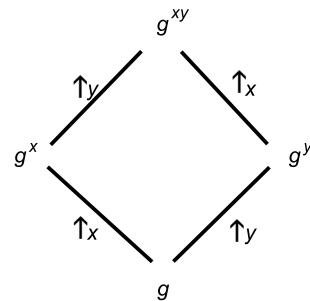
a) *Basic idea:*

We are using the fact that exponentiation is commutative, i. e., it does not matter in what order exponentiation is done.

This, put into a formula:

$$(g^x)^y = g^{xy} = g^{yx} = (g^y)^x$$

And, graphically:



b) *Anonymity:*

Solution on the organizational level: there is no reason why public keys should have to be linked to real identities — they could belong to pseudonyms just as well.

Solution on the protocol level: instead of using his commonly known public key the sender chooses a new key pair for every encrypted message that is sent out. He uses the new private key to calculate a common secret key for communication with the recipient and encrypts accordingly. After that, he sends out the message along with the new public key<sup>7</sup> to his partner. At reception the public key can be used together with the recipient's private key to get the common secret key and decrypt the message. (If you have ever heard of asymmetric crypto using the

---

<sup>7</sup>that is why this modification of Diffie–Hellman key exchange can not be used for stenography with public keys, see §4.1.2

ElGamal-style encryption system: that is a special case of what we just discussed. Elegiacal uses modular multiplication as the symmetric encryption routine in the hybrid encryption system laid out here. [ElGa\_85][Schn\_96, p 478]):

Now, what if the sender wants an anonymous reply? Simple: the common secret key can just be re-used.

So asymmetric encryption systems and Diffie-Hellmann key exchange are on par with respect to anonymity, too.

c) *Individual choice of parameters*

Every participant chooses their own  $p$  and  $g$  to publish as part of their public key.

Every one can exchange secret keys with everybody else by using their  $p$  and  $g$  and then proceed as was laid out in §3.9.1. After that the new, pair-specific public keys can be told to the other participants.

Because of that last step, this modification of the Diffie-Hellman protocol is not adequate for steganography with public keys, see §4.1.2. This modification is used in PGP 5 and over, unless “Faster Key Generation” is selected. This is de facto the protocol variant from exercise part b) except that every instance also chooses and publishes their own  $p$  and  $g$ .

Even though this way public keys are chosen on a per participant pair basis, it is important to make sure public keys are authentic. This can be ensured by using certificates (see §3.1.1.2) and testing of certificates by the recipient. Wherever the sender can create signatures which the recipient can validate, it is best to have the sender sign his pair-specific public keys himself. (see also §9.2).

d) *DSA (DSS) as a basis*

On this basis Diffie-Hellman key exchange works flawlessly, because all parameters provide sufficient security. Depending on whether  $p$  and  $g$  are not the same for participants, things will have to be done according to b). DSA (DSS) can — contrary to its original purpose — be used as a basis for encryption and — if  $p$  and  $g$  are the same for all participants — even for steganography with public keys §4.1.2.

It is much harder to argue whether Diffie-Hellman key exchange when carried out this way is as secure as the method in §3.9.1 and b), because there is a subtle difference in distribution of  $g$  and a great difference in the length of  $x$ . To my knowledge though, this procedure can be regarded as secure.

[Exercise at page 429.](#)

### 3-29 Blind Signatures with RSA

Let 2 be the text (plus the result from the collision-resistant hash function, see §3.6.4.2) that is to be signed blindly, and 17 the masking factor. Thus we get for the masked text

$$2 \times 17^t \bmod 33 = 2 \times 17^3 = 2 \times 29 \bmod 33 = 58 \bmod 33 = 25$$

## B Answers

(When making up such an example, of course, you go the other way around: 25 is given for the to-be-signed message from exercise 3-20b). Therefore we are looking for two numbers, whose product (mod 33) is 25. After choosing 2 and 29 the third root of 29 (mod 33) has to be calculated. We are using the inverse exponent:

$$\begin{aligned} 29^7 &\equiv 29^2 \times 29^2 \times 29^2 \times 29 \quad \text{mod } 33 \\ &\equiv (-4)^2 \times (-4)^2 \times (-4)^2 \times (-4) \quad \text{mod } 33 \\ &\equiv 16 \times 16 \times 16 \times (-4) \quad \text{mod } 33 \\ &\equiv 256 \times (-64) \quad \text{mod } 33 \\ &\equiv 25 \times 2 \quad \text{mod } 33 \\ &\equiv 17 \quad \text{mod } 33 \end{aligned}$$

From exercise 3-20b) we now use the masked text: 25, 31

(since  $31^3 \equiv (-2)^2 \equiv (-8) \equiv 25 \text{ mod } 33$ )

The recipient now divides the second component, the blind signature, by the masking factor 17.

Euclidean Algorithm: calculate  $17^{-1} \text{ mod } 33$ :

$$\begin{aligned} 33 &= 1 \times 17 + 16 \\ 17 &= 1 \times 16 + 1 \end{aligned}$$

And back again:

$$\begin{aligned} 1 &= 17 - 1 \times 16 \\ &= 17 - 1 \times (33 - 1 \times 17) \\ &= 2 \times 17 - 33 \end{aligned}$$

Therefore  $17^{-1} \text{ mod } 33 = 2$ .

So the signature is:

$$31 \times 17^{-1} \text{ mod } 33 = 31 \times 2 \text{ mod } 33 = 29$$

Double-checking our result for correctness — how the recipient of the blind signature can test whether the signer is trying to shortchange him:

$$\text{Signature}^t \equiv (-4)^3 \equiv -64 \equiv 2 \text{ mod } 33 = 2$$

With relief we note that  $\text{Signature}^t \text{ mod } 33 = 2 = \text{Text}$  and thus our calculation probably is correct, too.

[Exercise at page 430.](#)

### 3-30 Threshold scheme

The smallest adequate prime number  $p$  is 17, because the numbers between 13 and 17 are not prime.

The polynome  $q(x)$  is  $q(x) = 2x^2 + 10x + 13 \text{ mod } 17$ .

Substitution yields  $(i, q(i))$ :

$$\begin{aligned} q(1) &\equiv 2 + 10 + 13 \equiv 25 \text{ mod } 17 = 8 \\ q(2) &\equiv 8 + 20 + 13 \equiv 41 \text{ mod } 17 = 7 \\ q(3) &\equiv 18 + 30 + 13 \equiv 61 \text{ mod } 17 = 10 \\ q(4) &\equiv 32 + 40 + 13 \equiv 85 \text{ mod } 17 = 0 \\ q(5) &\equiv 50 + 50 + 13 \equiv 113 \text{ mod } 17 = 11 \end{aligned}$$

The formula from §3.9.6 gives us:

$$\begin{aligned}
 q(x) &= 8 \times \frac{x-3}{1-3} \times \frac{x-5}{1-5} + 10 \times \frac{x-1}{3-1} \times \frac{x-5}{3-5} + 11 \times \frac{x-1}{5-1} \times \frac{x-3}{5-3} \mod 17 \\
 &= \frac{8}{8} \times (x-3) \times (x-5) + \frac{10}{-4} \times (x-1) \times (x-5) + \frac{11}{8} \times (x-1) \times (x-3) \mod 17 \\
 &= (x-3) \times (x-5) + \frac{10}{-4} \times (x-1) \times (x-5) + \frac{11}{8} \times (x-1) \times (x-3) \mod 17 \\
 &= (x-3) \times (x-5) + 10 \times 4 \times (x-1) \times (x-5) + 11 \times 15 \times (x-1) \times (x-3) \mod 17 \\
 &= (x-3) \times (x-5) + 6 \times (x-1) \times (x-5) + 12 \times (x-1) \times (x-3) \mod 17 \\
 &= 2x^2 + 10x + 13
 \end{aligned}$$

$q(0)$  of course values to the constant term, which in this case is 13.

Alternatively it might be more convenient substituting 0 for  $x$  from the start instead of figuring out the polynome  $q(x)$ .

$$\begin{aligned}
 &8 \times \frac{-3}{1-3} \times \frac{-5}{1-5} + 10 \times \frac{-1}{3-1} \times \frac{-5}{3-5} + 11 \times \frac{-1}{5-1} \times \frac{-3}{5-3} \mod 17 = \\
 &\frac{120}{8} + \frac{50}{-4} + \frac{33}{8} \mod 17 = \\
 &15 + 50 \times 4 + 33 \times 15 \mod 17 = \\
 &15 + (-1) \times 4 + (-1) \times 15 \mod 17 = \\
 &13
 \end{aligned}$$

[Exercise at page 430.](#)

### 3-31 Provably secure usage of several encryption systems?

Having  $n$  encryption systems we divide up our plaintext into  $n$  pieces using an information theoretically secure threshold scheme in such a way that all  $n$  pieces are necessary to reconstruct the plaintext. We then encrypt every single piece using a different encryption system.

We get additional expense from the application of the threshold scheme as well as from having to submit all  $n$  pieces to the recipient (instead of just the result from the product cipher). The overall expense we get is about  $n$  times as big.

None of the two combinations beats the other with regard to security:

- It is true that it is impossible to prove anything in a product cipher about the effectiveness of the ciphers 2 to  $n$ . But since the attacker never gets to know the intermediate results, a product cipher generally is much more secure than the most secure cipher in the chain, and even more secure than the sum of all used encryption systems. What this says is that the effort necessary to break the product cipher is substantially higher than the sum of efforts needed to break each individual cipher.
- In order to be able to prove that this combination is as hard to break as the most securely used cipher (and even as the sum of all encryption systems), we need to make use of the strong assumptions that were given in the exercise. In general they should hold. But as in the discussion of the security of the product cipher we are now assuming things that are true ‘in general’. That is why, from my point of view, the product cipher out-performs this combination not only in terms of efficiency, but also in terms of security.

## B Answers

A very simple implementation of an information theoretically secure threshold scheme in which all  $n$  parts are needed in order to decrypt the secret would be applying the Vernam cipher  $n-1$  times: the  $n-1$  keys together with the ciphertext make up the  $n$  parts necessary.

See p. #464  
[Exercise at page 430.](#)

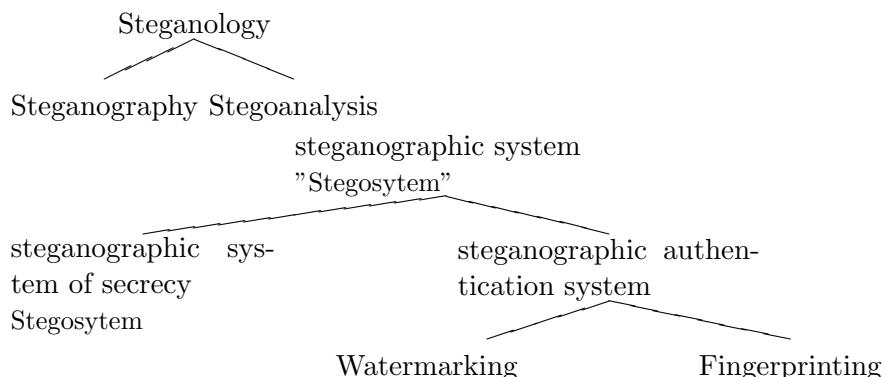
## B.4 Answers for “Basics of Steganography”

### 4-1 Terms

Steganology integrates steganography (the knowledge of the “good people”’s algorithms) and steganalysis (the knowledge of the attacker’s algorithms on a steganographic system). It is crucial to know about steganalysis to be able to estimate the security of steganographic systems.

Steganographic systems are either encryption systems (security aim: hiding confidential communication) or authentication systems (security aim: embedding protected author information). The next criterion for differentiation is whether only author information is embedded (watermarking) or also user related information (fingerprinting).

The advantage of introducing the term “stegosystem” as the root of the right concept tree is that now we have a short, popular term to our disposal. The disadvantage, however, lies in the loss of differentiation between secrecy and authentication. So I use this term for steganographic systems of secrecy only — when using it in a broader sense, I will always denote that with quotation marks. My concept tree looks like this.



[Exercise at page 430.](#)

### 4-2 Properties of in- and output of cryptographic and steganographic systems

- Below you will find figures copied from §3.1.1.1, which are precisely block diagrams for cryptographic systems.

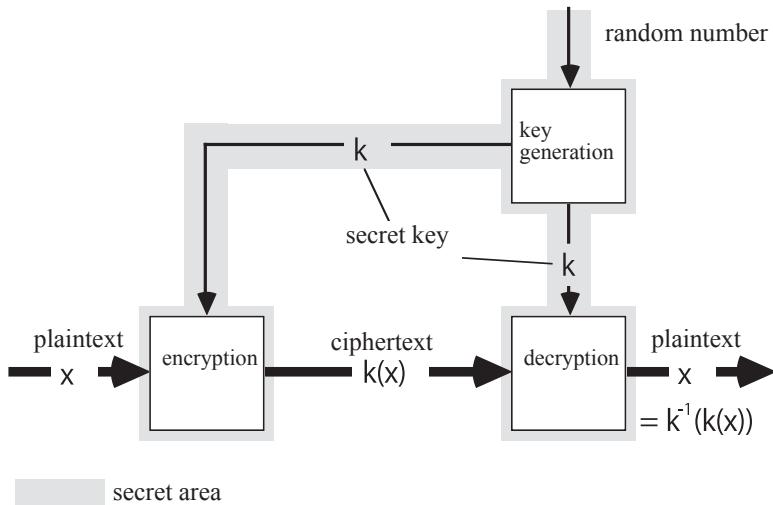


Figure 3.2: symmetric encryption system.

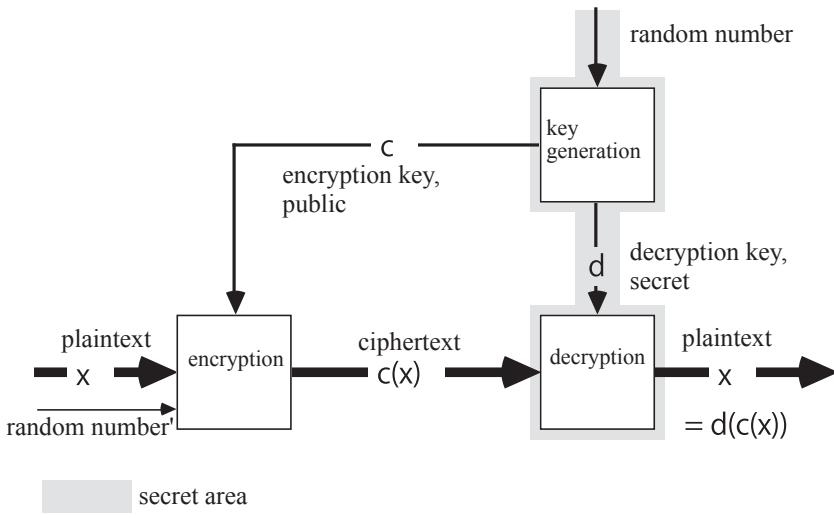


Figure 3.4: asymmetric encryption system.

When considering, for each diagram, all three algorithms as one system, the difference between in- and output symmetric and asymmetric encryption systems is only whether or not encryption needs an extra random number. Another difference lies in the requirements to the channel for key distribution. When looking at the interfaces of the three algorithms, the difference between symmetric and asymmetric encryption is whether the same key is used for de- and encryption (so it has to be kept secret) or keys for de- and encryption differ (so they are public).

A cryptographic system has (and is allowed) to assume that random numbers it gets as input are really random — where necessary random number generation should be part of the system that uses it. A good cryptographic system should

## B Answers

not make any assumptions about the plaintext — that is input from outside the system (and therefore a given). Therefore, when publishing the key, encryption has to be carried out indeterministically, so ‘simple’ plaintexts can not be guessed, encrypted, and compared to the ciphertext. On the other hand though, there are no restrictions to the resulting ciphertext — except that it perhaps should not be excessively long.

- b) Below you will find a figure that is copied from §4.1.1.1, which is precisely a block diagram for steganographic secrecy according to the embedding model.

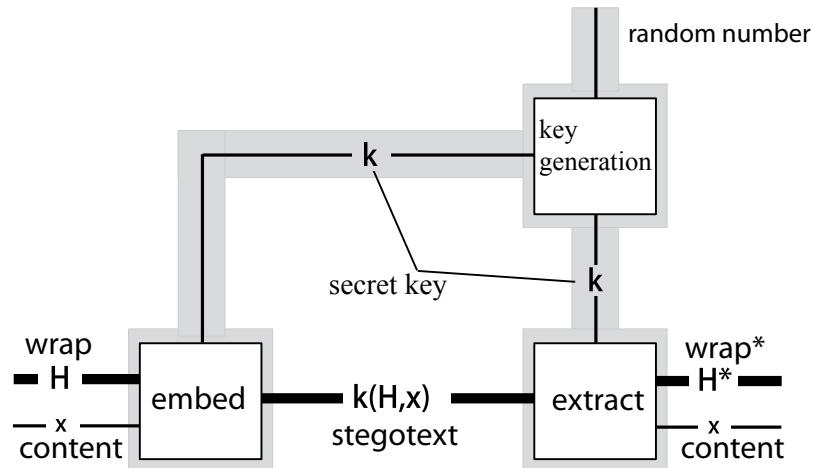


Figure 4.4: steganographic system of secrecy according to the embedding model.

There is only one block diagram, because there are no known asymmetric steganographic systems of secrecy.

Ideally a steganographic system of secrecy would not have to make any assumptions about the wrap, because in this model this is input (and therefore a given). (If we created a wrap in our steganographic system — synthetic steganographic secrecy — we would have a plausibility issue in our application context.) Unfortunately, no stegosystem will be able to work with arbitrary wraps, so certain assumptions have to be made and therefore every stegosystem has to rely on them to be adhered to. (The stegosystem should check adherence to those assumptions to a certain extent and refuse to embed the content into an inappropriate wrap.)

For all other pieces of input the same is true as for cryptography. Additionally, the output has to be exactly as plausible as the input wrap.

- c) In encryption systems there are no assumptions the implementation is unable assure. This is different in steganographic secrecy (according to the embedding model — not according to the synthetic model): Assumptions about the wrap can not be assured by the implementation.

[Exercise at page 431.](#)

### 4-3 Is encryption superfluous when good steganography is used?

If it is impossible for an attacker to tell whether a secret message is embedded, he can not possibly understand it.

I would still always use encryption in practice. The first reason is to make sure that at least the secret message stays confidential even if the steganographic system of secrecy is not as secure as was hoped for. The second — and at least equally important — reason though, is to make the content indistinguishable from white noise. This makes it possible for the steganographic system of secrecy — more accurately: to its embedding function — to make strong, yet realistic assumptions about the input content.

[Exercise at page 431.](#)

### 4-4 Examples for steganographic systems

1. From my collection of about 90 recent holiday pictures I choose the one that by coincidence contains the four bits I would like to send. Security would be excellent — provided there is such a picture, which is very likely.
2. I choose one picture and scan it over and over again until the result contained the four bits I want to send. Security would be excellent.
3. In order to send 40 bits I choose 10 pictures as I did in 1. (which will still work out with fair probability). Every once in a while I will have to send the same picture twice. This will only provide reasonable security if the sender is known to be a bit absent-minded at times. So this crypto system would be primarily for notoriously abstracted professors.
4. In order to send 40 bits I choose 10 of my favorite pictures and scan them as in 2. Security would be good, although the time that scanning takes could be a concern.

[Exercise at page 432.](#)

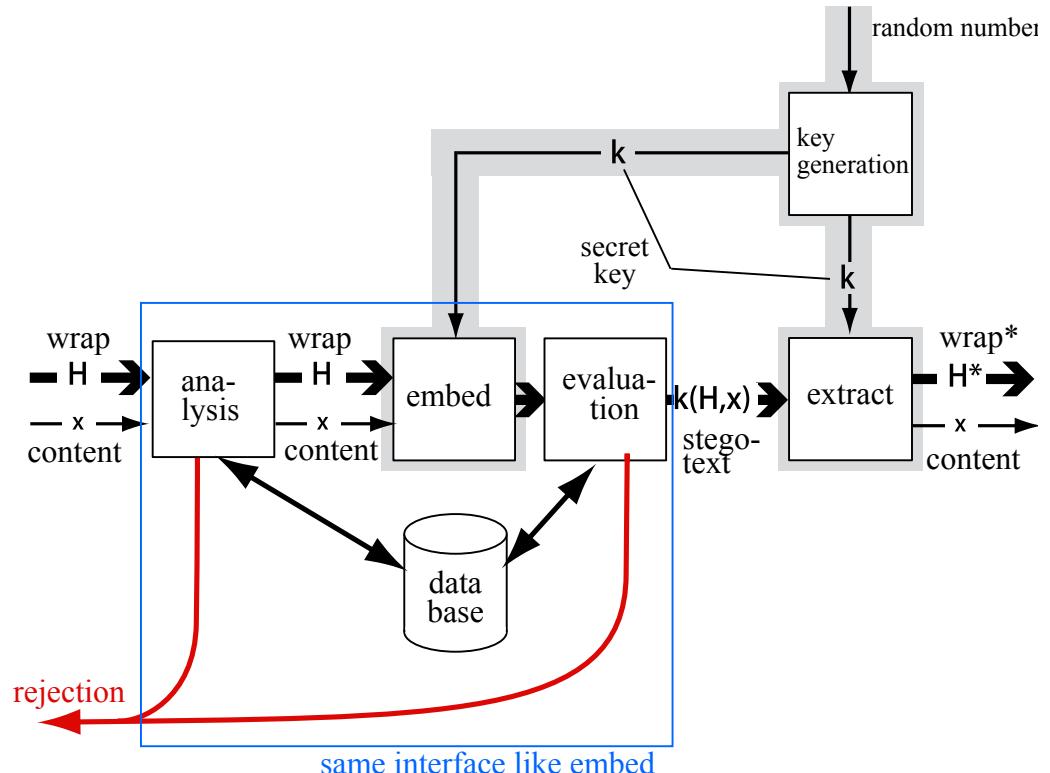
### 4-5 Modeling steganographic systems

Making up models is more an art than a deterministic process — there are no right (or wrong) models but models that are more (or less) appropriate for a given purpose than others. A smart person once said, you are not done constructing a machine whenever there is nothing you could add, but when there is nothing you could take out of it. This certainly is a good creed to work by in the art of model building.

This, applied to the context of steganographic systems, means that Anja, Berta, and Caecilie all do make good points about certain steganographic systems. I.e. if you, for example, can be sure that wrap material never repeats bit by bit, you do not need a data base etc. So, since analysis, data base, and rating are not part of every steganographic system, and also since they can be thought of as part of the implementation of the model given by Figure 4.4 (see the following figure), I tend to agree with Doerthe... to a certain extent. This is because Anja’s idea, that at least in some steganographic systems

## B Answers

there should be some error output that rejects inappropriate wrap material, is actually a new one. There could be discussion in a gentleman's round some time about whether this should be included in the model explicitly, or whether it is implied because every operation may fail and there needs to be error output that the embedding system has to react to. Anyway: Our ladies' round's decision is that models of steganographic systems should include error output.



[Exercise at page 432.](#)

### 4-6 Steganographic secrecy with public keys?

To start with, I would not call it steganographic secrecy with public keys, because embedding and extraction are always done using the same information. I would say this is bootstrapping a steganographic system with key using a keyless steganographic system. It can be useful for example where keyless steganography provides less net bandwidth, requires more computing power, or is less secure.

[Exercise at page 433.](#)

### 4-7 Enhancement of security by using several steganographic systems

There is not much sense in combining several steganographic systems of secrecy, because the existence of a confidential message in a chain of steganographic systems (see exercise 3-5) is hidden only by the outmost system. Also a great data overhead would be created.

## B.5 Answers for chapter “Security in communication networks”

When talking about steganographic authentication systems, the answer will be less simple: it might be useful to use a chain of just a few systems, since they would all have to be broken. On the other hand every application of a system will cause a certain — though very small — disturbance of the artwork. Hence not too many systems should be used.

[Exercise at page 433.](#)

### 4-8 Crypto- and steganographic secrecy combined?

Combining those two is a good idea, because that way the confidentiality of the message is at least as strong as the weaker one of the two systems (as long as stego and crypto keys are chosen independently of each other!). To make sure the existence of the message is hidden as well as when using steganography only, it needs to be encrypted first with a cryptographic system (or two, or more — see exercise 3-5) and then embedded steganographically. The recipient, of course, first has to extract and then to decrypt the message.

[Exercise at page 433.](#)

### 4-9 Crypto- and steganographic authentication in which order?

Steganographic authentication first (i.e., watermarking or fingerprinting) then digital signature of the steganographically modified artwork (otherwise steganographic authentication would invalidate the digital signature).

[Exercise at page 433.](#)

### 4-10 Stego-capacity of compressed and non-compressed signals

When using lossless compression all the entropy of the signal (i.e., its stego-capacity) is preserved. If there is any redundancy in the signal, it is just encoded in a shorter way. Hence per-bit stego-capacity is increased by the use of Gzip, Stuffit et al.

Lossy compression, however, reduces the overall entropy of the signal and therefore the overall stego-capacity. What this means for the per-bit stego-capacity depends on the ratio of reduction of redundancy to reduction of entropy.

[Exercise at page 433.](#)

## B.5 Answers for chapter “Security in communication networks”

### 5-1 Quantification of unobservability, anonymity, and unlinkability

[Exercise at page 434.](#)

### 5-2 End-to-End or link encryption

[Exercise at page 434.](#)

## B Answers

### 5-3 First encrypting and then encoding fault-tolerance or the other way round?

[Exercise at page 434.](#)

### 5-4 Treatment of address- and error-recognizing fields in case of encryption

[Exercise at page 434.](#)

### 5-5 Encryption in case of connection-oriented and connection-less communication services

[Exercise at page 435.](#)

### 5-6 Requesting and Overlaying

[Exercise at page 435.](#)

### 5-7 Comparison of “Distribution” and “Requesting and Overlaying”

[Exercise at page 436.](#)

### 5-8 Overlayed Sending: an example

[Exercise at page 436.](#)

### 5-9 Overlayed sending: Determining a fitting key combination for a alternative message combination

[Exercise at page 438.](#)

### 5-10 Several-access-methods for additive channels, e. g., overlayed sending

[Exercise at page 438.](#)

### 5-11 Key-, overlaying-, and transmission topology in the case of overlayed sending

[Exercise at page 439.](#)

### 5-12 Coordination of overlaying- and transmissionalphabets in case of overlayed sending

[Exercise at page 439.](#)

### 5-13 Comparison of “Unobservability of adjacent wires and stations as well as digital signal regeneration” and “overlayed sending”

[Exercise at page 439.](#)

**5-14 Comparison of “overlaid sending” and “MIXes that change the encoding”**

[Exercise at page 440.](#)

**5-15 Order of basic function of MIXes**

[Exercise at page 440.](#)

**5-16 Does the condition suffice that output messages of MIXes must be the same lengths?**

[Exercise at page 440.](#)

**5-17 No random numbers → MIXes are bridgeable**

[Exercise at page 440.](#)

**5-18 Individual choice of a symmetric encryption system at MIXes?**

[Exercise at page 440.](#)

**5-19 Less memory effort for the generator of untraceable return addresses**

[Exercise at page 440.](#)

**5-20 Why changing the encoding without changing the length of a message**

[Exercise at page 441.](#)

**5-21 Minimal expanding of length in a scheme for changing the encoding without changing length of a message for MIXes**

[Exercise at page 441.](#)

**5-22 Breaking the direct RSA-implementation of MIXes**

[Exercise at page 441.](#)

**5-23 Use of MIX-channels?**

[Exercise at page 441.](#)

**5-24 Free order of MIXes in case of fixed number of used MIXes?**

[Exercise at page 441.](#)

**5-25 How to ensure that MIXes receive messages from enough different senders?**

[Exercise at page 442.](#)

## B Answers

### 5-26 Coordination protocol for MIXes: Wherefore and where?

[Exercise at page 442.](#)

### 5-27 Limitation of responsibility areas – Functionality of network terminals

[Exercise at page 442.](#)

### 5-28 Multilaterally secure Web access

[Exercise at page 443.](#)

### 5-29 Firewalls

[Exercise at page 443.](#)

## B.6 Answers for “Value exchange and payment systems

### 6-1 Electronic Banking – comfort version

- a) On the one hand, this is ultra-comfortable for the customer only as long as his trust in the bank is ultimate — because in case of a lawsuit there are no documents that could serve as proof. On the other hand, the bank has to trust its customers — or trust that they will always win in court.
- b) Unfortunately not; in today's electronic banking systems the bank (its banking machine) gets nothing (no digital messages) from the customer it could not forge itself.
- c) This does not change a thing, because — provided the bank has a copy of the keys — it could still create all the messages it could receive from the customer. This is true for all symmetric authentication systems as was laid out in §3.1. In digital signature systems this would be different if the keys were created by the customer himself.

[Exercise at page 443.](#)

### 6-2 Personal relation and linkability of pseudonyms

Role-pseudonyms grant more anonymity than personal pseudonyms, because actions of a user that were done using different pseudonyms can not be linked. That way no information about a pseudonym can be collected which could make deanonymization possible. In the same way transaction-pseudonyms provide more anonymity than business-relation-pseudonyms because even actions that were committed in the same business relation can not be linked. Thus no information accumulates for a given pseudonym that excesses what is necessary for the transaction.

[Exercise at page 443.](#)

### 6-3 Self-authentication vs. foreign-authentication

Self-authentication means authorization by the one that is making a statement, e.g., by referring to an earlier statement he made. Authentication in this case means proving to be the same person who acquired a certain right back then. This can be done by signing a message according to a pseudonym (=test key of a digital signature system) that was used then. A typical example would be spending digital money (=transfer of an ownership right) that was given to s.o. using that pseudonym.

Foreign authentication means authorization by a third party, which for example signs an authorization statement containing the pseudonym (=test key of a digital signature system) in use. Grade sheets, driver's licenses, passports, ratings of creditworthiness, debit and credit cards are typical examples of authorization statements.

The reason why the difference is crucial is that very often the one giving the authorization accepts responsibility in case of abuse. Therefore, with foreign authentication, two parties are liable instead of one with self-authentication. This is important especially when designing protocols in which some principal can act anonymously (more precisely: pseudonymously).

[Exercise at page 444.](#)

### 6-4 Discussion about security properties of digital payment systems

In my opinion it is necessary that banks and vendors can not create consumer profiles of their clients. Additionally it would be desirable that they can not do so even with combined forces.

It would be great if media of exchange that were lost in hard-disk crashes or together with the hardware could be returned to the respective owners (so-called loss tolerance). This can be seen as part of the “he only loses a claim if he intends to” property.

#### Multilaterally secure digital payment system

- Security aim confidentiality

- Payment data should be kept secret from everyone except transaction partners (payer, payee, where app. bank, witness).
- Payer and/or payee should stay anonymous to each other and should not be observable by third parties (incl. payment system provider, bank, witness, ...).

- Security aim integrity

- Claims are only given up when intended.
- When a recipient for a claim has been specified by its owner, precisely that recipient receives the claim.
- The payer is able to provide evidence for a successful transaction to third parties when necessary.
- Even when cooperating, users can not increase the sum of their monetary claims.

## B Answers

- **Security aim availability**
  - Users can transfer any claims they received.

[Exercise at page 444.](#)

### 6-5 How far do concrete digital payment systems fulfill your protection goals?

- **Telephone Banking:** Security aims are not met at all with regard to confidentiality. Security aims integrity and availability are satisfied only if the bank is not suspected to be potentially malicious i.e., if there is no need for provability towards the bank. We also have to assume that phone lines can not be tapped, otherwise attackers can use passwords, TANs, PINs they overheard, or whatever else is used to identify customers and check their authorization to violate integrity by transferring other people's claims and thus keep their legitimate owners from accessing them — at least temporarily.
- **HBCI:** Using encryption, confidentiality can be achieved towards third parties, although of course not towards the bank or payer/payee. No anonymization provisions are provided for. When using the interim solution triple-DES as a hardware based signature substitute, integrity (and thus availability) is only guaranteed, if the bank is not seen as a potential attacker. When using RSA with key generation on the client's side, integrity and availability are obtained only as long as their computer works correctly.

[Exercise at page 444.](#)

## B.9 Answers for “Regulation of security technologies”

### 9-1 Digital signatures at “Symmetric authentication allows concealment”

Classically the answer would have been ‘no’, because everybody can test a signature and thus, whether authentication is valid, which enables them to detect the appropriate bits (or message fragments) and combine them.

To make the answer ‘yes’, precisely that needs to be prevented: everybody being able to test the signatures. These are ways to accomplish this:

- A digital signature system is used, but the test key is made available to the recipient only. This corresponds to the inclusion relation between digital signature systems and symmetric authentication systems depicted in fig. 3.12. The question whether we can still talk about digital signatures in this context can of course be argued.
- A special digital signature system is used, where digital signatures can only be tested by certain specified partners [Chau\_95] and only one partner is specified [Rive\_98].

[Exercise at page 444.](#)

## 9-2 “Symmetric authentication allows concealment” vs. Steganography

The procedure proposed by Shamir is somewhat similar to steganography in that the confidential message is embedded into an envelope bitstream and that confidentiality is ensured, because the message is not detected in the bitstream. The difference from good steganography is that it is possible for the observer to take notice of the anomaly of the bitstream: it contains all possible bit combinations. The aim of steganography — hiding the very existence of the secret message — therefore is not met by Rivest’s scheme.

[Exercise at page 444.](#)

## 9-3 Crypto-regulation by prohibition of freely programmable computers?

Even this does not do the trick:

At the very least people could come together to exchange one-time pads and for example encrypt e-mails by hand. Modular addition and subtraction is commonly known and not hard to do — at least the XOR operation is simpler than writing and could even be learned in kindergarten.

Also people could use Key-Escrow without retroactive decryption at a time when total observation is not yet allowed to exchange long-enough keys so they do not even have to meet in person.

[Exercise at page 444.](#)