

Contents

1 Einführung	10
1.1 Wozu Sicherheit?	10
1.2 Safety vs. Security	10
1.3 Wie viel Sicherheit?	11
1.4 Sicherheitsprobleme	11
1.5 Sicherheitssysteme	11
1.6 Wer sind die Angreifer?	12
1.7 Forensic Readiness	12
1.8 Sicherheitsziele	12
1.9 Vertraulichkeit/Geheimhaltung/Datenschutz	12
1.10 Anonymität	13
1.11 Integrität/Authentizität	13
1.12 Zurechenbarkeit/Verbindlichkeit	13
1.13 Verfügbarkeit	14
1.14 Wo liegen die Schwachstellen?	14
1.15 Designprinzipien für sichere Systeme	14
2 Zugriffskontrolle	16
2.1 AAA	16
2.2 Grundbegriffe: Zugriffskontrolle	16
2.3 Unterschiedliche Ebenen der Zugriffskontrolle	17
2.4 Referenzmonitor	17
2.5 Zugriffsmatrix (Lampson 1974)	17
2.6 Zugriffskontrolllisten	18
2.7 FreeBSD-ACLs (POSIX .1e)	20
2.8 NFSv4-ACLs	20
2.9 Windows-ACLs	20
2.10 Capabilities	22
2.11 Mandatory Access Control (MAC)	23
2.12 Bell-LaPadula-Modell (BLP)	24
2.12.1 Konzepte	24
2.13 Chinese Wall-Modell	26
2.13.1 Interessenkonflikte	26
2.13.2 Begrifflichkeiten	26

2.13.3	Simple Security Property	27
2.13.4	Chinese Wall-Modell: *-Property	28
2.13.5	Bewertung/Einordnung	28
2.14	Andere Modelle für Zugriffskontrolle	28
2.15	Domain-Type-Enforcement (DTE)	29
2.16	Rollenbasierte Zugriffskontrolle	29
2.16.1	RBAC gemäß ANSI-Standard	30
2.16.2	Formalisierung von RBAC gemäß ANSI-Standard	30
2.16.3	Komponenten von RBAC	31
2.16.4	Erweiterung von RBAC: Separation of Duty	32
2.16.5	Saperation of Duty (SoD)	32
2.16.6	Statische Separation of Duty	32
2.16.7	Dynamische Separation of Duty	33
2.16.8	Authorization Constraints	33
2.16.9	Einsatzgebiete von RBAC	33
2.16.10	Stärken	34
2.16.11	Schwächen	34
2.17	SELinux - Security Enhanced Linux	34
2.17.1	Ziele	35
2.17.2	Konzepte	35
2.17.3	Sicherheitsarchitektur	35
2.17.4	Abschließende Anmerkungen	36
2.18	Sandboxing	36
2.18.1	Java-Sandboxing im JDK 1.0 (veraltet)	37
2.18.2	Sandboxing in Android	37
3	Kryptographie	39
3.1	Symmetrische Verschlüsselung	39
3.2	Terminologie	40
3.3	Kryptoanalyse: Angriffsklassen	40
3.4	Historische Kryptographie	41
3.4.1	Transposition	41
3.4.2	Caesar-Chiffre	41
3.4.3	Kerckhoffs-Prinzip	42
3.4.4	Substitution	42
3.5	Perfekte Chiffren	43
3.6	Blockchiffren: DES und AES	44
3.6.1	Nutzung von Blockchiffren: Modes of Operation (Betriebarten)	44
3.7	Funktionsweise: ECB	45
3.8	CBC: Cipherblock Chaining	45
3.9	Padding	46
3.10	Weitere Stromchiffren-Modi	47
3.10.1	Funktionsweise: CTR	47
3.10.2	Angriffe gegen Stromchiffren	47
3.11	Schlüsselerzeugung	48

3.11.1	Qualität von Zufallszahlen	48
3.11.2	Entropie	48
3.11.3	Zufallszahlen	48
3.11.4	Block-Chiffre als Pseudo-Zufallszahlengenerator	49
3.12	Hash-Funktionen	50
3.12.1	Kryptographischer Hash	50
3.12.2	Geburtstagsparadoxon	50
3.12.3	Wir bauen uns einen Hash: Merkle-Damgård	52
3.12.4	Padding: Auffüllen auf Blockgröße	52
3.13	Message Authentication Codes (MAC)	55
3.13.1	Einsatz von Hash-Funktionen für MACs: HMAC	55
3.13.2	Exkurs: CBC-MAC (Message Authentication Code)	56
3.13.3	Kombination MAC/Verschlüsselung	56
3.13.4	Authenticated Encryption with Associated Data (AEAD)	56
3.14	Wie sicher ist mein Kryptosystem?	57
3.14.1	Levels of Security	57
3.15	Asymmetrische Kryptographie	59
3.15.1	Schlüsselverteilungsproblem	59
3.16	Asymmetrische (Public Key-) Verschlüsselung	59
3.16.1	Allgemeine Eigenschaften asymmetrischer Verfahren	60
3.16.2	Beispiele für asymmetrische Verfahren	60
3.17	Einweg-Funktionen	60
3.17.1	Einweg-Funktionen mit Falltür	61
3.17.2	Primitivwurzel	61
3.17.3	Problem des diskreten Logarithmus /1	62
3.17.4	Problem des diskreten Logarithmus /2	62
3.18	Schlüsselvereinbarung nach Diffie und Hellman	63
3.19	ElGamal-Verschlüsselungsverfahren	63
3.19.1	Funktionsweise: ElGamal-Verschlüsselung	64
3.20	Asymmetrische Kryptographie auf Basis elliptischer Kurven	64
3.20.1	Praktische Bedeutung von ECC	65
3.21	RSA-Algorithmus	65
3.21.1	RSA: Funktionsweise	66
3.21.2	Basis der Sicherheit von RSA: Schwierigkeit der Faktorisierung	66
3.21.3	Vorgriff: Einsatz von RSA für digitale Signaturen	66
3.22	Hybride Verschlüsselung	67
3.22.1	Ablauf: Hybride Verschlüsselung	67
3.23	Man-in-the-middle-Angriff	67
3.24	Anwendung der Kryptographie: E-Mail-Sicherheit	68
3.24.1	E-Mail-Sicherheit als Anwendung der Kryptographie	68
3.24.2	Sicherheitsprobleme im Zusammenhang mit E-Mails	68
3.25	PGP - Pretty Good Privacy	69
3.25.1	PGP - Implementierungen	69
3.25.2	Sicherheitsdienste von PGP	69
3.25.3	Schlüsselverteilung in PGP	70
3.25.4	Web of Trust	71

3.25.5	Vertrauensstufen für öffentliche Schlüssel in PGP	71
3.25.6	Berechtigung von Vertrauen in einen öffentlichen Schlüssel	71
3.25.7	Rücknahme von öffentlichen Schlüsseln	72
4	Digitale Signatur, Zertifikate, PKI, E-Mail-Sicherheit	73
4.1	Digitale Signaturen	73
4.1.1	Anforderungen an digitale Signaturen	73
4.1.2	Digitale Signatur mit RSA	74
4.1.3	Digitale Signatur mit ElGamal	74
4.1.4	Prinzip der digitalen Signatur	75
4.1.5	Wiederholung: Kryptographischer Hash	75
4.1.6	Funktionsweise: Bildung und Verifikation der digitalen Signatur	76
4.2	Digital Signature Algorithm (DSA)	76
4.2.1	DSA: Status	76
4.3	Zertifikate	77
4.3.1	Zertifikate vs. Schlüsselpaare	77
4.3.2	X.509 v3-Zertifikate	77
4.3.3	Ketten von Zertifikaten	78
4.3.4	Public Key Infrastructure	79
4.3.5	Fazit zu Signaturen, PKI	80
4.4	S/MIME - Secure Multipurpose Internet Mail Extension	80
4.4.1	Sicherheitsdienste von S/MIME	80
4.4.2	Authentisierung in S/MIME	81
4.4.3	Vergleich PGP - S/MIME	83
5	Authentisierung, Sicherheitsprotokolle	84
5.1	Authentisierung	84
5.2	Arten der Authentisierung	85
5.3	Authentisierung durch Passwörter	85
5.3.1	Angriffe auf Password-Systeme	85
5.3.2	Password-Cracking	86
5.3.3	Wahl guter Passwörter	86
5.3.4	NIST SP 800-63B (June 2017)	88
5.3.5	Sicherheit von Unix-Passwörtern	88
5.3.6	Zusammenfassung: Authentisierung durch Passwörter	89
5.4	Hardware-basierte OTP-Verfahren: ID-Token	89
5.5	Authentisierung durch Biometrie	90
5.5.1	Problembereiche	91
6	Schwachstellen, Bedrohungen, Angriffe	92
6.1	Sicherheitsprobleme	92
6.2	Verwundbarkeit und "Exploit"	92
6.3	CERTs und Advisories	93
6.4	Software im Netz	93
6.5	Buffer-Overflows	93

CONTENTS	5
6.5.1	Nicht korrekt geprüfte Eingaben 94
6.5.2	Call-Stack 94
6.5.3	Überschreiben der Rücksprungadresse 95
6.5.4	Präparation des Eingabe-Strings 96
6.5.5	Fallstricke für einen Angreifer 96
6.5.6	Gegenmaßnahme: NX-Bit 97
6.5.7	Gegenmaßnahme: Canaries 97
6.5.8	Gegenmaßnahme: ASLR 97
6.5.9	Fazit: Buffer-Overflows 98
6.5.10	Ausblick: Buffer-Overflows 98
6.6	SQL-Injection 98
6.6.1	...Angreifer gibt die Daten ein 99
6.6.2	Fazit 99
6.7	Cross Site Scripting (XSS) 99
6.7.1	Durchführung eines Angriffs 100
6.7.2	Beispiel für verwundbaren JSP-Server-Code 100
6.7.3	Fazit: XSS-Angriffe 100
6.8	Zwischenfazit 100
6.9	TOCTOU-Angriffe 101
6.10	Spoofing 101
6.11	(D)DoS-Angriffe 102
6.12	Syn-Flood-Angriff 102
6.12.1	TCP-Handshake 102
6.13	Trojanische Pferde 103
6.13.1	Weitere Varianten von Trojanischen Pferden/Malware . . 103
6.14	Social Engineering 104
6.15	Viren 104
6.15.1	Virentypen 105
6.15.2	Funktionsweise: Programmivirus 105
6.15.3	Gegenmaßnahme: Virensorter 105
6.16	Würmer 106
6.16.1	Gegenmaßnahme: Firewall 106
6.17	Können wir gewinnen? 106
6.18	Die vier Phasen eines Angriffs 107
7	Netz-Sicherheit 108
7.1	Software im Netz 109
7.2	Angriffe auf verschiedenen Schichten: Schicht 1 109
7.3	Angriff auf verschiedenen Schichten: Schicht 2 109
7.4	Angriffe auf verschiedenen Schichten: Schicht 3 110
7.4.1	Finding victims: Scanning 110
7.5	Angriffe auf verschiedenen Schichten: Schicht 4 111
7.5.1	RN1 Wiederholung: Three-way handshake 111
7.5.2	SYN-flood attack 112
7.5.3	Countering Resource Depletion 112
7.5.4	Example: TCP SYN Cookies 113

7.5.5	SYN Cookies	114
7.6	Angriffe auf verschiedenen Schichten: Schicht 7	114
7.6.1	Gemeinsamkeiten	114
7.7	The Internet Threat Model	115
7.7.1	Types of attacks	115
7.7.2	Combinations	116
7.8	Man-in-the-middle-Angriff = MITM = Janus-Angriff	116
7.8.1	On-path vs. off-path attacks	116
7.9	DoS-Angriffe	117
7.10	SQL-Slammer/Sapphire	117
7.11	Internet Background Radiation	117
7.12	Geschützte Werkstätten (Unternehmensnetze und Firewalls)	118
7.12.1	Classifying Traffic: The E(vil)-Bit	118
7.12.2	Firewall	119
7.12.3	Paketfilter	119
7.12.4	Stateful Packet Inspection	120
7.12.5	Aufgaben eines ALG	121
7.12.6	DMZ (Grenznetz)	121
7.12.7	NAT: kein Firewall	122
7.12.8	Logging	122
7.12.9	Der Einfluss von Firewalls auf das Internet	123
7.12.10	“firewall friendly”	123
8	Sicherheitsprotokolle (Theorie)	124
8.1	Sicherheitsprotokolle	124
8.2	Grundlegende Notation	124
8.3	Challenge-Response-Protokolle	125
8.3.1	Angriff auf das Mutual Challenge-Response-Protokoll	126
8.4	Einschub: Reflection-Angriff	126
8.4.1	Lösung des Problems	127
8.5	Das Denning-Sacco-Protokoll	127
8.5.1	Die einzelnen Schritte des Denning-Sacco-Protokolls	127
8.5.2	Der Angriff	128
8.5.3	Problembehebung	128
8.6	Zusammenfassung	128
8.7	Protokollanalyse mit Formalen Methoden	129
9	Sicherheitsprotokoll: TLS (SSL)	130
9.1	Transport Layer Security: SSL/TLS	130
9.2	SSL/TLS	130
9.2.1	Grundlegende Idee	130
9.2.2	Teilprotokolle	131
9.2.3	SSL/TLS-Handshake mit RSA: Vorausgesetzte Notationen und Bezeichnungen	131
9.2.4	SSL/TLS: Handshake mit RSA	132
9.2.5	SSL/TLS: Handshake mit RSA	133

9.2.6 Einschub: PRF	134
9.2.7 SSL/TLS mit RSA: Handshake (Part 2)	135
9.2.8 SSL/TLS-Handshake: Überblick	135
9.2.9 Ergänzung: SSL/TLS-Handshake mit DH	135
9.2.10 SSL/TLS: Handshake mit DH	136
9.2.11 Record-Protokoll	137
9.2.12 Alert-Protokoll	137
9.2.13 RFC 8446: TLS 1.3	137
10 Drahtlose Sicherheit	138
10.1 Überblick	138
10.2 Traditionelle Sicherung des Zugangsnetzes	138
10.3 Elemente der traditionellen Sicherheit für den Netzzugang	140
10.4 802.1X: Sicherung des Zugangsnetzes im Ethernet	140
10.5 Was ist zu schützen?	141
10.6 WLANs sind anders	141
10.6.1 WLAN-Sicherheit: Anforderungen (Sicht Universität)	141
10.6.2 WLAN-Sicherheit: Ansätze	142
10.6.3 WLAN-Zugangskontrolle: Warum 802.1X besser ist	142
10.6.4 WLAN-Zugangskontrolle: Warum ein VPN besser ist	143
10.6.5 WLAN-Zugangskontrolle: Warum Web-basiertes Filtern besser ist	144
10.6.6 Eingebaute Sicherheitsmechanismen	144
10.6.7 Verbergen der SSID	145
10.6.8 Filtern von MAC-Adressen	145
10.6.9 Verbinden mit dem AP	145
10.6.10 Wired Equivalent Privacy (WEP)	146
10.6.11 Angriff auf die Länge des IVs	148
10.6.12 FMS-Angriff (Weak IV-Angriff)	148
10.6.13 Zusammenfassung: WEP	148
10.7 Virtual Private Networks (VPN)	149
10.7.1 Probleme des VPN-Ansatzes	149
10.8 802.1X-Authentisierung	150
10.9 Extensible Authentication Protocol (EAP)	151
10.10 EAP-MD5	151
10.11 LEAP (Cisco Wireless)	151
10.12 EAP-TLS	152
10.13 EAP-TTLS (Bob Funk), PEAP von Microsoft und Cisco	152
10.14 WPA-Schritte (Enterprise)	153
10.14.1 802.1x Authentisierung + PMK	153
10.14.2 Situation nach erfolgreicher Durchführung von EAP	154
10.14.3 4-Way-Handshake und PTK	155
10.14.4 KRACK: Key Re-installment AttaCKs	156
10.14.5 Gruppenschlüssel	156
10.14.6 Hole 196	157
10.14.7 TKIP (Temporal Key Integrity Protocol)	157

10.14.8 Der MIC-Tradeoff	157
10.14.9 WPA-PSK	158
10.14.10 WPA3	158
10.15 Was ist beim Ausrollen eines WLANs zu berücksichtigen?	159
10.16 Fazit	159
11 Software Security	160
11.1 Sicherheitslücken in Software: Heartbleed Bug	160
11.2 Trinity of Trouble	161
11.3 Software Security als eigene Disziplin	161
11.4 SDL - Security Development Lifecycle	162
11.4.1 SDL von Microsoft	162
11.4.2 Seven Touchpoints nach McGraw	163
11.4.3 Wichtige Aktivitäten innerhalb eines SDLs	163
11.4.4 Wiederholung: Buffer-Overflows	163
11.4.5 Integer Overflows	164
11.4.6 Probleme mit der Speicherverwaltung	164
11.5 Code Review mittels statischer Programmanalyse	165
11.5.1 Phasen eines Compilers	165
11.5.2 Statische Programmanalyse	165
11.5.3 Statische Programmanalyse: SonarQube	166
11.5.4 Statische Programmanalyse: Clang	167
11.5.5 Clang Static Analyzer	168
11.5.6 Statische versus dynamische Programmanalyse	169
11.5.7 Dynamische Analyse mittels Code-Instrumentierung	169
11.6 Nicht nur Bugs, sondern auch Flaws	171
11.7 Architekturelle Risikoanalyse	171
11.7.1 Weitere Schritte der architekturellen Risikoanalyse nach McGraw	172
11.8 Penetrationstests	172
11.8.1 Software Penetrationstests - Werkzeuge nutzen	173
11.9 Fuzzing	173
11.9.1 Beispiel: american fuzzy lop	173
12 Sicherheitskriterien	175
12.1 Motivation: Sicherheitskriterien	175
12.2 Kriterienkataloge	175
12.3 Trusted Computer Evaluation Criteria - TCSEC	175
12.3.1 Die Hierarchiestufen von TCSEC	176
12.3.2 Bewertung: TCSEC	177
12.4 ITSEC	177
12.5 Die Common Criteria	178
12.5.1 Zusammenhang mit ITSEC	178
12.5.2 Struktur der CC-Dokumente	178
12.5.3 Funktionale Sicherheitsanforderungen	179
12.5.4 Vertrauenswürdigkeits-anforderungen	179

12.5.5 Vertrauenswürdigkeitsstudien	180
12.6 Schutzprofil	181
12.6.1 Bestandteile eines Schutzprofils	182
12.6.2 Erklärung - Rationale	182
12.6.3 Sicherheitsvorgaben	183
12.6.4 Vorgehensweise zur Evaluation	184
12.6.5 Zertifizierte Produkte	184
12.6.6 Abschließende Bemerkungen und Bewertung	185
12.7 Informationssicherheitsmanagement	186
12.8 BS 7799	186
12.8.1 Themenbereiche BS 7799	187
12.8.2 ISO 27001 / BS 7799	187
12.9 Security Management nach BSI-Grundschutz	188
12.9.1 BSI-Grundschutzkompendium	188
12.9.2 Schichtmodell	188
12.9.3 Vorgehensweise nach IT-Grundschutz	191
12.9.4 Infrastrukturanalyse	191
12.9.5 Schutzbedarfsfeststellung	192
12.9.6 Modellierung nach IT-Grundschutz	192
12.9.7 Basis-Sicherheitscheck	193
12.9.8 Ergänzende Risikoanalyse	193
12.9.9 Planung der Realisierung	193
12.9.10 Zusammenfassung: Vorgehensweise nach GS-Grundschutz	194
12.9.11 Grundschutz-Zertifizierung	194
12.9.12 Einordnung	195
12.10 Vergleich CC/BSI GS/BS 7799	195

Chapter 1

Einführung

1.1 Wozu Sicherheit?

- Erwartung an IT-Systeme: **Verlässlichkeit**
 - Immer mehr, immer wichtigere Aufgaben werden IT-Systemen übertragen
- Problem: Bugs, Abstürze, Fehlfunktionen, Naturkatastrophen
- Problem: Böse Absicht (aber auch böse Zufälle)

Start vereinfacht:

- Sicherheit (**Safety**): System hat von sich aus keine Fehlfunktionen
- Sicherheit (**Security**): Sicherheit gegen böse Absicht

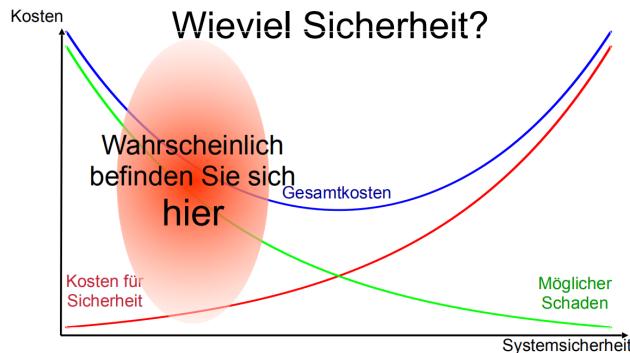
1.2 Safety vs. Security



- Nicht immer klare Trennung
- Sicherheit gegen Angriffe (Security) liefert oft auch Sicherheit gegen unbeabsichtigte Fehlbedienung

- Betriebssicherheit (Safety) vs. Funktionssicherheit
- Gelegentlich Widerspruch (Beispiel Fluchttür)

1.3 Wie viel Sicherhet?



1.4 Sicherheitsprobleme

- Für ein System bestehen Sicherheitsziele (security objectives)
- Sicherheitssysteme haben Schwachstellen (weaknesses)
- Verwundbarkeiten (vulnerabilities) erlauben das Umgehen (oder den Missbrauch) von Sicherheitsmechanismen
- Eine Bedrohung (threat) ist die Möglichkeit eines Angriffs (attack)
- Angriffe erzeugen u.U. Schaden (damage)
- Risiko (Risk) = $p(\text{attack}) \times \text{cost}(\text{damage})$

1.5 Sicherheitssysteme

- Erfolgreiche Angriffe
 - Verhindern (prevention)
 - Erkennen (detection)
 - Eingrenzen (Schadensbegrenzung) (containment)
- Sicherheitsregeln (security policy)
 - Richtlinien; Schulung der Mitarbeiter
 - Notfallplanung, -training
 - Management-Unterstützung, Schutz der Sicherheitsverantwortlichen

1.6 Wer sind die Angreifer?

- **Insider** (faul, anders fokussiert, frustriert, kriminell)
 - Eventuell als Folge von **Social Engineering**
- **”Hacker”** (richtiger: Cracker), ”script kiddies”
- **Professionelle** Angreifer (Spionage, Geheimdienste)
- Organisiertes **Verbrechen**
 - Z.B. Erpressung
 - Z.B. Ausschalten eines Kokurrenten, Wirtschaftsspionage
 - Z.B. Beschaffung einer Plattform für weitere Angriffe

1.7 Forensic Readiness

- Prevent
- Detect
- Contain
- **Prosecute** (or at least fend off the inevitable lawsuits)

1.8 Sicherheitsziele

- **Vertraulichkeit**/Geheimhaltung/Datenschutz - Anonimität
- **Integrität**/Authentizität
- Zurechenbarkeit/Verbindlichkeit
- **Verfügbarkeit**

1.9 Vertraulichkeit/Geheimhaltung/Datenschutz

- Vertraulichkeit (confidentiality): Verpflichtung zur Geheimhaltung der Informationen anderer
- Geheimhaltung (secrecy): Einschränkung des Zugriffs
- Datenschutz (privacy): Recht auf Schutz eigener (persönlicher) Informationen.

Achtung: Oft ist die Tatsache einer Kommunikationshandlung bereits geheimzuhalte Information (vs. traffic analysis)

1.10 Anonymität

Anonymität (anonymity): Durchführung von Handlungen ohne Preisgabe der Identität:

- Eventuell auch Preisgabe eines Pseudonyms
- Anonymität = Vertraulichkeit der Identität

1.11 Integrität/Authentizität

- **Integrität (integrity)** der Daten: Schutz vor unautorisierte und unbemerkter Veränderung von Daten (vgl. Integritätsbegriff aus den Datenbanken). Beispiel: Kontendaten in einer Bank.
- **Authentizität (authenticity):** Information ist integer und frisch; eindeutig einer Identität zuzuordnen.

1.12 Zurechenbarkeit/Verbindlichkeit

- Zurechenbarkeit (accountability): Eine durchgeführte Handlung kann einem Kommunikationspartner eindeutig zugeordnet werden.
- Verbindlichkeit (3rd-party verifiability, 'non-repudiation'): kein unzulässiges Abstreiten durchgeföhrter Handlungen. Notwendige Beispiele für:
 - Abschließen von elektronischen Kaufverträgen
 - Digital unterschriebene Gerichtsanträge

	A	B	C (Dritter)
Authentisierung	Will → Weiß		—
Zurechenbarkeit	(...)	Weiß	—
Verbindlichkeit	(...)	Will → Weiß	

1.13 Verfügbarkeit

- Verfügbarkeit (availability): Schutz vor unbefugter Beeinträchtigung der Funktionalität von Komponenten, Diensten etc.
 - vs. Denial-of-Service (DoS-) Angriffe
- Ergibt zusammen mit Korrektheit: Verlässlichkeit (dependability): Funktionssicherheit; zuverlässige Erbringung der Funktion (reliability)

Eselsbrücke: CIA

- Confidentiality
- Integrity
- Availability

1.14 Wo liegen die Schwachstellen?

- Schlechtes Design: z.B. fehlende Kontrollen, zu grobe Rechtevergabe
- Schlechte Implementierung: z.B. Pufferüberläufe, schwache Mechanismen, Umgehungswege
- Schlechte Systemadministratoren: z.B. Account mit Standartpasswort, offene Ports im Firewall, Einsatz ungeeigneter Systeme und Werkzeuge
- Schlechtes Management: z.B. unklare Sicherheitspolitik, unklare Sicherheitsregeln, fehlendes Sicherheitsbewusstsein der Mitarbeiter, keine Mittel für Sicherheitsüberprüfungen

1.15 Designprinzipien für sichere Systeme

Saltzer, Schroeder (1975):

- **Principle of Economy of Mechanism:** The protection mechanism should have a simple and small design.
- **Principle of Fail-safe Defaults:** The protection mechanism should deny access by default, and grant access only when explicit permission exists.
- **Principle of Complete Mediation:** The protection mechanism should check every access to every object.
- **Principle of Open Design:** The protection mechanism should not depend on attackers being ignorant of its design to succeed (no **security by obscurity**). It may however be based on the attacker's ignorance of specific information such as passwords or cipher keys.

- **Principle of Separation of Privilege:** The protection mechanism should grant access based on more than one piece of information.
- **Principle of Least Privilege:** The protection mechanism should force every process to operate with the minimum privileges needed to perform its task.
- **Principle of Least Common Mechanism:** The protection mechanism should be shared as little as possible among users.
- **Principle of Psychological Acceptability:** The protection mechanism should be easy to use (at least as easy as not using it).
- **Principle of Defence in Depth:** There should be multiple layers of defense before a high-value target is compromised (No Maginot lines).
- **Principle of Securing the Weakest Link:** The protection mechanism should not have weak spots that allow circumventing the well-secured parts. (Security often is a chain)
- **Principle of Reluctance to Trust:** The protection mechanism should not give unwarranted trust to any mechanism or entity. (Healthy skepticism)

Chapter 2

Zugriffskontrolle

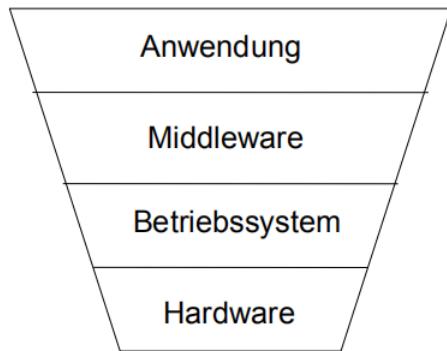
2.1 AAA

- Zugriffskontrolle (**access control**) regelt: "**Wer** darf in einem IT-System auf **welche** Ressourcen **wie** zugreifen".
- Alternativer Begriff: Autorisierung (**Authorisation**)
 - AAA = Authentication, **Authorisation**, Accounting (Authentisierung, **Authorisierung**, Abrechnung)

2.2 Grundbegriffe: Zugriffskontrolle

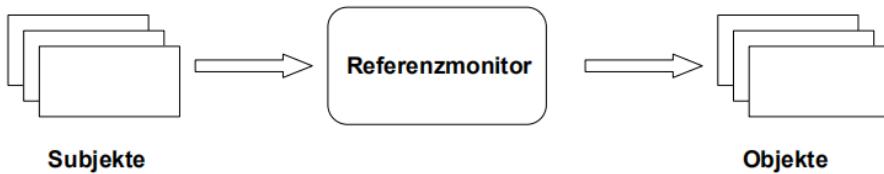
- Subjekt (**subject**): Initiator der Handlung
 - Benutzer (z.B. Alice, Bob), Gruppen, Prozesse , Rechner/Geräte
 - Genauer: **principal** als IT-Realisierung einer **person** oder Funktion
- Objekt (**object**): Gegenstand der Handlung, Ressource
 - Dateien, SQL-Tabellen, Konto, Prozesse
 - Achtung: Subjekte können auch Objekte sein
- Berechtigung (**permission**) bzw. Zugriffsrecht (**access rights**):
 - Was genau darf gemacht werden
 - Z.B.: lesen, schreiben, ausführen, löschen

2.3 Unterschiedliche Ebenen der Zugriffskontrolle



- Abhängigkeit der höheren Ebenen von den Zugriffskontrollmechanismen der niedrigeren Ebenen
- Zugriffskontrollmechanismen sind in höheren Ebenen komplexer
- Fehler in niedrigeren Ebenen können zum Umgehen der Sicherheitsmechanismen der höheren Ebenen führen

2.4 Referenzmonitor



Principle of Complete Mediation: The protection mechanism should check every access to every object.

2.5 Zugriffsmatrix (Lampson 1974)

- Zugriffskontrolle wird in ihrer grundlegenden Form dargestellt durch eine Zugriffskontrollmatrix (**access control matrix**) M mit:

$$M : S \times O \rightarrow 2^R$$

wobei **S** die Menge der Subjekte, **O** die Menge der Objekte und **R** die Menge der Zugriffsrechte wie z.B. read, write und execute sind.

- Subjekt s darf Objekt o genau dann lesen, wenn $\text{read} \in M(s, o)$

Beispiel für eine Zugriffsmatrix:

Subjekt/ Objekt	Datei 1	Datei 2	Datei 3	Prozess 1
Alice	{read, write}		{write}	
Bob		{execute}		{suspend}
Janet		{read}		
Frank	{read}			

$M(\text{Alice}, \text{Datei1}) = \{\text{read}, \text{write}\}$

2.6 Zugriffskontrolllisten

- Zugriffsmatrix ist oft dünn besetzt, d.h. eine vollständige Speicherung ist zu speicherintensiv $O(|S| \times |O|)$
- **Zugriffskontrolllisten (access control lists, ACLs):** Realisierung der Matrix nach Spalten
- Subjekte und deren Zugriffsrechte werden **beim Objekt** gespeichert
- **Beispiel:** $\text{ACL}(\text{Datei1})=((\text{Alice}, \{\text{read}, \text{write}\}), (\text{Frank}, \{\text{read}\}))$

Subjekt/ Objekt	Datei 1	Datei 2	Datei 3	Prozess 1
Alice	{read, write}		{write}	
Bob		{execute}		{suspend}
Janet		{read}		
Frank	{read}			

- Am häufigsten eingesetztes Konzept für die Zugriffskontrolle im gängigen Betriebssystemen (Windows NT(=2000/XP), Unix/Linux)
- Vorteile:
 - Vergebene **Zugriffsrechte** sind effizient für **Objekte bestimmbar**
 - **Rechterücknahme** (pro Objekt) ist meist effizient realisierbar
 - Einfach zu implementieren
- Nachteile:
 - Bestimmen der **Subjekt-Rechte** i.d.R. **sehr aufwendig** (alle Objekte im System untersuchen)
 - Aufwendige ACL-Kontrollen bei jedem Zugriff
 - **Schlechte Skalierbarkeit** bei sehr dynamisch wechselnder Menge von Subjekten, falls ACLs für **einzelne Subjekte/Nutzer** vergeben werden.

Beispiel: Zugriffskontrolle in Unix

- Subjekte:
 - Benutzer: User-ID, Group-ID, supplementary Group IDs (bis zu 16)
 - Prozess: User-ID/Group-ID (effective, real, saved), supplementary Group IDs
- Dateien:
 - Owner (User-ID), Group (Group-ID)
 - rwx-bits für: Owner, Group, Other
 - sUid, sGid: Rechteveränderungen bei Dateiaufruf
- Verzeichnisse:
 - Ebenso, plus:
 - t-Bit (nur der Eigentümer darf Dateien in diesem Verzeichnis löschen)
 - sGid-Bit (steuert Gruppe bei Datei-Erzeugung)
- Schutzbüts sind einfache Form von ACL
 - Informationen für die Zugriffskontrolle den Dateien zugeordnet
- Schutzbüt-ACL so nicht feingranular genug:
 - Alive kann nicht **Bob allein** nur Lese-Rechte auf eine bestimmte Datei geben
 - Uni Bremen: "grp" zur Einrichtung von Gruppen durch Benutzer
- FreeBSD, Solaris haben zusätzliches ACL-Konzept
- POSIX-Kommandos: getfacl, setfacl
 - Und chmod-Erweiterungen: +a/-a/=a

2.7 FreeBSD-ACLs (POSIX .1e)

- Menge von Tripeln Typ:Name:Rechte
 - user::rwx (Dateieigner) 1
 - group::r-x (Gruppe der Datei) 2
 - other::r-x (Der Rest) 5
- Neu:
 - user:fritz:r-x (spezifischer Benutzer) 2
 - group:katzen:rwx (spezifische Gruppe) 4
- Einschränkungen über **Maske** (chmod-Kompatibilität)
 - mask::r-x (schränkt alles außer 1 und 5 ein)

2.8 NFSv4-ACLs

- Allgemeiner Nachfolger der POSIX-ACLs
- Liste von ACEs (Access Control Entry)
 - 17 permission bits, 4 inheritance flags
- Triviale ACL: Wie traditionelles UNIX

```
$ ls -v file.1
-r--r--r-- 1 root      root      206663 May  4 11:52 file.1
  0:owner@:write_data/append_data/execute:deny
  1:owner@:read_data/write_xattr/write_attributes/write_acl/write_owner:allow
  2:group@:write_data/append_data/execute:deny
  3:group@:read_data:allow
  4:everyone@:write_data/append_data/write_xattr/execute/write_attributes
    /write_acl/write_owner:deny
  5:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
$
```

- Kommandos: ls -v, chmod -A

2.9 Windows-ACLs

- Zugriffsrechte auf Dateien, Ordner oder Drucker werden mit **Access Control Lists** verwaltet.
- Pro **Objekt**:
 - Security identification number (SID) als Bezeichner für **Eigner** (owner)

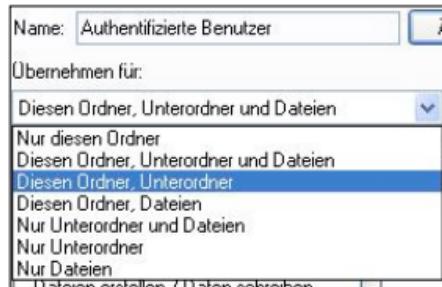
– **ACL**

- ACL besteht aus einzelnen Access Control Entries (ACEs):
 - Typ der ACE (Erlauben, Verweigern, Überwachen)
 - Subjekt (Benutzer oder Gruppe), für den/die der ACE gilt
 - * SID als Bezeichner für Subjekt
 - Berechtigungen, für die der ACE gilt (Bitmaske)
 - Vererbbarkeit der ACE auf Unterobjekte

- ACL:

Berechtigungseinträge:					
Typ	Name	Berechtigung	Geer...	Übernehmen für	
Zulassen	Administratoren (HE...)	Vollzugriff	D:\	Diesen Ordner, Untero...	
Zulassen	SYSTEM	Vollzugriff	D:\	Diesen Ordner, Untero...	
Zulassen	Seth (HELLBOY\S...	Vollzugriff	D:\	Nur diesen Ordner	
Zulassen	ERSTELLER-BESI...	Vollzugriff	D:\	Nur Unterordner und ...	
Zulassen	Benutzer (HELLBO...	Lesen, Ausfü...	D:\	Diesen Ordner, Untero...	
Zulassen	Benutzer (HELLBO...	Speziell	D:\	Diesen Ordner, Untero...	

- Vererbbarkeit:



- Besitzer hat immer Vollzugriff
 - Besitzer kann auch eine Gruppe sein
- Alle ACEs werden in Reihenfolge ausgewertet

Berechtigungen	Zulassen	Verweigern
Ordner durchsuchen / Datei ausführen	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Ordner auflisten / Daten lesen	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Attribute lesen	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Erweiterte Attribute lesen	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Dateien erstellen / Daten schreiben	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Ordner erstellen / Daten anhängen	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Attribute schreiben	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Erweiterte Attribute schreiben	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Unterordner und Dateien löschen	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Löschen	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Berechtigungen lesen	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Berechtigungen ändern	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Besitzrechte übernehmen	<input checked="" type="checkbox"/>	<input type="checkbox"/>

- GUI: Deny hat Vorrang vor Allow-Regeln
 - **Vorsicht:** Verbietet man einer Gruppe alles, gibt es keine Möglichkeit, Einzelnen Rechte zu geben!

2.10 Capabilities

- Zeilenweise Realisierung der Matrix
- Capability, Zugriffsticket mit Objekt-ID und Rechtebits
- Capability-Besitz berechtigt zur Wahrnehmung der Rechte
- Für jedes Subjekt s eine Capability-Liste (CList)
Beispiel: Clist (Alice) = ((Datei1, {read,write}), (Datei3, {write}))

Subjekt/ Objekt	Datei 1	Datei 2	Datei 3	Prozess 1
Alice	{read, write}		{write}	
Bob		{execute}		{suspend}
Janet		{read}		
Frank	{read}			

- Beispiele von Systemen mit Capability-Konzept: IBM System/38, Mach- μ -Kern (Ports), Amoeba

- UNIX: File Descriptor = flüchtige Capability
 - Deskriptor-Tabelle in User-Structure = C-List
 - Übergabe von Deskriptoren über UNIX-Domain-Sockets
- Achtung: **Keine** Capabilities in diesem Sinne: "POSIX Capabilities" (z.B. von Linux 2.2 unterstützt)
 - Besseres Wort: Privileges
- Renaissance des Capability-Konzeptes durch Zertifikate (X.509, später in VL)

Vorteile:

- Einfache Bestimmung der Subjekt-Rechte
- Einfache Zugriffskontrolle: nur noch Ticketkontrolle!
- Einfache Delegation von Rechten anderer Benutzer/Subjekte

Nachteile:

- Rechterücknahme schwierig (Ticket aus der Hand)
- Keine Subjekt-Ticket Kopplung, Besitz berechtigt automatisch zur Wahrnehmung der Rechte
- Objekt-Sicht auf Rechte schwierig: Wer darf was?

Bis hierher: DAC (Discretionary Access Control)

Bislang:

- **Subjekte** entscheiden über die Zugriffsberechtigungen (vgl. chmod in UNIX/Linux oder Dialogsgenster zur Rechtevergabe in Windows 2000/XP)
- Subjekt bestimmt weitergehend die Security Policy für "seine" Objekte (**benutzerbestimmte Zugriffskontrolle bzw. discretionary access control**)

2.11 Mandatory Access Control (MAC)

- In besonders sicherheitskritischen Bereichen mit z.B. hohen Anforderungen an die Vertraulichkeit ist benutzerbestimmte Zugriffskontrolle nicht angemessen
 - Typische Beispiele: Militär, Geheimdienste
- Einführung geeigneter Modelle, die **vom System** umgesetzt werden
- Systembestimmte Zugriffskontrolle (**Mandatory Access Control**)
- Bekanntestes Beispiel für Mandatory-Access-Control: **Bell-LaPadula-Modell**

2.12 Bell-LaPadula-Modell (BLP)

- David Bell und Len LaPadula, 1973: auf Initiative der US Air Force
- Zweck:
 - Einfache Sicherheitsmechanismen für die Verifikation
 - Formales Sicherheitsmodell
 - Beschränkung des Informationsflusses, d.h. Sicherheitsziel ist die **Vertraulichkeit**

2.12.1 Konzepte

Ein vereinfachtes BLP-Modell:

- Multilevel Security System (MLS)
 - Alle Subjekte und Objekte erhalten eine Sicherheitsmarkierung (**label**) gemäß der Stufe ihrer Vertraulichkeit wie z.B. top secret ; secret ; confidential ; unclassified
 - Idee: Höher klassifizierte Daten dürfen nur von Mitarbeitern mit einer ausreichenden Sicherheitsstufe gelesen werden
- Formalisierung der Idee:
 - **M** sei die Zugriffsmatrix
 - **SC** sei eine Menge von Sicherheitsmarkierungen, und auf **SC** gelte eine partielle Ordnung \leq
 - $o \in O : SC(o) \in SC$ sei die Sicherheitsmarkierung von Objekt o (**classification**)
 - $s \in S : SC(s) \in SC$ sei die Sicherheitsmarkierung von Subjekt s (**clearance**)
 - **Simple Security-Property (no read-up):**
 $\forall s \in S. \forall o \in O : \text{read} \in M(s, o) \Rightarrow SC(o) \leq SC(s)$
- Die no-read-up-Regel reicht nicht: **Trojanische Pferde** könnten höher klassifizierte Daten einfach nach unten schreiben
- Die entscheidende Neuerung des BLP-Modells ist mithin die zweite Regel: ***-Property (no-write-down):**
 $\forall s \in S. \forall o \in O : \text{write} \in M(s, o) \Rightarrow SC(s) \leq SC(o)$
- Manchmal fordert man auch, dass sich die Sicherheitsmarkierungen weder für Objekte noch für Subjekte während der Systemlaufzeit ändern (Strong Tranquility Property)

Beispiele: BLP

- Viele Betriebssysteme bieten BLP-Erweiterungen: u.a. Sun Trusted Solaris 7, Trusted HP-Unix, Linux-Derivate
 - Erweiterter Referenz-Monitor zur Überprüfung der BLP-Regeln
- Daten-Diode (-"Pumpe")
 - Daten können über ein Kommunikations-Device (Daten-Diode) **von Low nach High** transferiert (gepumpt) werden, aber nicht in der umgekehrten Richtung
 - Beispiel: NRL-Pump (US Naval Research Laboratory), 1993

Fazit: BLP

- Stärken:
 - * Einfach zu Implementieren (Anpassung des Referenzmonitors)
 - * Formales Modell (man kann Sätze über Eigenschaften beweisen)
 - * Gut geeignet zum Nachbilden hierarchischer Informationsflüsse: z.B. beim Militär, Geheimdienst
- Schwächen:
 - * Beschränkte Ausdrucksfähigkeit: keine Integrität (blindes Schreiben)
 - * Keine Modellierung von verdeckten Kanälen (**covert channels**), über die Informationen dann doch unberechtigt fließen können.

Verdeckte Kanäle

- Zugriffskontrolle schützt den Datencontainer, nicht die Information
 - Möglichkeit von verdeckten Kanälen
- Verdeckte Kanäle (**covert channels**):
Informationsfluß über einen Kanal, der nicht explizit zur Informationsübertragung vorgesehen ist

Beispiele:

- Ressourcen-Konflikt: High-Level-Prozess erzeugt eine Datei, so dass ein Low-Level-Prozess eine Fehlermeldung erhält, wenn die Datei mit gleichem Namen erzeugt wird (1 Bit Information)
- High-Level-Prozess kann gemeinsam genutzte Ressourcen (Position des Festplattenkopfes, Inhalt des Caches) in einem Zustand hinterlassen, so dass der Low-Level-Prozess anhand von Antwortzeiten zusätzliche Informationen gewinnen kann

Wiederholung: MAC und BLP

- MAC: Systembestimmte Zugriffskontrolle (mandatory access control)
- Bekanntestes Beispiel für MAC: **Bell-LaPadula**-Modell (BLP)
- Idee:
 - Klassifizierung von Subjekten und Objekten mit **Labels**
 - **Simple Security-Property** (No read-up-Regel)
 - ***-Property** (No write-down-Regel)

Weiteres MAC-Beispiel: Chinese Wall-Modell

2.13 Chinese Wall-Modell

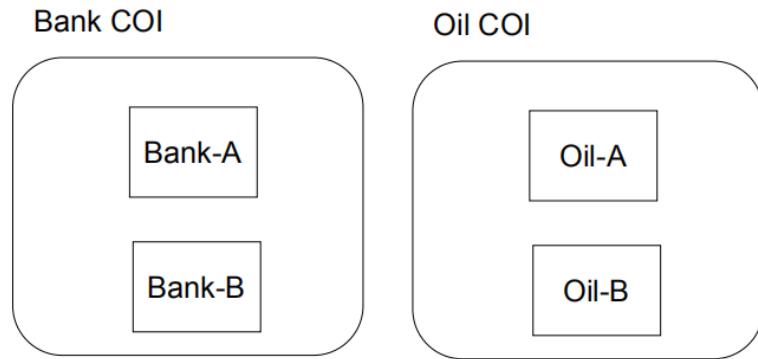
- Brewer und Nash, 1989
- Modell aus dem Finanzbereich
- Ausgangssituation:
 - Berater sind für verschiedene Unternehmen/Organisationen (z.B. Banken, Ölfirmen) tätig
 - Insiderwissen darf nicht genutzt werden

2.13.1 Interessenkonflikte

- Vermeidung von **Interessenkonflikten** (**conflict of interest, COI**): Ein Berater darf nicht mehrere Firmen/Organisationen aus derselben **Konfliktklasse** (**COI class**) beraten
- Beispiele für Konfliktklassen:
 - Beispiele: Banken, Ölfirmen

2.13.2 Begrifflichkeiten

- Drei Arten von Begrifflichkeiten (hierarchische Strukturierung der Daten):
 - 1. **Objekte** wie in BLP, Zugriffsrechte read, write
 - 2. Objekte gehören eindeutig zu Firmen/Organisationen
 - * Menge der Firmendaten: **Company Data Set (CD)**
 - * **CD(o)** gibt das eindeutige Company Data Set von Objekt **o** an
 - 3. Firmen/Organisationen gehören eindeutig **Konfliktklassen (COIs)** an
 - * COIs enthalten also die CDs der dazugehörigen Firmen/Organisationen

Beispiel**2.13.3 Simple Security Property****• Simple Security Property:**

Ein Berater darf auf ein Objekt o lesend zugreifen, nur falls eine der folgenden Bedingungen erfüllt ist:

- 1. Der Berater hat schon auf andere Objekte dieser Firma/Organisation zugegriffen oder
- 2. Das Objekt gehört zu einer Konfliktklasse, für die der Berater noch kein Objekt gelesen hat
- Berücksichtigung der **Zugriffshistorie**; Aufbau einer Chinese Wall

Beispiel**• Ausgangssituation**

- Bank-A und Bank-B
- Oil-A und Oil-B
- Alice und Bob sind Berater
- Annahme: Bob berät Bank-B und hat noch nie eine Ölfirma beraten

• Auf die folgenden Objekte darf Bob dann noch lesend zugreifen:

- Alle Objekte von Oil-A oder Oil-B
- Alle Objekte von Bank-B (aber nicht von Bank-A)

Reicht die Simple Security-Regel aus?

- Beispiel:
 - Bob berät Bank-B und Oil-A;
 - Alice berät Bank-A und Oil-A.
 - Alice hat insbesondere auch Schreibrechte auf Objekte aus Oil-A.
 - Simple Security-Regel gilt offenbar.
- Ein Trojanisches Pferd könnte Informationen über Bank-A von Alice nach Oil-A kopieren.
 - Bob hat nun mittels Oil-A Zugriff auf diese Informationen.
 - Bob hat also auf Objekte aus Bank-B und (indirekt) Bank-A Zugriff.
- Vergleicdh mit der Problematik "Verdeckte Kanäle"
- Einführung einer neuen Regel erforderlich

2.13.4 Chinese Wall-Modell: *-Property

- ***-Property:** Ein Berater hat schreibenden Zugriff auf ein Objekt **o**, wenn er bislang nur Objekte aus der Firma von Objekt **o**, aber keine anderen gelesen hat.
- Für unser Beispiel bedeutet dies: Alice darf **nicht** Objekte in Klasse Oil-A schreiben, weil sie lesenden Zugriff auch auf Objekte aus Bank-A besitzt.

2.13.5 Bewertung/Einordnung

- Chinese Wall-Modell zugeschnitten auf bestimmte Problematik aus dem Finanzbereich
- Nur Vertraulichkeit, keine Integrität
- Umsetzung durch organisatorische Maßnahmen und weniger durch IT-Systeme

2.14 Andere Modelle für Zugriffskontrolle

- **Biba-Modell** (Ken Biba, 1977), Integrität (aber kein Informationsfluß)
 - Ab MS-Vista enthält Windows Biba-Konzepte (z.B. Beschränkung von Änderungen von Registry-Einträgen)
- **Clark-Wilson** (David Clark, David Wilson, 1987): Sicherheitsmodell für Banken- und Buchhaltungsprogramme

- **Non-Interference-Modelle** (z.B. Meseguer, Goguen, 1982): → Verdeckte Kanäle
- Domain Type Enforcement

2.15 Domain-Type-Enforcement (DTE)

- Beobert und Kain, 1985
- **Idee:**
 - Zuordnung von Objekten (z.B. Dateien, Sockets, Prozesse) zu **Typen**
 - Zuordnung von Subjekten (z.B. Nutzer, Prozesse) zu Domains
- Zusätzliche Zugriffsmatrix
 - nicht für einzelne Objekte und Subjekte, sondern für **Typen** und **Domains**

Beispiel: $M(\text{admin_d}, \text{writeable_t}) = \{\text{create, write, read, search}\}$

- Vertraulichkeit und Integrität
- Anwendung: SELinux

2.16 Rollenbasierte Zugriffskontrolle

- **Idee der rollenbasierten Zugriffskontrolle (role-based access control, RBAC):**
Knüpfung der Rechte nicht mehr direkt an Benutzer, sondern an **Rollen (roles)**
- Rollen entsprechen häufig **Funktionen/Aufgaben**, die in einer Organisation (Krankenhaus, Bank, Behörden, Unternehmen) auftreten
- Geeignet für hierarchisch aufgebaute Organisationen
- Andere Form von MAC
- Beispiel: Bank
 - Kassierer, Kassenprüfer, Zweigstellenleiter
- State-of-the-Art!

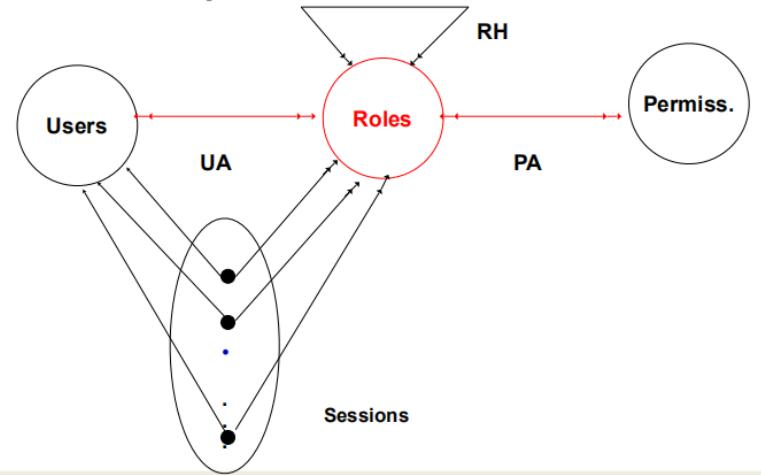
2.16.1 RBAC gemäß ANSI-Standard

- ANSI-Standard für RBAC (ANSI-INCITS 359-2004)
- Entitäten des ANSI-Standards für RBAC:
 - **Benutzer (users)** sind für Personen (und keine Prozesse, vgl. Subjekt-Begriff)
 - **Zugriffsberechtigung (permissions)** sind paare (Operation,Subjekt). Beispiele: (read,File1), (read,File2), (debit, account1), (credit, account2)
 - Eine **Rolle** ist eine Sammlung von Zugriffsberechtigungen
- Rollen werden Benutzern zugewiesen. Eine Rolle ist also Mittler zwischen Benutzern und Berechtigungen.
- Rollen werden von Benutzern in **Sitzungen (sessions)** aktiviert.
 - Unterscheidung von Rollenzuweisung und -aktivierung. Welches Sicherheitsprinzip?
 - Least Privilege
- Formalisierung?

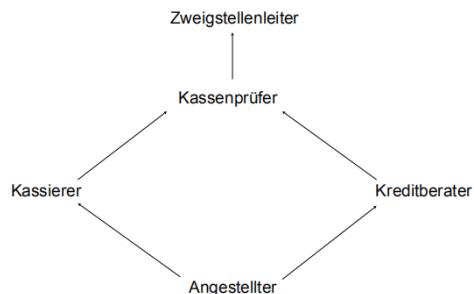
2.16.2 Formalisierung von RBAC gemäß ANSI-Standard

- **Mengen:**
 - Users, Roles, Permissions, Sessions
- **Relationen:**
 - $UA \subseteq \text{Users} \times \text{Roles}$ **user assignment**
Zuordnung von Benutzern zu Rollen
 - $PA \subseteq \text{Permissions} \times \text{Roles}$ (**permission assignment**)
Zuordnung von Zugriffsberechtigungen zu Rollen
 - RBAC soll Organisationsstrukturen nachbilden, also Einführung von Rollenhierarchien (**role hierarchies**):
 $RH \subseteq \text{Roles} \times \text{Roles}$, RH ist eine partielle Ordnung auf $\text{Roles} \times \text{Roles}$
- **Funktionen:**
 - session_user : Sessions → User - eindeutiger Besitzer einer Session
 - session_roles → 2^{Roles} - die in einer Session aktivierten Rollen

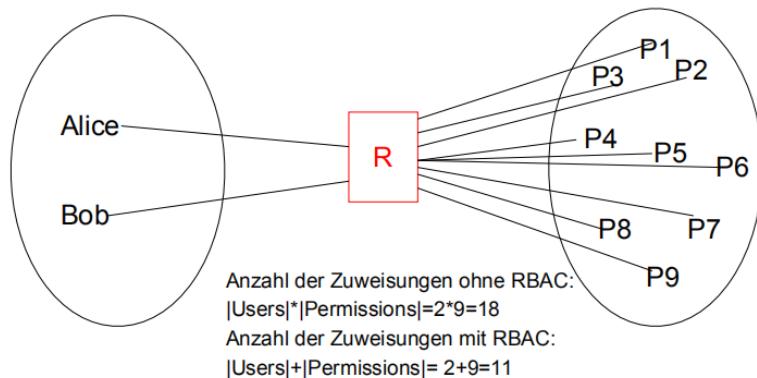
2.16.3 Komponenten von RBAC



Rollenhierarchie: Beispiel



Warum vereinfacht RBAC das Berechtigungsmanagement?



2.16.4 Erweiterung von RBAC: Separation of Duty

Motivation

- Niedergang der Barings-Bank, 1995

”In a fatal mistake, the bank allowed Leeson to remain Chief Trader while being responsible for settling his trades, a job that is usually split. This had made it much simpler for him to hide his losses.”

”Der Starhändler musste keine Rechenschaft über seine Transaktionen geben, eine Compliancestelle oder eine eigene Abwicklungsabteilung gab es nicht. Im Grunde kontrollierte sich Leeson weitgehend selbst.” ARD, 27.02.2020, 25 Jahre später

- Ähnlicher Fall, aber mit etwas geringeren Folgen: Betrug bei der Société Générale, Jérôme Kerviel, 2008

2.16.5 Saperation of Duty (SoD)

- Separation of Duty (SoD) - **Aufgabentrennung**
 - Manchmal auch ”Vieraugen-Prinzip” genannt
- Natürliche Abbildung von **organisationsinternen Kontrollregeln** mittels RBAC
- Beispiele für die Aufgabentrennung:
 - Die Rollen **Kassenprüfer** und **Kassierer** dürfen nicht von ein und derselben Person angenommen werden.
 - Trennung der Rollen **Kunde** und **Kreditberater**

2.16.6 Statische Separation of Duty

- Im RBAC-Formalismus:

CR (Menge von Konfliktrollen) - z.B. CR={Kassierer, Kassenprüfer}

roles(u) = { $r \in R \mid (u, r) \in UA$ } - Rollen von u

Statische Separation of Duty:

$$\forall u \in \text{User}: |\text{roles}(u) \cap CR| \leq 1$$

”User u darf höchstens eine Konfliktrolle annehmen”.

- Darf ein **Kreditberater** auch **Kunde** in der Bank sein, in der er arbeitet?
 - Einführung von dynamischer SoD

2.16.7 Dynamische Separation of Duty

- Trennung von Rollen nicht beim UA, sondern beim Aktivieren von Rollen in Sitzungen (flexibler als statische SoD)
- Im RBAC-Formalismus:

active_roles(u)

$$= \{r \in R \mid (u, r) \in UA \wedge \forall s \in S. (\text{user}(s) = u \wedge r \in \text{session_roles}(s))\}$$

Dynamic Separation of Duty:

$$\forall u \in \text{User} : |\text{active_roles}(u) \cap CR| \leq 1$$

”User u darf höchstens eine Konfliktrolle **aktivieren**“.

2.16.8 Authorization Constraints

- Statische und dynamische SoD sind Ausprägungen des allgemeineren Konzepts der **Authorization Constraints**.
- Rollenbasierte Authorization Constraints sind Einschränkungen der Rollenrelationen und drücken **organisatorische Regeln** aus
- Andere Formen von rollenbasierten Authorization Constraints:
 - Verschiedenste (teilweise applikationsabhängige) SoD-Beschränkungen:
 - * Object-based Dynamic SoD: Ein Mitarbeiter aus der Finanzabteilung eines Unternehmens darf nicht gleichzeitig eine Bestellung vorbereiten **und dann** genehmigen
 - * Werden oft im Workflow-Kontext verwendet
 - Kardinalitätsbeschränkungen
 - Prerequisite Roles
 - Temporale Beschränkung

2.16.9 Einsatzgebiete von RBAC

- Betriebssysteme (RACF von IBM, Windows NT/2000/7/10 zumindest als Gruppenkonzept, SELinux - später)
- Middleware (z.B. Web Services)
- Rollenbasierte Administrationswerkzeuge wie z.B. DirXMetaRole von Siemens oder IBM Tivoli
- Datenbanken (z.B. Oracle, Sybase, Informix)

- Enterprise Resource Planning Systeme wie z.B. SAP
- Krankenhausinformationssysteme, Bankenanwendungen (weitere Verbreitung im Bankbereich), E-Goverment

2.16.10 Stärken

- Rollenkonzepte sind **flexibel verwendbar**
- **Vereinfachtes Berechtigungsmanagement** (wenn bspw. Benutzer häufig wechseln, dann braucht i.d.R. nur die Relation UA angepasst zu werden)
- Abbildung von **hierarchischen Organisationsstrukturen**
- Intuitive Abbildung auf Geschäftsprozesse
- Möglichkeit, **organisationsinterne Sicherheits- und Kontrollregeln** abzubilden (Authorization Constraints)
- Sowohl **Vertraulichkeit** als auch **Datenintegrität** (BLP kann durch RBAC simuliert werden)

2.16.11 Schwächen

- Woher bekommt man überhaupt die Rollen, die für eine Organisation relevant sind? (Problem des **Role Engineering**)
- Automatische Zuweisung von Berechtigungen zu Rollen wird nicht ausreichend unterstützt (manuelle Zuweisung in großen Organisationen nicht praktikabel)
- Rollenproliferation (Beispiele: Krankenhaus mit 300 Rollen, Verwaltung der Universität: 800 SAP-Rollen)
 - Faustregel: $|\text{Roles}| / |\text{User}| = 0.04$ (20,000 Mitarbeiter der Verwaltung?!)
- Viele Projekte zur Einführung von RBAC scheitern aus o.g. Gründen
- Kein Schutz gegenüber Covert channels (wie bei allen Modellen für die Zugriffskontrolle)

2.17 SELinux - Security Enhanced Linux

- Von der NSA (National Security Agency) entwickelt, 2001
- Erweiterung von Linux um MAC-Konzepte; frei verfügbar
- Unterstützung von SUSE-, Debian-, Ubuntu-Linux, auch von Android
- Mittlerweile Einsatz in britischen Behörden, um BLP-artige Zugriffskontrolle umzusetzen

2.17.1 Ziele

- Konfigurierbare Sicherheitsrichtlinien (**security policies**) für die Zugriffskontrolle, u.a.:
 - RBAC
 - BLP
 - Chinese-Wall
 - Biba
 - Clark-Wilson
 - Eigene, organisationsabhängige Security policies
- Trennung von Definition und Durchsetzung (**enforcement**) der Sicherheitsrichtlinien
- Trennung von Prozessen

2.17.2 Konzepte

Drei Zugriffskontrollkonzepte/-modelle:

- 1. DTE als Basis:
 - Der Einfachheit halber aber keine Domains, sondern nur Typen
- 2. RBAC:
 - Kein PA, sondern Zuordnung von DTE-Typen zu Rollen
 - UA wie immer
 - Auch Unterstützung von Authorization Constraints
- 3. IBAC (identity-based access control)
 - Jeder Prozess behält seine ID: im Gegensatz zum s-Bit in Linux

2.17.3 Sicherheitsarchitektur

- Sicherheitsrichtlinien werden in verschiedenen Textdateien definiert
- Integration eines **Security Servers** in den Linux-Kernel
- Laden der Sicherheitsrichtlinien beim Start in den Kernel
 - Dynamisches Nachladen von Sicherheitsrichtlinien möglich
- Überprüfung von Zugriffen durch den Security Server:
 - Werden beim Zugriff die definierten Sicherheitsrichtlinien verletzt?

- Security Server als API, der Rest des Kernels ruft diesen auf (enforcement)
- Subjekte und Objekte haben **Security Identifier** (SID), die bei der Überprüfung verwendet werden
- SID: Handler auf **Security Kontext**, der die eigentlichen Sicherheitsinformationen (Rolle, UserID, Typen...) enthält (Indirektion)

Wie sehen Sicherheitsrichtlinien in SELinux aus?

- DTE-Zugriffsmatrix
`allow type_1 type_2:class {perm_1 ... perm_n}`
`allow user_t tmp_t:dir {read search add_name remove_name}`
 "Jeder Nutzer kann im /tmp-Verzeichnis Dateien erzeugen und löschen"
- User Assignments
`user username roles {role_1, ..., role_n}`
- Type Assignments
`role rolename types {type_1, ..., type_n}`
- Zusätzlich: Datei für Authorization Constraints

2.17.4 Abschließende Anmerkungen

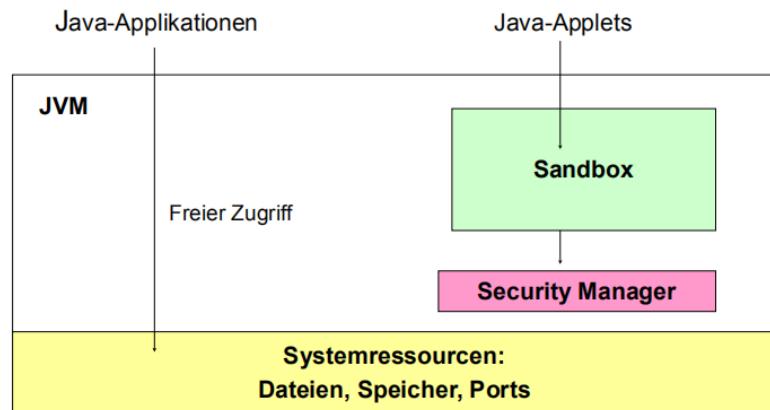
- Erhöhte Sicherheit durch: MAC-Unterstützung, Trennung von Prozessen, kein s-Bit
- Kompatibilität zu Linux-Distributionen/-Derivanten, z.B. SUSE, Debian, Android (Ubuntu setzt eher auf AppArmor)
- Flexible Konfigurationsmöglichkeiten von Sicherheitsrichtlinien (aber Dateien mit Sicherheitsregeln gut schützen!)
- Nachteil: Komplexität der Policy-Definition
 - Werkzeuge zur Policy-Analyse wünschenswert, Usability-Aspekt
 - Vorgegebene Policy-Definitionen
- Mehr über SELinux: <https://github.com/SELinuxProject>

2.18 Sandboxing

- **Einschränkung der Funktionalität** von Programmen/Prozessen (z.B. durch eine virtuelle Maschine)
- "Programm kann nur in einem Sandbox spielen"
- Prominentes Beispiel: Java-Applets

- Heute Renaissance im App-Konzept für Smartphones (App-Konzept aber deutlich umfassender)
- Beispielsweise kein schreibender und lesender Zugriff auf Dateien, keine beliebigen Netzverbindungen, kein Zugriff auf nativen Code
- Umsetzung dieser Sicherheitsrichtlinien durch den Java-Security Manager
 - * Java-Systemklasse
 - * Referenzmonitor
- Nicht einfach umzusetzen, wie die Java-Sicherheitsprobleme zeig(t)en

2.18.1 Java-Sandboxing im JDK 1.0 (veraltet)



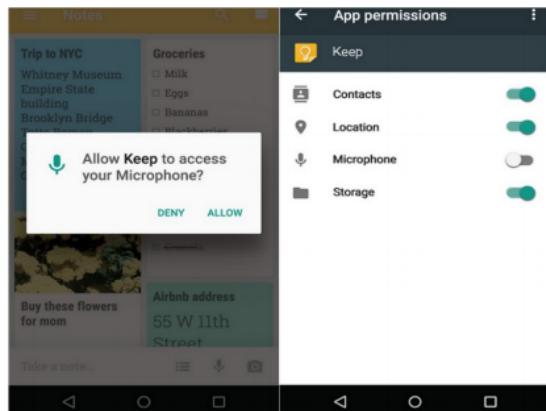
2.18.2 Sandboxing in Android

- Android ist eine der wichtigsten Smartphone-Plattformen (neben IOS), Marktanteil: >80%
- Linux- und Java-basierte Plattform, mit einigen Änderungen
- Apps können auf das Gerät geladen werden
 - Aus Google Play
 - Aus anderen Quellen
- Erweitertes Applet-Konzept, mit den bekannten Risiken
- **Android-Sandbox:** u.a. Apps werden jeweils unter eigener Unix-User ID installiert

Android Permissions

- Benutzer muss Berechtigungen bei der Installation freigeben (jedoch ab Android 6.0 zur Laufzeit)
- Typische vordefinierte Berechtigungen: INTERNET, LOCATION, RECEIVE_SMS
- Problem:
 - Benutzer muss Entscheidung fällen
 - Deshalb: Berechtigung grob granular (z.B. INTERNET)

Runtime-Bestätigung ab Android 6.0



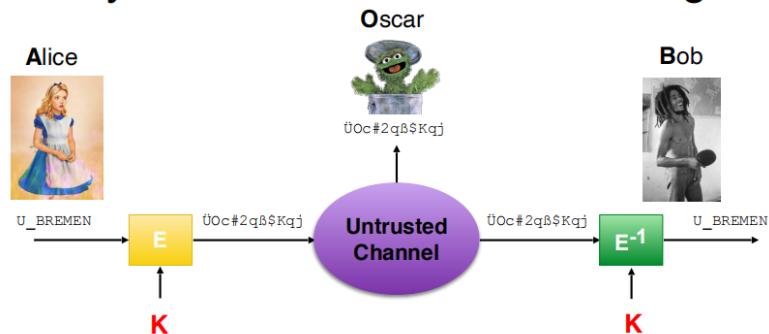
Chapter 3

Kryptographie

Kryptographie = Kryptographie + Kryptoanalyse

- **Kryptographie:** Methoden zur Ver- und Entschlüsselung von Nachrichten und damit zusammenhängende Methoden
- **Kryptoanalyse:** Entschlüsselung ohne Zugriff auf den Schlüssel
- Kryptographie benötigt die Kryptoanalyse, und die erreichte Sicherheit der Verfahren beurteilen zu können

3.1 Symmetrische Verschlüsselung



3.2 Terminologie

- Klartext (Plaintext): **P**
- Chiffertext/Geheimtext/Kryptotext (ciphertext): **C**
- Schlüssel (key): **K**

- Nachricht (message): **M**
 - Wenn unwesentlich ist, ob verschlüsselt wird oder nicht

3.3 Kryptoanalyse: Angriffsklassen

- Ziel: **Schlüssel** finden ohne jene Information, die beim Ent- (oder Ver-)schlüsseln von neuem Text hilft
- Chiffertext-Angriff (**ciphertext-only attack**):
 - Mehrere Chiffretexte stehen dem Angreifer zur Verfügung
 - Nutzt statische Eigenschaften zwischen Chiffretexten aus; ggf. auch die des zugrunde liegenden Klartextes (sofern bekannt)
- Angriff mit bekanntem Klartext (**known-plaintext attack**):
 - Sowohl Chiffertext/Klartextpaare stehen zur Verfügung
- Angriff mit gewähltem Klartext (**chosen-plaintext attack**):
 - Erzeuge eine Reihe von Klartexten
 - Erhalte die dazugehörigen Chiffretexte
- Angriff mit adaptiv gewähltem Klartext (oder Chiffertext) (**adaptive chosen-plaintext (or ciphertext) attack**):
 - Führe verschiedene Angriffe mit gewähltem Klartext (Chiffertext) durch
 - Nutze das Wissen der zuvor durchgeführten Angriffe zur Erzeugung von neuen Klartexten

3.4 Historische Kryptographie

* Heute zu leicht zu brechen

3.4.1 Transposition

- Transposition: Verschieben von Buchstaben innerhalb des Textes
- Beispiel: Gartenzaun-Verschlüsselung

D E E D N T I F M R E M T A E N
I S N U G R F T O G N I T G I

- Problem: Verfahren muss geheim gehalten werden
- Skytale: Lederstreifen wurde um Holzstück bestimmte Größe gewickelt und beschriftet



3.4.2 Caesar-Chiffre

- Caesar-Chiffre: Jeder Buchstabe wird durch dritt nächsten ersetzt (tim → wlp): $((x + 3) \% 26)$

P=A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
C=D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

- Beispiel: PRUJHQ IUXHK
- Variante der Caesar-Chiffre: ROT13 $((x + 3) \% 26)$

3.4.3 Kerckhoffs-Prinzip

- Kerckhoffs-Prinzip: Kryptosystem bleibt sicher, wenn alles (auch das Verfahren) außer einem **Schlüssel** bekannt - Auguste Kerckhoffs, 1883
 - Schlüssel kann leicht/oft getauscht werden
 - Offenes kryptographisches Verfahren erlaubt externe Analyse
- Erweiterte Caesar-Chiffre: Addend als Schlüssel
 - Nur 25 (26-1) mögliche Schlüssel \Rightarrow **brute force noch zu einfach**

3.4.4 Substitution

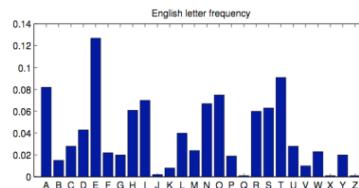
- Substitution: Ein Buchstabe wird durch einen anderen Buchstaben ausgetauscht
- Monoalphabetisch: Ein Buchstabe steht für genau einen anderen Buchstaben
- Problem wie bei der Caesar-Chiffre: Unwirksam, sobald Verfahren/Permutation bekannt ist

Historie: Substitutionsschiffre

Monoalphabetische Substitutionsschiffren

$$\begin{aligned} P = & \text{ A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z \\ C = & \text{ I | D | R | B | O | K | C | P | A | N | F | G | Q | X | H | L | V | Z | T | M | Y | E | W | J | S | U \end{aligned}$$

- Schlüssel ist Permutation $K : A \leftrightarrow A$
 - $21! \approx 3 \times 10^{26}$ verschiedene Schlüssel
 - Brute force ist bereits schwierig
- Statistik-Angriff:
 - Häufigste Buchstaben suchen (Detusch: E, N, I, S, R, A, ...)
 - Nach häufigen Diagrammen/Trigrammen suchen



Polyalphabetische Chiffren

P =	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
C1 =	Y	X	B	C	U	R	P	H	N	O	M	E	K	J	T	Z	W	I	G	D	A	F	Q	L	S	V
C2 =	Q	T	V	Y	R	X	A	J	W	N	U	C	I	F	L	M	B	E	H	Z	P	D	S	G	O	K

- "Vigenère"-Chiffre:

- $K = k_0 k_1 k_2 \dots k_{n-1}$
- $P = p_0 p_1 p_2 \dots p_{m-1}$
- $c_i = (p_i + k_{(i \% n)}) \% 26$ (via Codewort)
- Statt Modulo-Addition kann XOR oder jeder andere Gruppen-Operator verwendet werden
- Analyse: n finden; Angriff auf Stellen mit gleichem ($i \% n$)
- Verallgemeinerung: Änderung der Substitutionsregel bei jedem Zeichen

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

3.5 Perfekte Chiffren

- Angreifer erhält keine Informationen aus dem Chiffertext
 - Kenntnis eines Chiffretextes ändert nichts an der statischen Wahrscheinlichkeit möglicher Klartexte

- Einzige Lösung: One-time-pad
 - Schlüssel ist genauso lang wie Klartext und darf nur exakt einmal verwendet werden
 - Bietet informationstheoretische Sicherheit
 - $c_i = p_i \oplus k_i$ (XOR - jede andere Gruppe geht auch)
 - Problem: Wie Schlüssel transportieren?
 - * Wiederverwendung ist extrem gefährlich

3.6 Blockchiffren: DES und AES

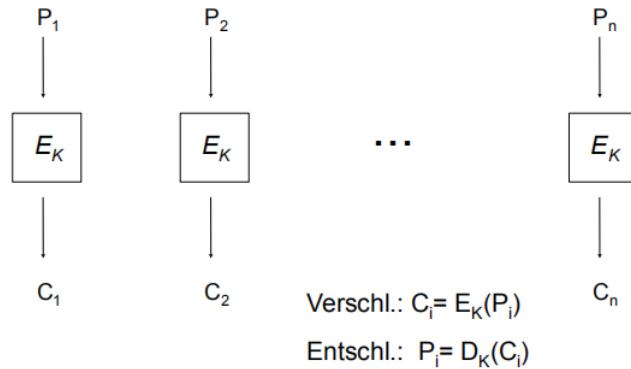
- Feste Blockgrößen und Schlüsselgrößen
 - DES: 64 bit pro Block, 56 bit pro Schlüssel
 - AES: 128 bit pro Block, 128, 196 oder 256 bit pro Schlüssel
- Produktchiffre (rundenbasierte Algorithmen)
 - DES: 16 Runden, AES (Rijndael): 10-14 Runden
- DES gilt als unsicher, 3DES = $E_{K1}(D_{K2}(E_{K3}(P)))$ (168 bit)
- AES wurde als Nachfolger von DES entwickelt
 - Offenes Auswahlverfahren
 - Viele Kryptoanalyse-Versuche vor und nach der Auswahl

3.6.1 Nutzung von Blockchiffren: Modes of Operation (Betriebsarten)

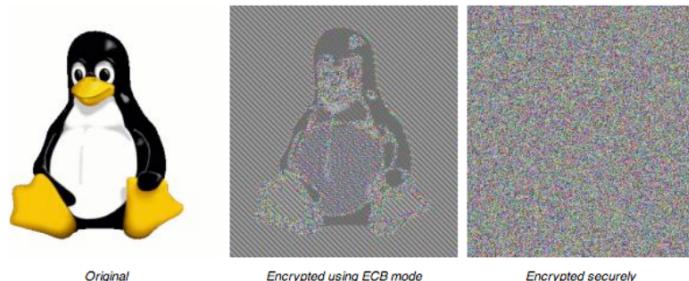
- ECB (Electronic Codebook): Je n Bit des Klartextes werden einzeln für sich verschlüsselt.
- Problem: Identitäten/Redundanzen im Klartext werden im Chiffertext sichtbar
 - Kryptanalyst kann sich mit known-plaintext "Bausteine" schaffen
 - Wiederholte Nachrichten sind leicht zu erkennen

3.7 Funktionsweise: ECB

- Je n Bits des Klartextes (=Block) werden einzeln für sich verschlüsselt



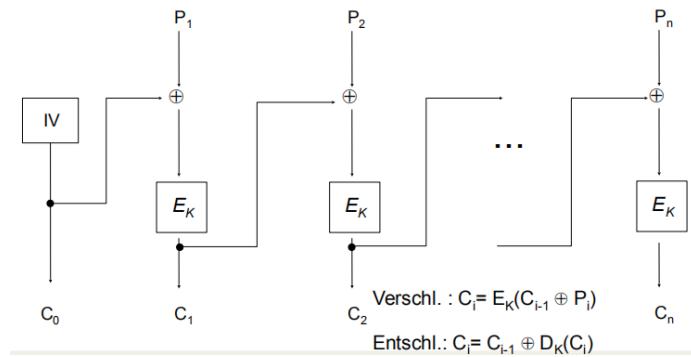
ECB: Visualisierung



3.8 CBC: Cipherblock Chaining

- Vor der Verschlüsselung wird P_i mit C_{i-1} verändert (XOR)
- Zum Schutz von P_1 wird zu Beginn eine Zufallszahl vorgegeben (Initialisation Vector, IV)
- Bitfehler zerstört aktuellen Block und führt zu Bitfehler im Folgeblock

Funktionsweise: CBC



3.9 Padding

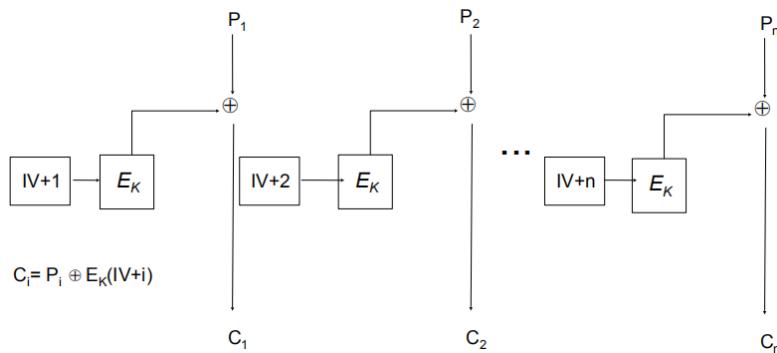
- Input ist nicht immer genau n Blöcke groß
- Auffüllen mit **Padding**
- Wie wieder entfernen?
- → Immer Padding,
 - z.B.: letztes Byte sagt die Anzahl der Bytes (SSLv3)
 - Besser: PKCS #7: Alle Padding-Bytes mit dem selben Wert

	BLOCK #1								BLOCK #2							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Ex 1	P	I	G													
Ex 1 (Padded)	P	I	G	0x05	0x05	0x05	0x05	0x05								
Ex 2	B	A	N	A	N	A										
Ex 2 (Padded)	B	A	N	A	N	A	0x02	0x02								
Ex 3	A	V	O	C	A	D	O									
Ex 3 (Padded)	A	V	O	C	A	D	O	0x01								
Ex 4	P	L	A	N	T	A	I	N								
Ex 4 (Padded)	P	L	A	N	T	A	I	N	0x08	0x08	0x08	0x08	0x08	0x08	0x08	0x08
Ex 5	P	A	S	S	I	O	N	F	R	U	I	T				
Ex 5 (Padded)	P	A	S	S	I	O	N	F	R	U	I	T	0x04	0x04	0x04	0x04

3.10 Weitere Stromchiffren-Modi

- **CTR** (Counter Mode):
 $R_i = E_K(i + O), C_i = P_i \oplus R_i$
- Offset O wird wie IV mit Chiffretext übertragen
- Vorteil: Direktzugriff in die Mitte des Stroms
- **CFB** (Cipher Feedback Mode):
 $R_i = E_K(C_{i-1}), C_i = P_i \oplus R_i$
 - Ein Bitfehler im Chiffretext erzeugt einen Bitfehler und zerstört den Folgeblock
- C_0 wird als IV mit Chiffretext übertragen

3.10.1 Funktionsweise: CTR

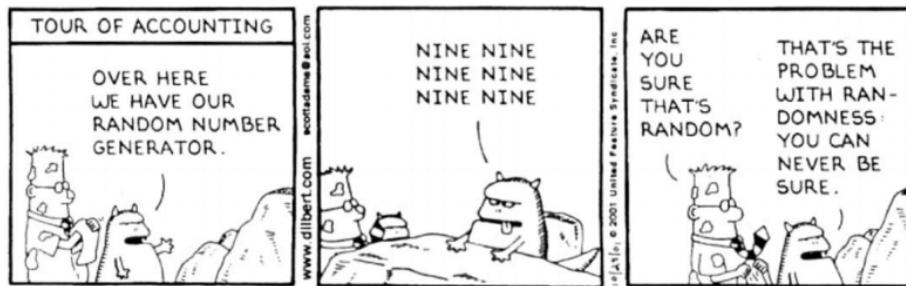


3.10.2 Angriffe gegen Stromchiffren

- Modifizierte Wiedereinspielung (vor allem OFB/CTR)
 - Ist Struktur bekannt, reicht eventuell ein "Bitfehler"
 - Schutz: Integritätsprüfung
- Bei Wiederverwendung der IV/K-Kombination:
 - $E(A) \oplus E(B) = (A \oplus R) \oplus (B \oplus R) = A \oplus B$
 - Schutz: K oft genug wechseln (und IV nie wiederholen)

3.11 Schlüsselerzeugung

3.11.1 Qualität von Zufallszahlen



3.11.2 Entropie

- Entropie in der Informationstheorie (Shannon 1948): Informationsgehalt "Maß der Unsicherheit" (Überraschung)
- N völlog zufällige Bits \rightarrow Entropie N bit
- Werden Bits vorhergesagt, sinkt die Entropie Redundanzen
Statische Regelmäßigkeit
- Verlustlose Kompression kann auf Entropie, aber nicht weiter komprimieren

3.11.3 Zufallszahlen

- Wichtig für Sitzungsschlüssel etc.
 - Viele Angriffe basieren auf der Möglichkeit, Zufallszahlen zu raten
- True RNG: Entropie wächst mit jedem Bit
- Pseudo-RNG, Deterministischer RNG: Entropie steckt im "Seed" (Startwert), dann konstant
 - Hybridformen: kontinuierliches Seeding
- Wie generieren?
 - Physikalische Prozesse und Entropiequellen
 - * Thermisches Rauschen, Jitter, Metastabile Prozesse, radioaktive Zerfallsprozesse
 - Befehlsweise zu verwenden
 - * Computerhardware (Festplatten, Sound- und Videokarten)

- * Benutzereingaben

- Nachteile von Hardware für Zufallszahlen:

- Oft langsam
- Nicht immer verfügbar
- Ohne Software-Hilfe sehr aufwendig

- Linux Entropiepool

- Benutzer: Tastatur, Maus
- Hardware: Netzlast, Interrupts

Linux: \dev\random

Aber: immernoch sehr langsam!

- \dev\urandom:

- Nimmt Daten aus Entropie-Pool
- Blockiert nicht auf leerem Pool
- Erzeugt dann selbst Zufallszahlen via DRNG (Deterministic Random Number Generator)

- Pseudozufallszahlengenerator (Pseudo Random Number Generator PRNG)

- Erzeugt aus einem Initialisierungswert (seed) immer gleiche Folge von Zufallszahlen

3.11.4 Block-Chiffre als Pseudo-Zufallszahlengenerator

- Kryptographisch starker PRNG: $R_i = E_K(R_{i-1})$
- **OFM** (Output Feedback Mode):
 $C_i = P_i \oplus R_i$ (mit $R_i = E_K(R_{i-1})$)
- **Achtung:** Wiederverwendung von K/R_0 = Bombe

3.12 Hash-Funktionen

3.12.1 Kryptographischer Hash

- Hash dampft variabel lange Binärdaten in Fingerabdruck ein:



► Beispiele: MD5, SHA-1, SHA-256:

- MD5: 128 bit
- SHA-1: 160 bit (obsolet)
- SHA-256: 256 bit



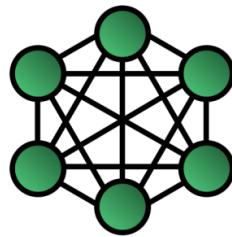
Was ist ein guter Kryptographischer Hash?

- Schwierig zu erreichendes Ziel:
 - Kollisionsresistent: schwer, $x \neq y$ zu finden mit $h(x) = h(y)$
 - * Zwei Personen finden, die am gleichen Tag Geburtstag haben
- Etwas weniger problematisch in der Sicherheitsbetrachtung:
 - Urbildresistent: schwer, zu einem a ein y zu finden mit $a = h(y)$
 - * Zu einem gegebenen Geburtstagsdatum die passende Person finden
 - Urbild-2: schwer, zu einem x ein $y \neq x$ zu finden mit $h(x) = h(y)$
 - * Eine zweite Person finden, die am gleichen Tag Geburtstag hat

3.12.2 Geburtstagsparadoxon

- 23 Personen sitzen in einem Raum
- Wie hoch ist die Wahrscheinlichkeit, dass eine dieser Personen am selben Tag wie ich Geburtstag hat?
- Wie hoch ist die Wahrscheinlichkeit, dass zwei von diesen Personen am selben Tag Geburtstag haben?
- Mit 6.1% Wahrscheinlichkeit hat eine Person von 23 am selben Tag Geburtstag
- Von 23 Personen haben mit 50% Wahrscheinlichkeit zwei am selben Tag Geburtstag

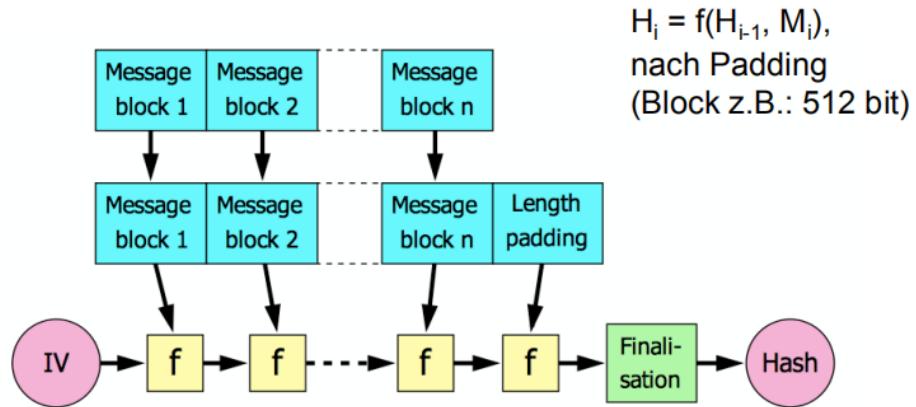
- $n \times (n - 1)/2$ Verbindungen
- Allgemein: aus k möglichen Werten reichen $k^{\frac{1}{2}}$ für ca. 50% Kollisionswahrscheinlichkeit



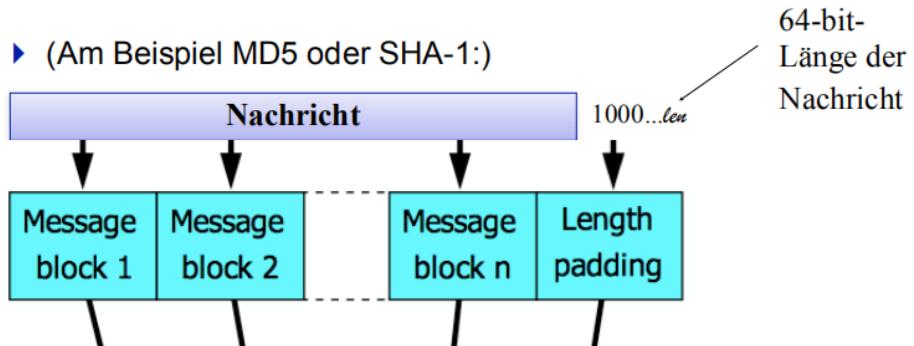
Einfluss auf das Geburtstagsparadoxons

- Erinnerung: Von 23 Personen haben mit 50% Wahrscheinlichkeit zwei am selben Tag Geburtstag
 - Allgemein: aus k möglichen Werten reichen $k^{\frac{1}{2}}$ für ca. 50% Kollisionswahrscheinlichkeit
- SHA-1: 2^{160} Möglichkeiten
 - 2^{80} Versuche ergeben mit $p \geq 0.5$ Kollision ("Geburtstagsangriff")
- Neue Angriffe auf SHA-1 reduzieren 2^{80} auf 2^{63}
 - Immernoch ähnlich 2^{64} (MD5 vor den Angriffen)
 - Übergang auf SHA-2 (SHA-256, SHA-512, ...)

3.12.3 Wir bauen uns einen Hash: Merkle-Damgård



3.12.4 Padding: Auffüllen auf Blockgröße



Merkle-Damgård

Hash-Funktion	Status, Haltbarkeitsdauer
MD5	Nicht mehr Kollisionsresistent, nur noch aus Rückwärtskompatibilität benutzen
RIPEMD-160	Europäische Spezialität, (~ SHA-1)
SHA-1	„Shattered“, war nur noch bis 2010 „sicher“
SHA-2 (SHA-256, SHA-512)	Z.Z. Funktionen der Wahl
„SHA-3“	Auswahlprozess (vgl. AES) → Keccak

Sponge

SHA-3 Competition

11/2/2007	SHA-3 Competition Began.
10/2/2012	<i>Keccak announced as the SHA-3 winner.</i>

IETF 86 NIST

Secure Hash Algorithms Outlook

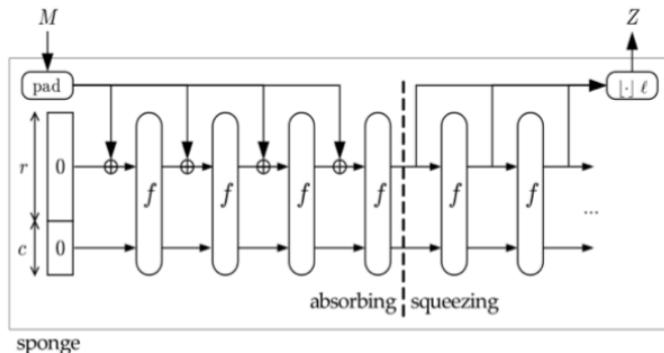
- ▶ SHA-2 looks strong.
- ▶ We expect Keccak (SHA-3) to co-exist with SHA-2.
- ▶ Keccak *complements* SHA-2 in many ways. Keccak is good in different environments.

Keccak is a sponge - a different design concept from SHA-2.

IETF 86

NIST

Sponge Construction

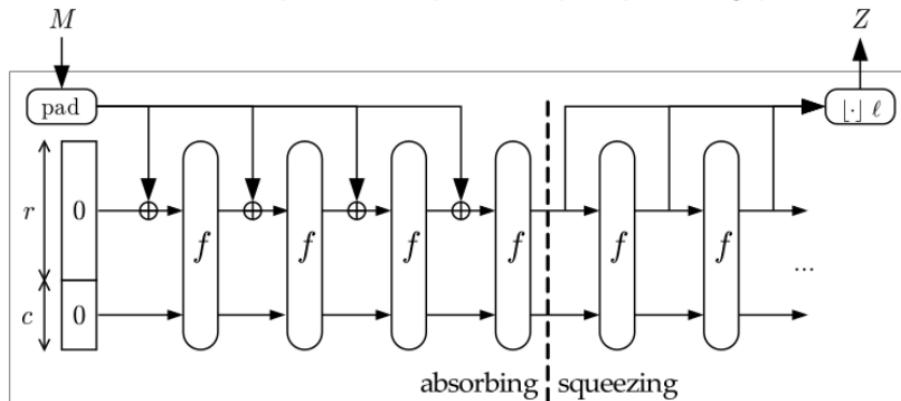


Sponge capacity corresponds to a security level: $s = c/2$.

IETF 86

NIST

Sponge mode: $b = r$ (bitrate) + c (capacity)

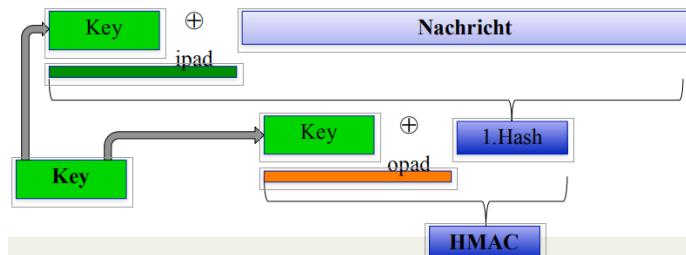


3.13 Message Authentication Codes (MAC)

- Nicht jede Nachricht möchte man signieren (aufwendig)
- Idee:
 - Erst gemeinsames Geheimnis vereinbaren
 - Sender verbindet dieses dann mit Nachricht kryptographisch
 - Empfänger wendet die gleiche Funktion an
 - Nachricht muss echt sein, wenn Ergebnis übereinstimmt
- "Hash mit Schlüssel" / Blockchiffren zur Nachrichtenauthentisierung

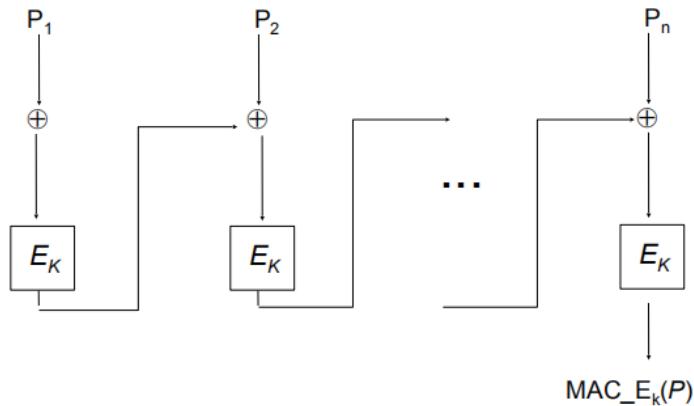
3.13.1 Einsatz von Hash-Funktionen für MACs: HMAC

$$\triangleright \text{HMAC}_K(M) = h(K \oplus \text{opad} \parallel h(K \oplus \text{ipad} \parallel M)) \quad (\text{RFC 2104})$$



3.13.2 Exkurs: CBC-MAC (Message Authentication Code)

- Integrität: Hash über Nachricht
- Kann nur mit Schlüssel K erstellt oder verifiziert werden



3.13.3 Kombination MAC/Verschlüsselung

- Stromchiffre braucht immer Integritätsprüfung
- Kombination möglich?
- OCB: Offset Codebook
 - Literatur:
 - * Rogaway, Bellare, Black, Krovetz: OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption
 - Problem: Patentiert
 - 802.11i verwendet stattdessen CTR + CBC-MAC
 - Details: RFC 3610

3.13.4 Authenticated Encryption with Associated Data (AEAD)

Authenticated Encryption

- Vertraulichkeit + Integrität + Authentizität
- effizient mit Block-Verschlüsselung realisierbar
→ benötigt Counter und CBC-MAC

Mechanismus: RFC 5116

- Verschlüsselung: $K \times N \times P \times A \rightarrow A \times C$
- Entschlüsselung: $K \times N \times A \times C \rightarrow A \times P$
- AHEAD-Algorithmus legt Länge von K,K.LEN, fest
K - Key, A Associated Data, P - Plaintext, N - Nonce, C - Cyphertext

3.14 Wie sicher ist mein Kryptosystem?

- Aussage: man müsste zum Brechen den gleichen Aufwand treiben wie für einen Brute-Force-Angriff mit 2^n Schritten
- n ist die Sicherheitsstufe in "Bits"
- z.B. AES-128: ungebrochen, also bräuchte man 2^{128} Schritte für einen Brute-Force-Angriff
 - "128-Bit-Sicherheit"
- Bei anderen Verfahren ist der Zusammenhang nicht immer so einfach!

3.14.1 Levels of Security

Table 7.1: Minimum symmetric key-size in bits for various attackers.

Attacker	Budget	Hardware	Min security
"Hacker"	0	PC	53
	< \$400	PC(s)/FPGA	58
	0	"Malware"	62
Small organization	\$10k	PC(s)/FPGA	64
Medium organization	\$300k	FPGA/ASIC	68
Large organization	\$10M	FPGA/ASIC	78
Intelligence agency	\$300M	ASIC	84

Level (Bits)	Encryption	Hash	RSA, D-H	ECC
80	(3DES)	(SHA-1)	1024	160
112	(3DES)	SHA-224*), SHA-256*)	2048	224
128	AES-128	SHA-256*)	3072	256
192	AES-192	SHA-384*)	7680	384
256	AES-256	SHA-512	15360	521

3.15 Asymmetrische Kryptographie

Das Problem mit der symmetrischen Verschlüsselung

- Die beiden Kommunikationspartner Alice und Bob brauchen ein **gemeinsames Geheimnis** K_{AB}



3.15.1 Schlüsselverteilungsproblem

- Wie Schlüssel K_{AB} etablieren, wenn man sich noch nicht kennt?
 - Symmetrische Verschlüsselung setzt einen vertraulichen Informationsskanal zur Übertragung des Schlüssels K_{AB} voraus!
(Problem des Schlüsselaustauschs bzw. Schlüsselsvereinbarung)
- Wie viele Schlüssel braucht man für 2, 3, 4, 10, 1000 Teilnehmer, wenn jeder jedem Nachrichten schicken können soll, die kein anderer lesen kann?
 - Für 2, 3, 4, 10, 10, 1000 Teilnehmer werden 1, 3, 6, 45, 499500 Schlüssel benötigt.
- Allgemein: Bei n potentiellen Kommunikationsteilnehmern benötigt man bis zu $n^x(n - 1)/2$ Schlüssel.
- $n = 10000$ entspricht ungefähr 50000000 Schlüsseln!

3.16 Asymmetrische (Public Key-) Verschlüsselung

- Whitfield Diffie, Martin E. Hellman (1976): Warum müssen eigentlich Sender und Empfänger denselben Schlüssel benutzen?
- Ein Schlüsselpaar (K_E, K_D) pro Kommunikationspartner
- Ein **öffentlicher** und ein **privater** Schlüssel
- Ein Schlüssel zum Verschlüsseln (K_E), der andere zum Entschlüsseln (K_D)



3.16.1 Allgemeine Eigenschaften asymmetrischer Verfahren

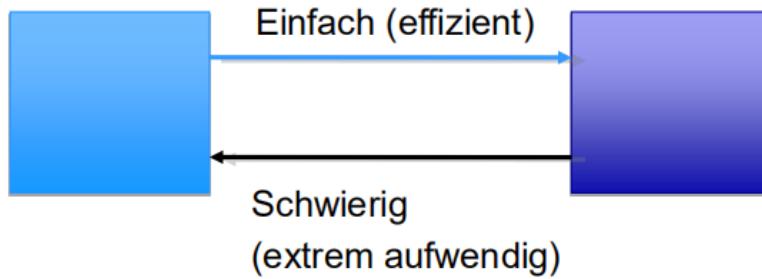
- K_E sei der **öffentliche**, K_D der **private** Schlüssel
- Die Schlüsselpaare (K_E, K_D) müssen folgende Eigenschaft erfüllen:
Für alle Nachrichten M : $D(E(M, K_E), K_D) = M$
(Eine Nachricht, die mit K_E verschlüsselt wird, wird mit K_D entschlüsselt und umgekehrt)
- Solche Schlüsselpaare müssen **leicht zu erzeugen** sein.
- Ver- und Entschlüsselung (E und D) sind **effizient durchführbar**
- K_D ist aus K_E **nicht mit vertretbarem Aufwand berechenbar**

3.16.2 Beispiele für asymmetrische Verfahren

- RSA ("Quasi-Standard")
- ElGamal-Verfahren
- Schlüsselvereinbarung nach Diffie und Hellman

3.17 Einweg-Funktionen

- Basis für asymmetrische Kryptographie sind Einweg-Funktionen (**one-way**)
 $f : X \rightarrow Y$
Eigenschaften von Einweg-Funktionen:
 - 1) $\forall x \in X$ gilt: $f(x)$ ist **effizient** berechenbar; und
 - 2) für fast alle $y \in Y$ gilt, dass es **nicht effizient** möglich ist, das Urbild zu y zu berechnen, also $x \in X$, mit $y = f(x)$
- Bis heute nicht bewiesen, ob überhaupt Einwegfunktionen existieren.
- Bewiesen ist: Sie existieren gdw. $P \neq NP$.



- ”Extrem aufwendig” = nicht in der Lebenszeit des Universums zu schaffen (“unmöglich”)

3.17.1 Einweg-Funktionen mit Falltür

- Einweg-Funktionen **mit Falltür** (**trapdoor one-way**):
 - mit Zusatzinformation sind Urbilder effizient berechenbar
- Gute Kandidaten für Einweg-Funktionen mit Falltür:
 - 1) **Potenzfunktion modulo n** mit $n=pq$, p und q Primzahlen $f : x \rightarrow x^e \bmod n$
 - 2) **Exponentialfunktion modulo p**, p Primzahl $f : x \rightarrow g^x \bmod p$
- Jetzt: Umkehrung der Exponentialfunktion

3.17.2 Primitivwurzel

- Vorausgesetzte Begrifflichkeit:
Gegeben p Primzahl. Eine **Primwurzel** $g \bmod p$ erzeugt alle von 0 verschiedenen Reste $\bmod p$ (Generator).

Beispiel: $p=7$. Dann ist $g=5$ eine Primitivwurzel $\bmod 7$. Denn:

$$\begin{array}{lll}
 5^1 & = 5 \bmod 7 \\
 5^2 = & 25 & = 4 \bmod 7 \\
 5^3 = & 4 \times 5 & = 6 \bmod 7 \\
 5^4 = & 6 \times 5 & = 2 \bmod 7 \\
 5^5 = & 2 \times 5 & = 3 \bmod 7 \\
 5^6 = & 3 \times 5 & = 1 \bmod 7
 \end{array}$$

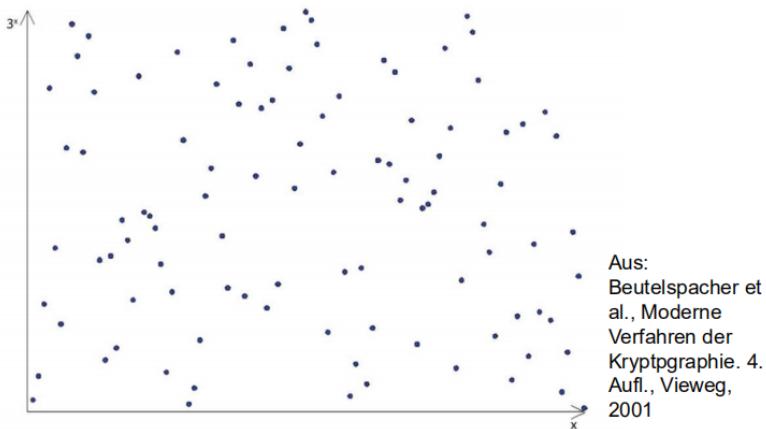
3.17.3 Problem des diskreten Logarithmus /1

- Gegeben p Primzahl, $g \leq p - 1$ primitive Wurzel und ganze Zahl y .
Bestimme die eindeutige Zahl k ($q \leq k \leq p - 1$) so, dass $y = g^k \pmod{p}$
- Bezeichnung für k : $\text{dlog}_g y$, z.B.: $\text{dlog}_5 4 = 2$
- **Ganzzahliges** Problem ("diskret")
- Diskreter Logarithmus als **Umkehrung** der diskreten Exponentialfunktion

3.17.4 Problem des diskreten Logarithmus /2

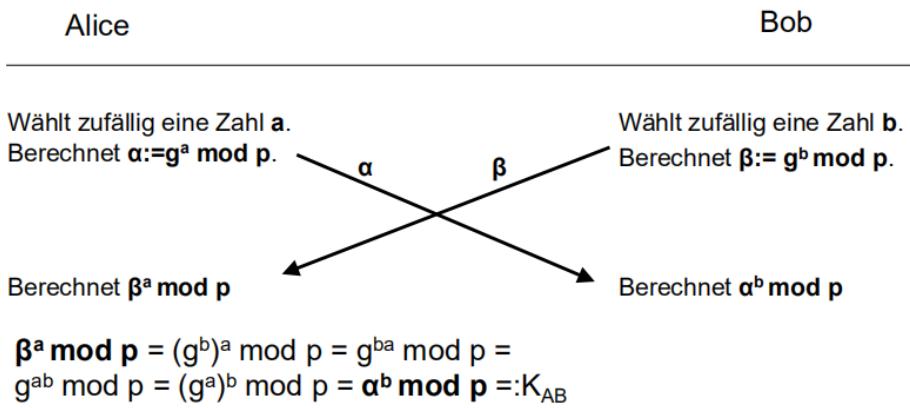
- Berechnung der Exponentialfunktion (auch für große p) effizient durchführbar
- Bestimmung des diskreten Logarithmus aber nicht
 - Nur Inspektion (alles durchprobieren)
 - Problem aus NP
- $f : x \rightarrow g^x \pmod{p}$ ist eine Einweg-Funktion mit Falltür (x)

Graph der diskreten Exponentialfunktion
 $3^x \pmod{101}$



3.18 Schlüsselvereinbarung nach Diffie und Hellman

- **Ziel:** Vereinbarung eines gemeinsamen geheimen Schlüssels, **ohne diesen über die Leitung zu schicken** (z.B. für eine symmetrische Verschlüsselung)
- **Achtung:** Diffie-Hellman allein liefert keine Verschlüsselung, keine Authentisierung der Partner!
 - Einsatz u.a. in TLS, Kerberos, IPsec-Protokollen, PGP, S/MIME
 - Basiert auf dem Problem des diskreten Logarithmus
- Funktionsweise:
 - Wähle große **Primzahl p** (allen Teilnehmern bekannt)
 - Wähle einen allen Teilnehmern bekannten **Wert g**, der primitive Wurzel von p ist
Es gilt also: $\{g^1, \dots, g^{p-2}, g^{p-1}\} = \{1, 2, \dots, p-2, p-1\}$.
 - g, p öffentlich bekannt



3.19 ElGamal-Verschlüsselungsverfahren

- Taher ElGamal, 1985
- Public-Key-Verfahren, basiert auch auf dem diskreten Logarithmus
- Variante von Diffie-Hellman, ähnliche Idee
- Wir setzen also wieder voraus: p Primzahl, g Primitivwurzel

3.19.1 Funktionsweise: ElGamal-Verschlüsselung

- **Schlüsselerzeugung** von Bob:
 - Wählt zufällig eine Zahl b und berechnet $\beta := g^b \text{ mod } p$
 - Veröffentlicht (β, g, p) als **öffentlichen Schlüssel**
 - b ist der **private Schlüssel**
- **Verschlüsselung** einer Nachricht M (mit $M \neq p$):
 - Alice schlägt (β, g, p) im Schlüsselverzeichnis nach.
 - Alice wählt eine *Nonce* (einmalige Zufallszahl) a und berechnet:

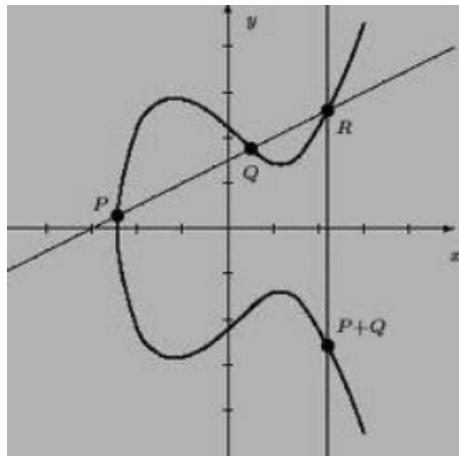
$$\begin{aligned} \alpha &:= g^a \text{ mod } p \\ k &:= \beta^a \text{ mod } p \\ c &:= kM \text{ mod } p \quad (\text{Maskierung der Nachricht } M \text{ mit } k) \end{aligned}$$
- Versenden der Nachricht: Alice verschickt die chiffrierte Nachricht (α, c) an Bob
 - **Der Chiffretext ist doppelt so lang wie der Klartext!**
- **Entschlüsselung:**
 - 1) Bob berechnet zunächst wieder k :

$$\begin{aligned} a^b \text{ mod } p &= (g^a)^b \text{ mod } p = g^{ab} \text{ mod } p = g^{ba} \text{ mod } p = (g^b)^a \text{ mod } p \\ &= \beta^a \text{ mod } p = k \end{aligned}$$
 - 2) Mittels k kann Bob dann wieder die Nachricht M berechnen:

$$k^{-1}c \text{ mod } p = k^{-1}kM \text{ mod } p = M \text{ mod } p = M$$

3.20 Asymmetrische Kryptographie auf Basis elliptischer Kurven

- Elliptic Curve Cryptography (ECC), Miller 1985, Koblitz 1987
- Verwendet elliptische Kurven (über Körper K) als algebraische Struktur
 - Zyklische Gruppe (Erzeugt aus primitivem Element)
 - Spezielle Addition auf dieser Gruppe
 - Beispiel: Elemente der Gruppe sind Punkte (x, y) mit der Eigenschaft $y^2 = x^3 + ax + b$



3.20.1 Praktische Bedeutung von ECC

- Problem des diskreten Logarithmus auch auf EC übertragbar
- Implementierung von Diffie-Hellman (ECDH) und ElGamal auf Basis von ECC
- Kleinere Schlüssellängen
 - 160 (224) Bit entspricht 1024 (2048) Bit bei herkömmlicher asymmetrischer Kryptographie
 - Weniger Speicherbedarf und Rechenleistung
 - Deshalb auch Einsatz auf Chipkarten
- Implementierungen in den meisten aktuellen Krypto-Bibliotheken vorhanden wie z.B. OpenSSL, Bouncycastle, Java

3.21 RSA-Algorithmus

- **RSA (1977/1978):** Ron Rivest, Adi Shamir, Len Adleman
 - 1983 patentiert, seit 2000 frei
 - Ähnliches System bereits 1973 von Clifford Cocks erfunden, aber geheim gehalten
- **Einsatzbereiche:** Verschlüsseln, Signieren, Schlüsselaustausch
- **Basis:** Primfaktorzerlegung

3.21.1 RSA: Funktionsweise

- Hier: nur skizziert; detaillierte Beschreibung in der Literatur
 - 1) Wähle Primzahlen p, q, **Modul n** = pq
 - 2) Wähle d so, dass $\text{ggT}((p-1)(q-1), d) = 1$,
 - 3) Berechne e mit $ed \equiv 1 \pmod{(p-1)(q-1)}$
(typischer Wert: $e = 65537 (= 2^{16} + 1)$)
- (e, n) öffentlicher Schlüssel; (d, n) geheimer Schlüssel
- **Verschlüsselung E:** $E(M) = M^e \pmod{n} = C$ (Einwegfunktion: Potenzfunktion)
- **Entschlüsselung D:** $D(C) = C^d \pmod{n} = M^{ed} \pmod{n} = M$
(Falltüren: z.B. Kenntnis von d oder Faktorisierung von p und q)
- Aktuell sichere GröÙe für n: 2048 Bits, z.T. sicherheitshalber **3072 Bits**

3.21.2 Basis der Sicherheit von RSA: Schwierigkeit der Faktorisierung

- Die Berechnung des Produkts aus zwei großen Primzahlen ist **effizient durchführbar** $n = pq$
- Die Berechnung von p und q aus n ist dagegen **sehr schwierig**
- Faktorisierungsproblem ist eines der ältesten Probleme der Zahlentheorie
- Es gibt einige Lösungsverfahren, aber der Berechnungsaufwand ist sehr groß
- Wenn man p und q kennt, dann hat man eine Falltür

3.21.3 Vorgriff: Einsatz von RSA für digitale Signaturen

Verschlüsselung:

- **Verschlüsselung E:** $E(M) = M^e \pmod{n} = C$
- **Entschlüsselung D:** $D(C) = C^d \pmod{n} = M^{ed} \pmod{n} = M$

Digitale Unterschrift:

- **Signatur D:** $D(M) = M^d \pmod{n} = S$
- **Verifikation E:** $E(S) = S^e \pmod{n} = M^{de} \pmod{n} = M$

3.22 Hybride Verschlüsselung

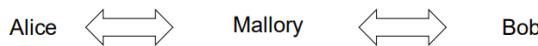
- Public-Key-Verfahren wesentlicher ineffizienter (Faktor 1000) als symmetrische Verfahren (z.B. XOR, Shift, Vertauschungsoperationen)
- In der Praxis deshalb häufig in **hybriden Verfahren** eingesetzt:
 - **Public-Key-Verfahren:** **Austausch des geheimen Schlüssels K** für die symmetrische Verschlüsselung
 - **Symmetrisches Verfahren zur effizienten Daten-Verschlüsselung**
- Typische Beispiele hierfür:
 - SSL/TLS
 - SSH
 - PGP
 - S/MIME

3.22.1 Ablauf: Hybride Verschlüsselung

- **Beispiel:** vertrauliche Kommunikation zwischen A und B: Schlüsselaustausch (mittels Public Key-Verfahren):
 - A: Erzeugt Sitzungsschlüssel K_{AB} für symmetrisches Kryptoverfahren und verschlüsselt diesen mit öffentlichem Schlüssel von B ($K_E B$)
 - B: Entschlüsselt mit dem privaten Schlüssel von B ($K_D B$)
 - Eigentliche Verschlüsselung zwischen den Partnern A und B mit symmetrischem Verfahren K_{AB}

3.23 Man-in-the-middle-Angriff

- **Problem:** keine ausreichende Authentisierung bei Public Key-Verfahren
 - Woher weiß Alice, dass der gefundene Public Key wirklich Bob gehört?
- Gefahr eines **Man-in-the-Middle-Angriffs** (besser: Middle-Person-Angriff):
 - Angreifer Mallory gibt sich gegenüber Alice als Bob aus.
 - Angreifer Mallory gibt sich gegenüber Bob als Alice aus.
 - ”Problem der Schachgroßmeister”
- Lösungsansätze später



3.24 Anwendung der Kryptographie: E-Mail-Sicherheit

3.24.1 E-Mail-Sicherheit als Anwendung der Kryptographie

- Einsatz kryptographischer Verfahren zur Absicherung von E-Mails zur Erreichung der
 - Vertraulichkeit,
 - Integrität, Authentizität
 - Verbindlichkeit
- Zwei verbreitete Ansätze
 - S/MIME (später)
 - Pretty Good Privacy (PGP)

3.24.2 Sicherheitsprobleme im Zusammenhang mit E-Mails

- E-Mails sind mit Postkarten vergleichbar.
- E-Mail-Inhalte können von Unbeteiligten gelesen werden.

Schutzziel: Sicherstellung der **Vertraulichkeit**

- **Sicherheitsproblem 2:**

- E-Mails können auf dem Kommunikationsweg verändert werden.

Schutzziel: Sicherstellung der **Integrität** der Nachricht

- **Sicherheitsproblem 3:**

- Der Absender einer E-Mail kann gefälscht werden (s. Postbank-Nachrichten)

Schutzziel: Authentizität

- **Sicherheitsproblem 4:**

- Wie kann man nachweisen (auch gegenüber einem neutralen Dritten), dass eine E-Mail wirklich versendet bzw. empfangen worden ist?

Schutzziel: Nicht-Abstreitbarkeit, Verbindlichkeit

3.25 PGP - Pretty Good Privacy

- Entwickelt von Phil Zimmermann, 1991
- Bietet Authentisierung, Verschlüsselung, digitale Signaturen für E-Mails:
 - Schutzziele: Vertraulichkeit, Integrität und Authentizität
- Auch Dateiverschlüsselung möglich
- OpenPGP-Standard, RFC 4880

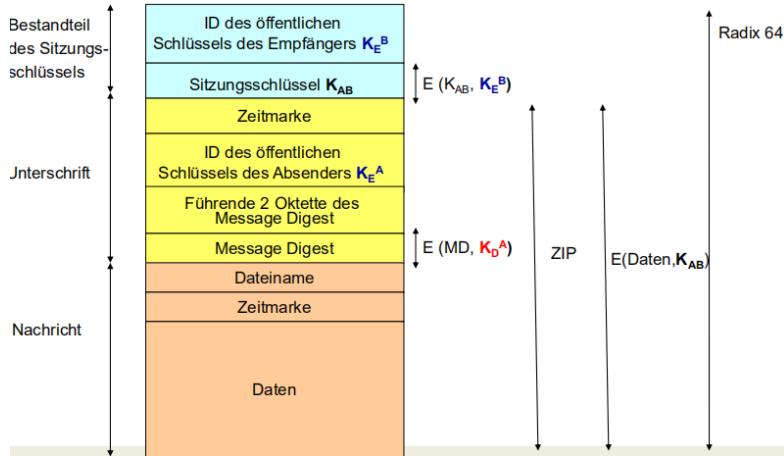
3.25.1 PGP - Implementierungen

- Freie Software und kommerzielle Produkte, z.B.:
 - Lizensierte Fassung (50\$), jetzt von Symantec übernommen: <https://www.pgp.com>
 - Gpg4win
 - Enigmail
- Plug-Ins für gängige E-Mail-Clients wie z.B. Outlook, Thunderbird
- PGP ist vor allem für Privatnutzer geeignet

3.25.2 Sicherheitsdienste von PGP

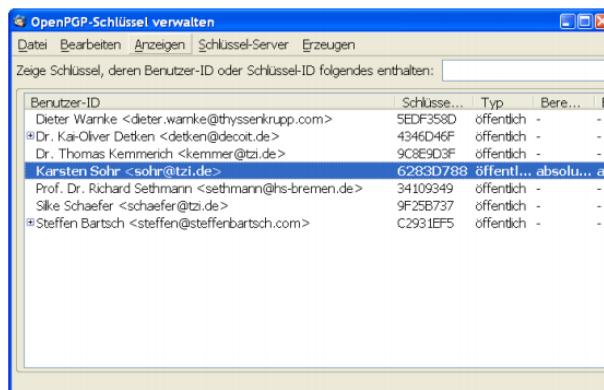
- Einsatz von gängigen Verfahren für Verschlüsselung, digitale Signatur und Authentisierung
- **Hybride Verschlüsselung**
 - Public Key-Verfahren zum Schutz des Schlüssels für die symmetrische Verschlüsselung
 - Eigentliche Verschlüsselung symmetrisch
- Sicherung des privaten Schlüssels durch Passphrase (→ Trojanisches Pferd)
- Kryptographische Verfahren:
 - RSA für symmetrische Verschlüsselung
 - IDEA, 3DES, CAST, nun auch AES für symmetrische Verschlüsselung
 - Hashverfahren: z.B. SHA 256
 - U.a. RSA für digitale Signatur, DSA

Format einer PGP-Nachricht (von A nach B)



3.25.3 Schlüsselverteilung in PGP

- Woher bekomme ich den öffentlichen Schlüssel des Kommunikationspartners?
 - Import von einem **Schlüsselserver** wie z.B.: <https://wwwkeys.de.pgp.net>
 - Von einem anderen Kommunikationspartner per E-Mail
- Erzeugung eines Schlüsselbundes mit den öffentlichen Schlüsseln der Kommunikationspartner
- **Problem:** Woher weiß ich, dass es sich wirklich um den öffentlichen Schlüssel des Kommunikationspartner handelt?
 - Auf den Schlüsselservern gibt es öffentliche Schlüssel wie z.B. merkel@bundesregierung.de.



3.25.4 Web of Trust

- Prinzip zur **sicheren Verteilung** der öffentlichen Schlüssel in PGP:
 - Kommunikationsteilnehmer **zertifizieren** (signieren) sich gegenseitig öffentliche Schlüssel (**Web of Trust, Vertrauensgeflecht**)
 - Einführung eines öffentlichen Schlüssels durch einen Dritten (z.B. gemeinsamer Bekannter, c't-Kampagne, ...)
 - Wenn man diesem Dritten vertraut, dann vertraut man auch öffentlichen Schlüsseln, die der Dritte unterzeichnet hat.
 - Signieren eines Schlüssels durch mehrere Kommunikationspartner möglich
- PGP-Implementierungen (wie GnuPG) stellen Software zum **Signieren von Schlüsseln** zur Verfügung.

Beispiel:

- Alice will Bob eine verschlüsselte E-Mail senden.
- Alice hat Bobs öffentlichen Schlüssel nicht im Schlüsselbund.
- Alice kann den Schlüssel von einem Schlüsselserver importieren.
- Charlie hat Bobs öffentlichen Schlüssel signiert, und Alice vertraut Charlie vollständig.
- Also vertraut Alice Bobs öffentlichem Schlüssel.

3.25.5 Vertrauensstufen für öffentliche Schlüssel in PGP

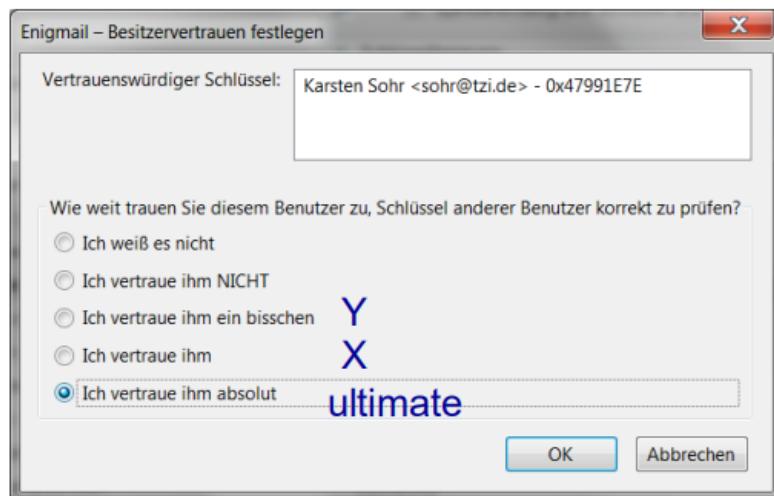
- Vertrauensstufen für öffentliche Schlüssel
 - 1) Undefiniertes Schlüsselvertrauen
 - 2) Schlüsseleigentum nicht gesichert
 - 3) Teilweises Vertrauen in den Schlüsselbesitz
 - 4) Volles Vertrauen in den Schlüsselbesitz

3.25.6 Berechtigung von Vertrauen in einen öffentlichen Schlüssel

- PGP berechnet das Vertrauen in den öffentlichen Schlüssel mittels einer **gewichteten Summe** aus dem Vertrauen in die Signaturen:
 - Forderung für volles Vertrauen in Schlüsselbesitz: gewichtete Summe ≥ 1
 - Absolut ("ultimate") vertrauenswürdige Signaturen haben ein Gewicht von 1

- Immer vertrauenswürdige Signaturen haben ein Gewicht von $1/X$, $1/Y$
- X, Y konfigurierbare Parameter: Im Falle von $X=2$, $Y=4$ würden z.B. eine immer vertrauenswürdige Signatur und zwei Signaturen mit gewöhnlichem Vertrauen ausreichen

Schlüsselvertrauen: Enigmail-Dialogfenster



3.25.7 Rücknahme von öffentlichen Schlüsseln

- Manchmal wird ein eigener privater Schlüssel gebrochen.
- Oder: Trojanisches Pferd liest den Schlüssel aus dem Speicher!
- Oder: Der Schlüssel war zu lange in Gebrauch.
- Rücknahme des Schlüsselpaars in PGP möglich.
- Vorgehensweise:
 - Veröffentlichen eines signierten **Rücknahmezertifikates**
 - Rücknahmezertifikat wird durch privaten Schlüssel des zurücknehmenden öffentlichen Schlüssels unterschrieben.
 - Letzte Aktion dieses Schlüssels

Chapter 4

Digitale Signatur, Zertifikate, PKI, E-Mail-Sicherheit

4.1 Digitale Signaturen

- Elektronisches Pendant zur eigenhändigen Unterschrift
- Electronic Signatures in Global and National Commerce Act ("E-SIGN", US, 2000), §106(5): "The term "electronic Signature" means an electronic sound, symbol, or process, attached to or logically associated with a contract or other record and executed or adopted by a person with the intent to sign the record."

4.1.1 Anforderungen an digitale Signaturen

- Digitale Signatur sollte nach Bruce Schneider folgende **Eigenschaften** besitzen:
 - 1. Überprüfbar (**verifiable**): Der Empfänger kann einfach überprüfen, dass die Nachricht von dem Unterzeichner unterschrieben wurde
 - 2. Nicht fälschbar (**unforgeable**): Nur der Unterzeichner kann die Signatur an das Dokument anhängen
 - 3. Nicht wiederverwendbar (**nonreusable**): Man kann nicht einfach eine Unterschrift von einem Dokument entfernen und an ein anderes anhängen
 - 4. Unveränderbar (**unalterable**): Nach der Unterzeichnung ist das Dokument nicht mehr veränderbar (vgl. **Datenintegrität**)
 - 5. Nicht zurücknehmbar (**nondeniable**): Die Unterschrift kann nicht zurückgenommen werden (vgl. **Verbindlichkeit**)

4.1.2 Digitale Signatur mit RSA

Digitale Signatur

- **Signatur D:** $D(M) = M^d \bmod n = S$
- **Verifikation E:** $E(S) = S^e \bmod n = M^{de} \bmod n = M$

In wie weit werden die fünf Anforderungen an eine digitale Signatur erfüllt?

Zum Beispiel:

- Verifikation (s.o.)
- Unveränderbarkeit: unterschiedliche Werte von M liefern verschiedene Werte für $M^d \bmod n$

4.1.3 Digitale Signatur mit ElGamal

- Keine Vertauschung von Ver- und Entschlüsselungsfunktion wie bei RSA
 - Nachricht M kann aus der Signatur nicht zurückgewonnen werden
- **Schlüsselerzeugung** von Alice (wie bei der Verschlüsselung):
 - Wählt zufällig eine Zahl a und berechnet $\alpha := g^a \bmod p$.
 - Veröffentlicht (α, g, p) als öffentlichen Schlüssel.
 - a ist der **private Schlüssel!**
- **Digitale Signatur** einer Nachricht M :
 - Alice wählt eine einmalige, geheime Zufallszahl (nonce) k und bildet:

$$r = a^k \bmod p$$
 - Dann berechnet Alice s , so dass
$$(a \times r + k \times s) = H(M) \bmod (p - 1)$$
 - (Erweiterten euklidischen Algorithmus anwenden und k^{-1} berechnen
 - Multiplikatives Inverses zu k
 - Digitale Signatur ist das Paar (r, s)

- **Verifikation** der digitalen Signatur einer Nachricht M :

- Empfänger (z.B. Bob) erhält M und Signatur (r, s)
Empfänger überprüft gegen öffentlichen Schlüssel α :

$$g^{H(M)} \bmod p = (\alpha^r \times r^s) \bmod p$$

Denn es müsste gelten:

$$g^{H(M)} \bmod p = g^{a \times r + k \times s} \bmod p = g^{a^r} \times g^{k^s} \bmod p = (\alpha^r \times r^s) \bmod p$$

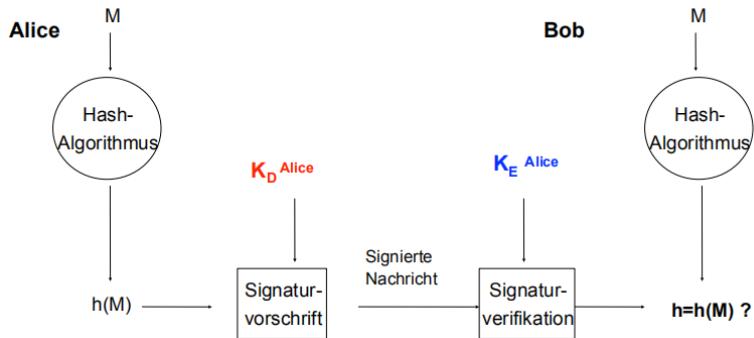
4.1.4 Prinzip der digitalen Signatur

- Problem bei obigem Signaturschemata mit RSA und ElGalam:
 - die Signatur ist zu lang
- Lösungsidee:
 - Alice bildet zunächst den **Hashwert** $H(M)$ (z.B. mit SHA-1; MD-5) von M und signiert diesen mit ihrem **privaten Schlüssel**.
 - Alice sendet die Nachricht und die Signatur an Bob.
 - Bob bildet ebenfalls den Hashwert $H(M)$ von M .
 - Bob entschlüsselt die "digitale Signatur" mit dem **öffentlichen Schlüssel** von Alice und erhält den Wert \mathbf{h} .
 - Gilt $\mathbf{h}=\mathbf{H}(M)$, so akzeptiert Bob die Signatur.

4.1.5 Wiederholung: Kryptographischer Hash

- Merkle-Damgård: Hash fester Größe aus variablen Strings
 - **MD5**: 128 Bit
 - **SHA-4**: 160 Bit $H_i = C(H_{i-1}, M_i)$, nach Padding (Block: 512 Bit)
 - RIPEMD-160 (Von der EU entwickelt)
- Heute: SHA-2 (Merkle-Damgård), SHA-3 (Keccak - sponge function)
- Ziele:
 - Kollisionsresistent: schwer, $x \neq y$ zu finden mit $h(x) = h(y)$
 - Urbildresistent: schwer, zu einem a ein y zu finden mit $a=h(y)$
 - (Urbild-2 schwer, zu einem x ein $x \neq y$ zu finden mit $h(x)=h(y)$)

4.1.6 Funktionsweise: Bildung und Verifikation der digitalen Signatur



4.2 Digital Signature Algorithm (DSA)

- United States Federal Goverment Standard für digitale Signaturen
- 1991 vom National Institute for Standards and Technology (NIST) als ein Algorithmus für den Einsatz im Digital Signature Standard (DSS) vorgeschlagen
- Basiert auf Algorithmus für Digitale Signaturen gemäß ElGamal
- Verwendung der Einweg-Hashfunktion SHA-1 (oder -2)

4.2.1 DSA: Status

- Erfordert guten RNG
- Abhilfe: RFC 6979 (Deterministic Usage)
 - "Zufallszahl" aus dem Hash bestimmten
- OpenSSH: DSA war eine Weile auf 1024 bit beschränkt
 - Das ist zu wenig gegen nationale Angriffe
- DSA ist jetzt in OpenSSH ganz deprecated
 - Man will eh ECDSA (oder ed25519) verwenden
- DSA ist aber weiterhin sicher (mit genug Bits, und mit gutem RNG oder RFC 6979)

4.3 Zertifikate

- Problem: woher weiß Bob, dass K_E^{Alice} zu Alice gehört?
 - Persönlicher Austausch des **öffentlichen Schlüssels** ist in der Regel nicht praktikabel
 - Gefahr eines Man-in-the-Middle-Angriffs
- Wie kann man einen **öffentlichen Schlüssel mit der Identität des Besitzers** (z.B. Alice) verknüpfen?
- Lösung: Ausstellung von Zertifikaten durch eine vertrauenswürdige Partei bzw. Zertifizierungsstelle (**Certification Authority**, CA)

4.3.1 Zertifikate vs. Schlüsselpaare

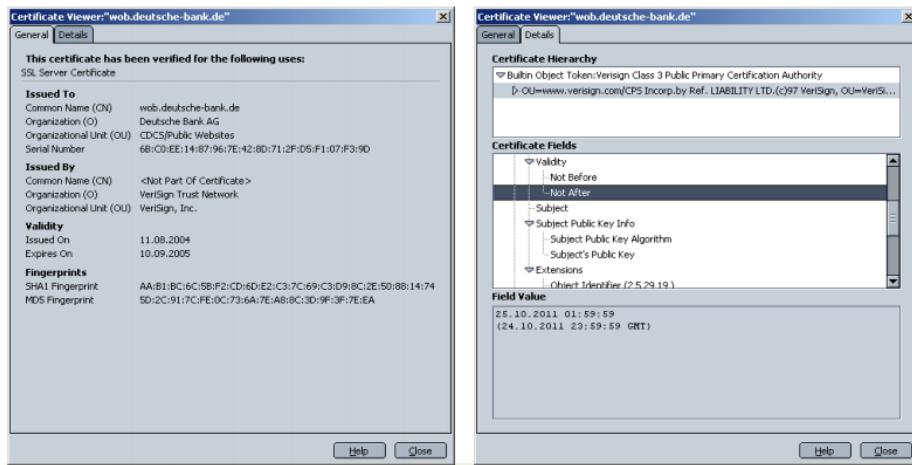
- Server authentisiert sich, indem er beweist, im Besitz des **privaten Schlüssels** zu sein
- Schlüsselpaar: Privater Schlüssel + Öffentlicher Schlüssel
 - wird im allgemeinen direkt auf dem **Server** generiert
- Zertifikat: Öffentlicher Schlüssel, Digitale Signatur der CA
 - wird von der **CA** aufgestellt
 - die braucht dafür nur den Öffentlichen Schlüssel + Metadaten: **Certificate Signing Request (CSR)**
- Zertifikate sind **öffentlich** und werden jedermann gegeben, damit diese den Server authentisieren können
 - für letzteres benutzt Server dann den geheimen **privaten** Schlüssel

4.3.2 X.509 v3-Zertifikate

- Festgelegt in RFC 5280 und RFC 6818 (war 3280, 2459): Standard für digitale Zertifikate
- **SubjectName:** C=DE, O=Uni-Bremen, OU=FB3, OU=People, CN=Emil Mustermann
- **IssuerName:** C=US, O=Verisign, CN=Public CA
- **SerialNumber:** 1B
- **Validity-NotBefore:** Wed Sep 18 09:12:25 2002 (020918071225Z)
- **NotAfter:** Mon Jan 1 00:00:00 2007 (061231230000Z)

- **SubjectKey:** Algorithm RSA (OID 1.2.840.113549.1.1.1), NULL
Public modulus (no. of bits = 1024):
0 CACAF080 64F76D97 EA2662FA 81FF10FF
Public exponent (no. of bits = 17):
0 010001

Beispiel: X.509 v3-Zertifikat und seine CA



X.509 v3-Zertifikate: Details

- Im Wesentlichen entspricht ein X.509-Zertifikat der folgenden kryptographischen Formulierung:

$$Cert_{CA}(A) = \{A, K_E^A, T, L\}_{K_D^{CA}}, \text{ wobei:}$$

A - SubjectName
T - Gültigkeitsbeginn des Zertifikates
L - Gültigkeitsdauer

- CA bestätigt also durch eine digitale Signatur, dass K_E^A zu A gehört und dass K_E^A aktuell ist.
- Woher aber kennt B nun den **öffentlichen Schlüssel** K_E^{CA} der CA?
- Bildung von Zertifikatsketten

4.3.3 Ketten von Zertifikaten

- **Kette von Zertifikaten** der Certification Authorities CA_1, \dots, CA_n
- CA_1 bestätigt öffentlichen Schlüssel von CA_{i+1}

- CA_n bestätigt öffentlichen Schlüssel des Benutzers
- Wer bestätigt das Wurzelzertifikat?
 - In einem Web-Browser geschieht dies durch Einbau der Wurzelzertifikate in den Browser.
 - **”Soziale” Argumentation:** Der Benutzer vertraut seiner Software (Browser) und somit auch den eingebauten Wurzelzertifikaten.

4.3.4 Public Key Infrastructure

- **Public Key Infrastructure (PKI):** ”Public Key Infrastructure (PKI), provides the means to bind public keys to their owners and helps in the distribution of reliable public keys in large heterogeneous networks.”
- NIST

The set of hardware, software, people, policies and procedures needed to **create, manage, store, distribute, and revoke Public Key Certificates** based on public-key cryptography. - IETF (PKIX WG)

”A system of CAs (and, optionally, RAs and other supporting servers and agents) that perform some set of certificate management, archive management, key management, and token management functions for a community of users in an application of asymmetric cryptography” - IETF (RFC 2828, 4949)

- **Komponenten einer PKI**

- **Certification Authority (CA):**
 - * Stellt Zertifikate aus und signiert sie
 - * Veröffentlicht aktuelle Zertifikate
 - * Erstellt und veröffentlicht Listen von ungültigen Zertifikaten (**Certificate Revocation List**, CRLs)
 - * bietet Online Certificate Status Protocol (**OCSP**) für Clients
- **Registration Authority (RA):**
 - * Arbeitet CA zu, bürgt für die Verbindung zwischen öffentlichem Schlüssel und Identitäten/Attributen der Zertifikatsinhaber
- **Verzeichnisdienst:** i.d.R. LDAP, Verteilung der Zertifikate und CRLs
- **Zeitstempeldienst:** signierte Zeitstempel (Gültigkeitsdauern, ...)

4.3.5 Fazit zu Signaturen, PKI

- Wissenschaftliche Fragestellungen sind **im Prinzip lange gelöst**
- Dem PKI-Hype Anfang 2000er folgte die Ernüchtigung:
 - Das Etablieren einer Unternehmens-PKI ist **aufwendig**
 - Das **Ausrollen** von Zertifikaten und Signaturkarten ist aufwendig, **Nutzen war gering** (Anwendungen?)
 - **Sperrlistenverwaltung**, Trust-Center-Aufbau etc. schwierig
 - Unternehmensübergreifende PKI-Strukturen **gibt es kaum** (Haltung?)
- **Einige Standards zur Thematik:**
 - **X.509:** Standard für digitale Zertifikate
 - **PKCS:** De-Facto Standards, definiert von RSA (vgl. RFC 3447 = PKCS#1)
 - **PKIX:** PKI-Gesamtstandard für das Internet
 - **ISIS-MTT:** deutscher PKI-Gesamtstandard, angelehnt an PKIX

4.4 S/MIME - Secure Multipurpose Internet Mail Extension

- Erweitert den MIME-Standard um Sicherheitsdienste
- Ursprünglich von der Firma RSA entwickelt
- Nähere Informationen zur aktuellen Version (3.1):
 - <http://www.ietf.org/html.charters/smime-charter.html>
 - Relevante RFCs: RFC 8550 (5750, 3850) (Certificate Handling), RFC 8551 (5751, 3851) (Message Specification), RFC 5652 (3852) (CMS = Cryptographic Message Syntax)
- Automatische Unterstützung in verschiedenen E-Mail-Programmen wie z.B.:
 - Outlook, Outlook Express, Lotus Notes, Netscape

4.4.1 Sicherheitsdienste von S/MIME

- Ähnliche Sicherheitsdienste wie bei PGP:
 - Verschlüsselung von Nachrichteninhalten (envelop):
 - * Hybrides Verfahren

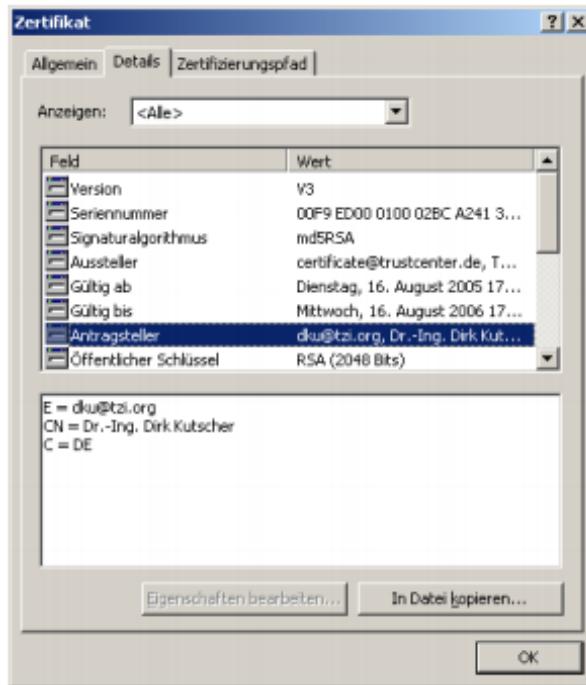
4.4. S/MIME - SECURE MULTIPURPOSE INTERNET MAIL EXTENSION81

- Signieren der Daten (inkl. E-Mail-Anhängen)
 - Signieren der Daten (inkl. E-Mail-Anhängen), aber Signatur von Nachricht getrennt (clear signing)
 - * Nicht S/MIME-fähige Empfänger können die Mail auch verarbeiten
 - Verschlüsselung und Signatur
 - * Erst Signatur, dann Nachricht Verschlüsseln
 - Signierte Bestätigungen (signed receipts)
 - * Stellen **Verbindlichkeit** sicher
- Verwendete kryptographische Verfahren (~~obsolet~~): RSA, ~~3-DES~~, AES, ~~40-Bit-RC2, SHA-1, MD-5~~, DSA, ECDSA, EdDSA, Diffie-Hellman, ECDH

4.4.2 Authentisierung in S/MIME

- **Bekanntes Problem:** Woher weiß ich, dass es sich wirklich um den öffentlichen Schlüssel des Kommunikationspartners handelt?
- Anstelle Web of Trust: X.509v3-Zertifikate
- X.509v3-Zertifikate werden mit der verschlüsselten bzw. signierten Nachricht mitgeschickt.
- Das S/MIME-fähige E-Mail-Programm übernimmt dies für den Absender.
- Ausstellung von X.509v3-Zertifikaten durch eine **Certification Authority (CA)**, z.B.:
 - VeriSign (DigitalID), Telekom, TC-Trustcenter in Hamburg, organisationsinterne CA

Beispiel für ein S/MIME-Zertifikat



- X.509 v3 Zertifikat
- Bindung des öffentlichen Schlüssels an den Namen und die E-Mail-Adresse des Antragsstellers
- Aussteller des Zertifikates ist TC Trustcenter

4.4. S/MIME - SECURE MULTIPURPOSE INTERNET MAIL EXTENSION83

4.4.3 Vergleich PGP - S/MIME

Merkmal	PGP	S/MIME
Verschlüsselung	+ (hybrid)	+ (hybrid)
Digitale Signaturen	+	+
Verschlüsselung und Signatur von Anhängen möglich	+ (sogar einzelne Dateien)	+
Stärke der kryptographischen Verfahren	Hoch (ursprünglich Einsatz von patentierten Verfahren wie IDEA)	Hoch, aber auch schwache Verfahren möglich wie RC2/40bit (Exportbeschränkung)
Authentisierung von öffentlichen Schlüsseln	<u>Web of Trust</u>	<u>X.509v3-Zertifikate</u>
Rücknahme von Schlüsselpaaren	+ (Benutzer selbst verantwortlich für Rücknahme)	+ (Benutzer selbst verantwortlich für Rücknahme; CRL)
Kosten	Freie Software, früher kommerzielle Fassung 50\$	Kosten für X.509-Zertifikate: z.B. ca. 20\$ pro Jahr, freemium Konstenlos in der Uni
Signed Receipts	-	+
Unterstützung in E-Mail-Programmen	Plug-Ins	Direkter Einbau in gängige E-Mail Programme wie Outlook Express
Einsatzgebiet	Vor allem Privatnutzer	Auch kommerzieller Einsatz

Chapter 5

Authentisierung, Sicherheitsprotokolle

5.1 Authentisierung

- **Ziel**
 - Eindeutige **Identifikation und Nachweis der Identität**
 - Abwehr von Spoofing-Angriffen
- **Problem:**
 - Nicht nur **Mensch-zu-Gerät** Interaktion, sondern auch
 - **Gerät-zu-Gerät** bzw. **Gerät/Dienst-zu-Gerät/Dienst**
- **Bemerkung:** zunehmende Vernetzung u. Miniaturisierung: **Geräte-zu-Geräte-Kommunikation steigt rapide an!**
- **Notwendig:** Konzepte und Verfahren, um sowohl
 - **menschliche Individuen**
 - als auch **Geräte** (Web-Server, Laptop, Mobiltelefon, ...) und
 - **Dienste** (Dateisystem, Amazon, Bankportal, ...)eindeutig zu identifizieren

Beispiel für die Authentisierung

- Viele Beispiele in der Praxis:
 - Benutzer gegenüber PC (Mensch-zu-Gerät)
 - Mobiltelefon gegenüber Netzbetreiber (Gerät-zu-Gerät)
 - Bankkunde gegenüber Bankautomat (Mensch-zu-Gerät)
 - WWW-Server gegenüber Nutzer (Dienst-zu-Mensch)
 - Laptop gegenüber Access Point im WLAN (Gerät-zu-Gerät)

5.2 Arten der Authentisierung

- **Authentisierung durch**
 - **Wissen:** z.B. Passwörte, PINs, kryptographischer Schlüssel ("Was ich weiß")
 - **Besitz:** z.B. Smartcard, USD-Token, SIM-Karte ("Was ich habe")
 - **biometrische Merkmale:** z.B. Fingerabdruck ("Was ich bin")
 - **Mehrfaktor-Authentisierung:** Kombination von Konzepten:
 - * z.B. SIM-Karte + PIN, Biometrie + Zugangskarte, ...

5.3 Authentisierung durch Passwörter

- **Vereinbartes Geheimnis** zwischen dem Benutzer und dem System
 - Meist speichert das System nur einen kryptographischen Hashwert des Passworts
 - * Zum Beispiel in Unix: /etc/passwd

5.3.1 Angriffe auf Password-Systeme

- Angriffe auf die Passwort-Eingabe:
 - "Über die Schulter schauen"
 - Mitschneiden ("Sniffen") von Passwörtern in einem LAN bei unverschlüsselter Kommunikation; Keylogger um Tastaturkabel
 - Nicht vertrauenswürdige Rechner: Keylogger im System
 - * Zum Beispiel in einem öffentlichen Terminalraum
 - Passwort auf Post-it-Zettel
- Social Engineering-Angriffe
 - Beispiel: Telefonanruf an einen Administrator: "Hallo, hier ist die Sekretärin der Geschäftsführung. Mein Chef braucht dringend das Administrator-Passwort für die Anwendung XYZ!"
- Angriffe auf den Passwort-Speicher
 - Besonders leicht bei unverschlüsselten Passwörtern: Nie Passwörter im Klartext speichern (besser kryptographische Hashwerte!)
 - Nutzen von Logging-Informationen: Ein nicht-existenter Login-Name wie z.B. **gzu%8yp** in einer Log-Datei ist ein lohnendes Ziel.
 - Password-Cracking

5.3.2 Password-Cracking

- Password-Cracker: **Offline-Angriff**
 - Enthalten **Wörterbücher (Dictionaries)**: unter Umständen in verschiedenen Sprachen
 - Unterstützen **Heuristiken**, um Kombinationen aus Sonderzeichen und Wörtern durchzuprobieren: Das Passwort Cars%ten kann leicht ermittelt werden
 - **Brute-Force-Angriff** (vor allem bei zu kurzen Passwörtern erfolgreich)
 - Üblicherweise könnten mit einem Password-Cracker ungefähr 21-25% der Passwörter eines Systems ermittelt werden (Studie).
- Gängige Password-Cracker:
 - Crack, L0phtCrack
 - Cain & Abel, <http://www.oxid.it/cain.html>, für Windows-Systeme.
 - John the Ripper (JtR), www.openwall.com/john, für Unix und Windows

5.3.3 Wahl guter Passwörter

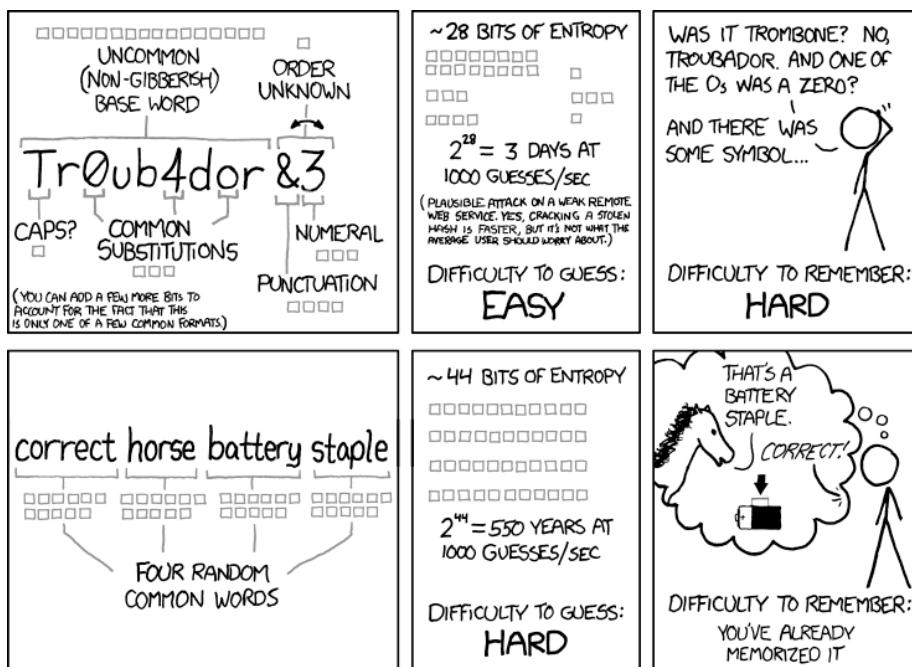
- Richtlinien für die Wahl guter Passwörter
 - 1. Nicht weniger als 8 Zeichen
 - 2. Kein Name oder Wort, das in einem Wörterbuch nachschlagbar ist
 - 3. Mindestens ein Sonderzeichen
 - 4. Keine Folge von Zeichen, die auf einer Tastatur nebeneinander liegen

... und Passwörter sollten in regelmäßigen Abständen geändert werden!

Wahl guter Passwörter

- ▶ Richtlinien für die Wahl guter Passwörter
 1. Nicht weniger als acht Zeichen
 2. Kein Name oder Wort, das in einem Wörterbuch nachschlagbar ist
 3. Mindestens ein Sonderzeichen
 4. Keine Folge von Zeichen, die auf einer Tastatur nebeneinander liegen

... und Passwörter sollten in regelmäßigen Abständen geändert werden!



THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

5.3.4 NIST SP 800-63B (June 2017)

- ”**Verifiers SHALL require subscriber-chosen memorized secrets to not be at least 8 characters in length. Verifiers SHOULD permit subscriber-chosen memorized secrets at least 64 characters in length. Truncation of the secret SHALL NOT be performed.** [...] Verifiers SHOULD offer guidance to the subscriber, such as a password-strength meter [Meters], to assist the user in choosing a strong memorized secret.”
- ”**Verifiers SHOULD NOT impose [...] composition rules** (e.g. requiring mixtures of different character types or prohibiting consecutively repeated characters) for memorized secrets. **Verifiers SHOULD NOT require memorized secrets to be changed arbitrarily (e.g. periodically).** However, verifiers SHALL force a change if there is evidence of compromise of the authenticator.”
- <https://pages.nist.gov/800-63-3/sp800-63b.html> : 5.1.1.2 Memorized Secret Verifiers

5.3.5 Sicherheit von Unix-Passwörtern

- Design-Fehler im ursprünglichen Unix:
 - Passwort-Datei /etc/passwd ist zwar verschlüsselt, aber **world-readable!**
 - Gefahr eines Wörterbuchangriffs durch einen Password-Cracker (offline)!
 - Lösung des Problems: shadow-Password Datei; /etc/passwd enthält dann Dummy-Einträge
- Sicherheit von Unix-Passwörtern
 - Unix-Passwörter können maximal acht Zeichen lang sein,
 - 95 druckbare Zeichen möglich

Es gibt somit $95^8 \approx 2^{52.6}$ Möglichkeiten zur passwordwahl. Eine gut ausgestattete Organisation (z.B. NSA) kann also **jedes** Unix-Passwort knacken.

5.3.6 Zusammenfassung: Authentisierung durch Passwörter

- Password-Authentisierung angreifbar
 - Dictionary-Angriffe
 - Sniffing
 - Social Engineering
 - Angriffe auf die Passwort-Speicherung
- **Lösungen: u.a.**
 - **Einmal-Passwort**-Verfahren, OTP (One-Time-Passwort) z.B. S/Key (Hash-Kette) oder RSA SecureID ("Key Fob")
 - Authentisierung zwischen Systemen; **Ticket-basiert** mit Nachweis, dass man der Eigentümer des Tickets ist (z.B. Kerberos)
- **Vielen dieser Lösungen ist gemeinsam:** Sie basieren auf einem Challenge-Response Verfahren.

5.4 Hardware-basierte OTP-Verfahren: ID-Token

- **OTP**-Verfahren zur Authentisierung beim Server. Beispiel: Online-Banking
- Benutzer erhält ein Hardware-Token
- **Token** besitzt eindeutige Nummer; Server kennt diese
- Token berechnet periodisch eine neue Zahl (Code), z.B. alle 60 Sekunden, **Code ist das OTP**
- Tokencode hängt von einem Seed ab, der Zeit, der Seriennummer des Tokens und ggf. einer PIN
- Benutzer: Eingabe des OTP an einem Terminal, dazu: ablesen des **Passworts vom Display des Tokens**
- **Server** muss Seed, Zeit, Seriennummer, PIN kennen, generiert ebenfalls das Passwort und vergleicht beide

Beispiel: RSA SecurID Token

- Basis für 2-Faktor-Authentisierung (Wissen und Besitz)
 - **Zeit-synchronisierte** Vorgehen
 - * Synchronisation von Server (RSA ACE/Server) und Token
 - Admin des Servers richtet Benutzer-Account ein, mit:
 - * Token-Nummer und 64-Bit Seed **s**

- * Seed s wird auch auf Token gespeichert
- * Token wird an Benutzer ausgegeben
- Erzeugen von OTPs:
 - * alle 60 Sekunden generieren Token und Server neues Passwort:
AES-Hashwert: Tokencode = AES(TokenID | s | Zeit)

Beispiel: S/Key

- Einmal-Passwörter über Hash-Ketten generieren
- Benutzer und System vereinbaren Passwort s
- Hash-Funktion f wird i -mal wiederholt: f^i
- Benutzer erhält s und einen PDA mit f
 - Alternativ: Benutzer erhält für n Einlogvorgänge $f^i(s)$ für $0 < i < n$
- System fordert mit n heraus (und merkt sich $i=n$)
 - Benutzer antwortet mit $f^i(s)$
- Beim nächsten mal fordert System mit $i:=i-1$ heraus
 - Benutzer antwortet mit $f^{i-1}(s)$ (nicht aus $f^i(s)$ ableitbar!)

Probleme von S/Key

- Wie alle Passwort-Mechanismen anfällig gegen Middle-Person-Angriff
- Untergeschobener Server kann Passwort für j abfragen mit $j < i$; daraus kann $f^i(s)$ berechnet werden
- Lösung: Server-Authentisierung

5.5 Authentisierung durch Biometrie

- ”Gilead besetze die nach Efraim führenden Übergange über den Jordan. Und wenn efraimische Flüchtlinge (kamen und) sagten: Ich möchte henüber!, fragten ihn die Männer aus Gilead: Bist du ein Efraimiter? Wenn er Nein sagte, forderten sie ihn auf: Sag doch einmal ‚Schibboleth‘. Sagte er dann ‚Sibboleth‘, weil er es nicht richtig aussprechen konnte, ergriffen sie ihn und machten ihn dort an den Furten des Jordans nieder. So fielen damals zweiundvierzigtausend Mann aus Efraim.” - Alte Testament, Richter 12:5-6
- Erster (?) dokumentierter militärischer Einsatz eines Protokolls zur Authentisierung basierend auf einem biometrischen Merkmal:

- Hier: der Akzent
- **Was ich bin** ("something you are")
- **Biometrisches Merkmal:** Verhaltenstypische oder psychologische Eigenschaft eines Menschen, die diesen eindeutig charakterisieren.
- Anforderungen an biometrische Merkmale:
 - **Universalität:** Jede Person besitzt das Merkmal.
 - **Eindeutigkeit:** Merkmal ist für jede Person verschieden.
 - **Beständigkeit:** Merkmal ist unveränderlich
 - quantitative **Erfassbarkeit** mittels Sensoren
 - **Performance:** Genauigkeit und Geschwindigkeit
 - **Akzeptanz** des Merkmals beim Benutzer
 - **Fälschungssicherheit**
- **Fingerabdruckerkennung:**
 - Weit verbreitetes Verfahren für viele Anwendungen: preiswerte, kompakte Sensoren sind verfügbar (auf Mobiltelefon, PDA, Laptop, ...)
- **Schreibdynamik** ist besonders geeignet für Willenserklärung (z.B. Signaturen): Unterschreiben als bewusster, willentlicher Akt
- **Iris-Erkennung** sehr fälschungssicher, aber aufwendige Technik und geringe Akzeptanz (vgl. Fraport-Pilot-Projekt)
- **Gesichtskontrolle:**
 - Gute Akzeptanz, aber z.Zt. noch nicht ausgereift, vgl. <http://www.bsi.bund.de/literat/studien/biop/biopabschluss.pdf>; Studie zu Gesichtserkennungssystemen zum Einsatz bei Lichtbildausweisen

5.5.1 Problembereiche

- Abweichungen zwischen Referenz- und Verifikationsdaten sind unvermeidlich.
- **Zwei Fehlertypen:**
 - Berechtigter Benutzer wird angewiesen ⇒ **Akzeptanzproblem (false negative, FRR false rejection rate)**
 - Unberechtigter wird authentisiert, Kontrollen zu locker ⇒ **Sicherheitsproblem (false positive, FAR false acceptance rate)**

Leistungsmaße zur Bewertung der Güte eines Systems

- Ergebnisse einer dreijährigen Studio BioTrustT: www.biitrust.de : Aktuelle Produkte besitzen nich **erhebliche Fehlerraten** (Stand 2002, aber durchaus noch aktuell).

Chapter 6

Schwachstellen, Bedrohungen, Angriffe

6.1 Sicherheitsprobleme

- Zugriffskontrolle setzt Sicherheitsziele (wie z.B. Integrität, Vertraulichkeit) durch
- Nun: Umgehung der Mechanismen zur Zugriffskontrolle
- Typische Sicherheitsprobleme:
 - Buffer-Overflows (Unchecked Buffer)
 - SQL-Injection
 - XSS-Probleme
 - TOCTOU-Angriffe
 - Bösartige Programme (Viren, Würmer, Trojanische Pferde)
 - Spoofing
 - DoS-Angriffe

6.2 Verwundbarkeit und "Exploit"

- Nicht jede Verwundbarkeit ist einfach zu nutzen
- "Exploit": Programm/Verfahren zur Nutzung der Verwundbarkeit

Heiß diskutierte Frage:

- Soll man Sicherheitsprobleme veröffentlichen?
 - Ja, weil sonst nicht in Ordnung gebracht wird?

- Nein, weil sie vermutlich sofort ausgenutzt wird?
- Soll man den "Exploit" gleich mitliefern?

6.3 CERTs und Advisories

- CERT = Computer Emergency Response Team
 - Behandelt Sicherheitsvorfälle, IT-Sicherheitsfeuerwehr
 - Typische CERT-Betreiber: Große Unternehmen, Behörden, Staaten
- Advisory: Recherchierte Mitteilung über Sicherheitsproblem und mögliche Abhilfen (z.B. zu finden unter: <https://www.kb.cert.org/vuls>)
 - Wird meist erst veröffentlicht, nachdem Hersteller Abhilfen (Patches) zur Verfügung gestellt haben.
 - Etwas schwerfälliger Vorgang
 - Bleibt meist nebulös, worin genau das Problem besteht
- Problem: Patches lassen sich analysieren
 - Exploits stehen ca. zehn Stunden nach Advisory zur Verfügung
 - "Patch race"

6.4 Software im Netz

- Server sitzen da und warten auf Verbindungen
 - Meist identifiziert über IP-Adresse und Port-Nummer
- "Remote exploit": Anfrage an Server, die unmittelbar zum unkontrollierten Zugriff führt
- "Local exploit": Angriff wird erst nach Authentisierung und Erlangung eigener Zugriffsrechte (Programmausführung) möglich

6.5 Buffer-Overflows

- Häufigste Einbruchsmethode in Server (insbesondere in Web-Server)
- Altbekanntes Problem (schon in den 60er Jahren bekannt)
- "Attack of the decade" (Bill Gates)
- Die meisten Viren/Würmer nutzen Buffer-Overflows aus (Morris-Wurm, Code Red, Blaster)
- **Ziel:** Einschleusen von Code
- Ausnutzen von Programmierfehlern

6.5.1 Nicht korrekt geprüfte Eingaben

```
char buf[42];
gets(buf);
```

- **Problem:**
 - C-Funktion gets überprüft nicht die Länge der Eingabe (**”unchecked buffer”**)
 - Ist die Länge der Eingabe größer als 41 (Nullterminierung beachten!), dann wird umgebener Speicher überschrieben.
 - Lokale Variablen sind auf dem Call-Stack
- Durch Verwendung von *fgets(buf,42,stdin)* wird das Problem vermieden

Ein lokaler ”unchecked buffer” (historisches Beispiel)

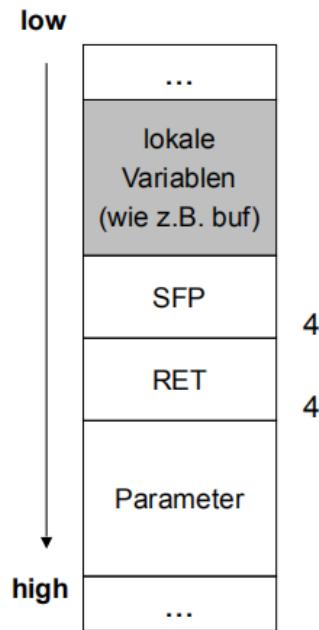
- UNIX Version 6 (c.a. 1975), Login-Programm:

```
char user[100], passwd[100], correct[100];
gets(user);
getpwnam(...);
strcpy(correct, ...);
gets(passwd);
if (strcmp(crypt(passwd, ...), correct)) ...
```

- Eingabe eines 108 Zeichen langen Passworts schreibt über das Ende von *passwd* hinaus und überschreibt den Vergleichswert *correct* aus dem Passwort-File.
- Speziell fabriziertes Passwort, das in seine eigenen letzten 8 Zeichen verschlüsselt, ist Generalschlüssel!

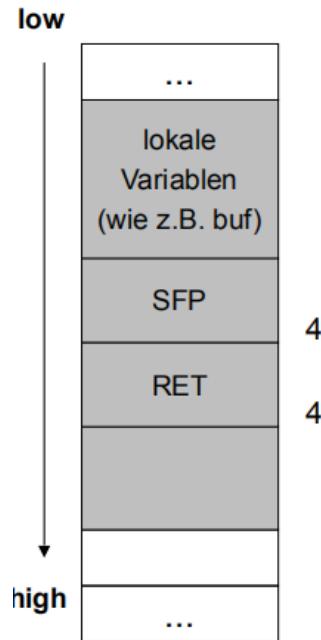
6.5.2 Call-Stack

- Lokale Variablen von Unterprogrammen werden auf Call-Stack abgelegt
- **Beispiel:** Stack für einen Unterprogrammaufruf bei der x86-Architektur
- RET: Rücksprungadresse
- SFP: Stack Frame Pointer



6.5.3 Überschreiben der Rücksprungadresse

- RET kann durch entsprechend präparierten Eingabestring so manipuliert werden, dass Rücksprung in den Exploit-Code führt
- Exploit-Code wird in Stack untergebracht
- **Beispiel für Exploit-Code unter Linux/Unix:** *execl* zur ausführung einer Shell auf angegriffenem System
- Nützlich, wenn das Programm mit der Buffer-Overflow-Verwundbarkeit mit root-Rechten läuft (Daemon oder s-Bit für den Besitzer root gesetzt)



6.5.4 Präparation des Eingabe-Strings

String-Anfang			String-Ende
Füllzeichen	Neue Rück-sprungadresse (u.U. mehrfach)	NOPs	Code des Angreifers (z.B. Shell)

Landing-Pad aus Nulloperationen (NOPs): Landezone vor dem eigentlichen Angriffscode, weil die genaue Adresse des Angriffscode oft nicht vorhersagbar ist.

6.5.5 Fallstricke für einen Angreifer

- Platz auf Stack zu klein? Eher unproblematisch:
 - Maschinencode ist **kompakt**
 - **Viele Funktionen** stehen zur Verfügung, **einfache Nutzung** vorhandener Dienste:
 - * z.B. Windows: viele Bibliotheken (DLLs) stehen bereits im Speicher und sind an den befallenen Prozess gebunden

- * **Nutzung aller API-Funktionen**, die bereits zum Programm gebunden sind. Z.B. *LoadLibrary*: **Nachladen beliebiger Funktionen** möglich!
- Was ist, wenn die Rücksprungadresse oder der Exploit-Code spezielle Zeichen wie '\0' oder '\n' enthält?
 - Viele Möglichkeiten, die gleiche Funktion zu kodieren

6.5.6 Gegenmaßnahme: NX-Bit

- Idee: Ausführen von Code auf dem Stack verbieten
 - Erweiterungen in aktuellen Prozessoren (AMD64: NX, Intel x86: Execute Disabled/ XD)
 - Windows: Data Execution Prevention (DEP)
 - Achtung: Datenbereiche z.B. für Shared Libraries oder Code-Generierung (JIT-Compiler!)
 - Einzelne Programme benutzen sogar Stack für Code
- Gegen-Idee: "Rücksprung" in Bibliothek, geeignete Parameter können den gleichen Effekt haben wie Code auf dem Stack
 - Aber erhebliche Erschwerung komplexer Angriffe

6.5.7 Gegenmaßnahme: Canaries

- Funktionsprolog legt einen **zufälligen** Wert (Canary) unmittelbar neben die Rücksprungadresse
- Vor Rücksprung wird überprüft, ob Canary nicht verändert worden ist
 - Wenn nicht: Log-Meldung/Abbruch
- Windows: Stack Cookies
- Nachteil: Kosten für Prozeduraufruf steigen

6.5.8 Gegenmaßnahme: ASLR

- **Problem:** Angreifer springt in Bibliothek, beispielsweise Aufruf von *system()* in Unix/Linux
- **Lösungsidee:** Address Space Layout Randomisation (ASLR)
- Z.B. Shared Libraries (Bibliotheken), Stack an **zufälligen** Adressen positionieren
- Angreifer kann schlecht die Position von Systemcalls erraten

- Unterstützung in modernen OS wie BSD Unix, Linux, Windows, Android, iOS
- Reaktive Maßnahme - keine Prävention des eigentlichen Problems

6.5.9 Fazit: Buffer-Overflows

- **Haupt-Ursache für Buffer-Overflows:** Programmierfehler: Verwendung von vordefinierten Bibliotheks Routinen in Programmiersprachen wie **C oder C++** ohne Bereichsprüfung:
 - `strcpy()`, `strcat()`, `gets()` in C
 - Es gibt Alternativen `strncpy()`, `strncat()`, `fgets()`
- **Bemerkung:** Die **meisten BS-Dienste** (Unix, Windows) sind in **C, C++ programmiert**. (z.B. Windows XP: c.a. 40 Millionen Lines of C-Code - Richtwert 5-50 Bugs pro 1000 Lines of Code!)
- Fast alle Würmer nutzen Buffer-Overflows aus

6.5.10 Ausblick: Buffer-Overflows

- So langsam haben sich die SW-Hersteller besser auf die Problematik eingestellt
 - Werkzeuge zur statischen Software-Analyse (HP-Fortify SCA, Coverity Prevent, IBM AppScan)
 - Was ist mit Embedded Systems?
- Nun geht es für einen Angreifer an die etwas höher hängenden Früchte wie z.B.:
 - Head-Overflows, Integer-Overflows
 - Race Conditions
 - Architekturelle Lücken: Mangelnde / inkonsistente Zugriffskontrolle in Multi-Tier-Applikationen (z.B. SOA), fehlerhafte Umsetzung von Sicherheitsprotokollen wie z.B. WPA2-Lücke (Krack; Okt. 2017)

6.6 SQL-Injection

- Viele Web-Anwendungen (z.B. PHP) speichern Kundendaten, Login-Informationen in Datenbanken
- Die Daten werden von den Benutzern in Eingabefelder eingegeben wie z.B. E-Mail-Adressen
- PHP bietet z.B. eine bequem benutzbare Datenbankschnittstelle, um die eingegebenen Daten direkt in die Datenbank auf dem Web-Server zu schreiben
- Wie bei den Buffer-Overflows: **Angreifer definiert Daten!**

6.6.1 ...Angreifer gibt die Daten ein

Beispiel: Eingabefeld für eine E-Mail-Adresse in einem Web-Formular für Kunden der Firma "Unsicher GmbH"

Angenommen, PHP-Anwendung nutzt SQL-Statement:

```
SELECT email , passwd , login_id , name  
FROM members  
WHERE email='Daten vom Netz '  
  
unsere Eingabe = bla@tzi.de'; DROP TABLE members;--  
Im SQL-Statement:
```

```
... WHERE email=bla@tzi.de' ; DROP TABLE members;--'
```

6.6.2 Fazit

- Ursache: nicht validierte Eingaben werden als **Programmcode** ausgeführt
 - Variante: PHP- oder Perl-Code
- SQL-Injection-Angriffe sind nicht einfach durchzuführen:
 - Woher kennt der Angreifer die Namen der DB-Relationen und Attribute? (PHP-Fehlermeldungen helfen allerdings)
 - Oft nur Anrichten von Schaden (→ Datenintegrität)
- Trotzdem: Fast täglich Meldungen über neue SQL-Injection-Angriffe in Mailinglisten für Sicherheitsprobleme wie z.B. Bugtraq

6.7 Cross Site Scripting (XSS)

- Webseiten können Skripte enthalten
 - Werden auf Kunden-Browser ausgeführt
 - Haben Zugriff auf Cookies der Website
- Webseiten können benutzerdefinierte Daten enthalten
 - Vorherige Eingaben eines anderen Benutzers
 - Evtl. auch unerwartete Reaktion auf URL-Parameter
- Angriff: Angreifer unterschiebt Opfer ein Skript

6.7.1 Durchführung eines Angriffs

- **Beispiel:** `http://auction.example.com/filename.html` liefert eine Webseite zurück mit der Meldung *404 page does not exist: filename.html*.
- Angreifer schickt dem Opfer nun einen präparierten Link (z.B. per E-Mail): `http://auction.example.com/;script;alert('hello');/script;`
- Beim Abruf des Links wird das Skript zurückgeliefert und aufgerufen (und zwar nicht auf dem Webserver, sondern auf dem Rechner des Opfers)

6.7.2 Beispiel für verwundbaren JSP-Server-Code

```
<c:if test="${param.sayHello}">
    <!--Let's welcome the user ${param.name} -->
    Hello ${param.name}!
</c:if>
```

param.name wird einfach ungeprüft vom Server zurück zum Client gespielt
(Beispiel aus Buch von Chess und West)

6.7.3 Fazit: XSS-Angriffe

- Angreifer kann auf Opferrechner beliebige Javascript-Befehle ausführen:
 - Cookie auslesen und woanders ablegen
 - Eingabefenster für Passwörter simulieren
- Session=Übernahme möglich durch Stehlen eines Session-Cookies (im Auktionsbeispiel evtl. besonders kritisch)
- Angegriffener: primär der Nutzer, aber z.B. durch Rufschädigung auch der Betreiber der Website
- Betreiber der Website sollte solche Fehler beseitigen
- **Schutz:** Validation von Eingaben (durch den Betreiber der Web-Site)

6.8 Zwischenfazit

- Ursache für Buffer-Overflows, SQL-Injection-Angriffe sowie XSS-Angriffe:
ungeprüfte Eingaben
- ALSO: Überprüfen von Eingaben, Überprüfen von Eingaben, Überprüfen von Eingaben!!!

- Nicht nach Problemen suchen (der Angreifer hat mehr Phantasie), sondern nur gesunde Eingaben durchlassen (vgl. UTF-8-Angriffe)

6.9 TOCTOU-Angriffe

- TOCTOU: Time of Check, Time of Use
- Anderer, allgemeiner Begriff: **Race Conditions** (Wettlaufsituationen)
- Basiert auf Nebenläufigkeit in Betriebssystemen und Anwendungen
- Operationen oft **nicht atomisch**
- **Beispiel:** *mkdir* in alten Unix-Systemen
 - 1. Schritt: Anlegen des inode für das Verzeichnis
 - 2. Schritt: Festlegen des Besitzers (des Verzeichnisses)
- Angreifer kann nach erstem Schritt in einem zweiten Prozess einen Link auf */etc/passwd* setzen,
- In Schritt 2 wird dann der Angreifer zum Besitzer von */etc/passwd*
- TOCTOU-Angriffe: Es kommt für den Angreifer auf den richtigen Zeitpunkt an
 - Ausprobieren, Angriff evtl. oft wiederholen, Angriff automatisieren
- Häufiges Muster für Verwundbarkeit:
 - 1. Überprüfung der Zugriffsrechte (Time of Check)
 - 2. Durchführung der sicherheitskritischen Operation (Time of Use)
- Zugriffskontrolle kann ausgehebelt werden

6.10 Spoofing

- Statt Berechtigung zu erschleichen: Vorgeben, ein Berechtigter zu sein
- Manche Systeme prüfen nur die Absender-IP-Adresse
 - Bei UDP extrem leicht zu fälschen
 - Bei TCP schwieriger, aber in bestimmten Fällen möglich
- Session hijacking: Verbindung nach der Authentisierung übernehmen
 - Einfache Gegenmaßnahmen in TCP, wenn kein Abhören möglich
 - Echter Schutz nur kryptographisch möglich

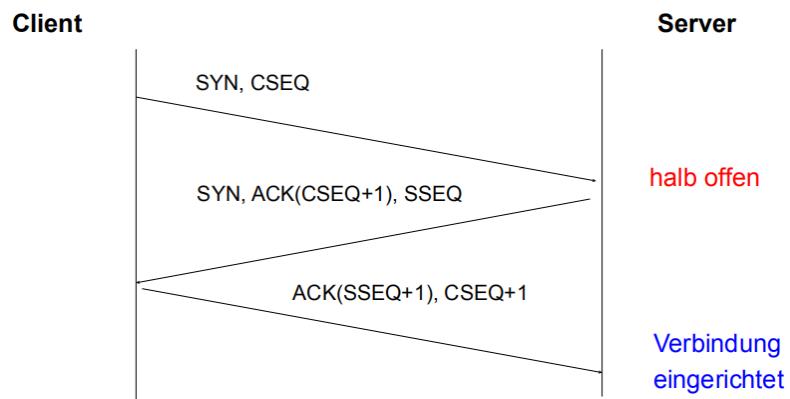
6.11 (D)DoS-Angriffe

- Denial-of-Service: Angriff auf Sicherheitsziel
- Server zum Absturz bringen
 - Programmfehler wie unchecked buffer nutzen
- Sicherheitsmaßnahmen aktivieren
 - Z.B. Account-Sperre nach dreimaliger Fehlereingabe des Passwortes
- Server (oder Netz) überlasten
 - DDoS: Distributed DoS: Farm übernommener "Zombies"

6.12 Syn-Flood-Angriff

- Ziel: Server verstopfen, ohne Identität der angreifenden Maschinen preiszugeben
- TCP: Schutz durch three-way-handshake
 - Verbindung kommt nur zustande, wenn Antwort vom Client stimmt
 - Absenderadresse kann nicht leicht gefälscht werden
- Idee: Nur SYN-Paket absenden
 - Absenderadresse leicht zu fälschen
 - Server muss Zustand halten (Timeout nach Minuten)
- 1 Gbit/s \approx 3E6 Pakete/s \approx 0.5E9 halboffene Verbindungen

6.12.1 TCP-Handshake



6.13 Trojanische Pferde

- Analog zur griechischen Sage:
 - Ein nach außen hin nützlich erscheinendes Programm, das eine verborgene Schadfunktionalität enthält
- Selbstständiges Programm (im Gegensatz zu Virus)
- "Werkzeuge zur Fernwartung" (remote administration)
- **Beispiele:**
 - BackOrifice (Windows 95/98, August 1998, Cult of the Dead Cow), BO2K (Windows, Juli 1999)
 - SubSeven: Funktionalität: Manipulieren der Windows Registry, entferntes Lesen und Schreiben von Daten, Keylogger, Port Scans vom Opferrechner aus, ...
- Ständige Veröffentlichung neuer Trojanischer Pferde auf Cracker-Webseiten

6.13.1 Weitere Varianten von Trojanischen Pferden/Malware

- Spyware:
 - Ziel: **Verdecktes Ausspähen** des Nutzers
 - Oft in Form eines Trojanischen Pferdes
 - Beispiele: "Web-of-Trust"-Add-on für Browser, Mitschneiden von Browser-Verläufen; Apps mit unautorisiertem Mikrofon- und Kameraszugriff für Smartphones
- Ransomware
 - Englisch: "ransom" - Lösegeld
 - Expresssoftware: z.B. Entschlüsseln von Daten gegen Lösegeld
 - **Beispiel: WannaCry**



6.14 Social Engineering

- Angreifer überzeugt das Opfer, etwas zu tun, was er will
- Beispiel:
 - Erfragen von Patientendaten per Telefon
 - Phishing
 - Gefälschte E-Mail-Adressen (E-Mail kommt vermeintlich von einem Bekannten, vgl. Verbreitung von Würmern)
- **I Love You** nutzte mehrfach Social Engineering-Techniken, z.B.
 - *LOVE-LETTER-FOR-YOU.txt.vbs* (Windows lässt oft bekannte Endungen weg, so dass das VBS-Script wie eine Text-Datei aussah)
- Kevin Metnick: "I was so successful in that line of attack that I rarely had to resort to a technical attack"
- Schutz: Schulung von Mitarbeitern, Sensibilisierung

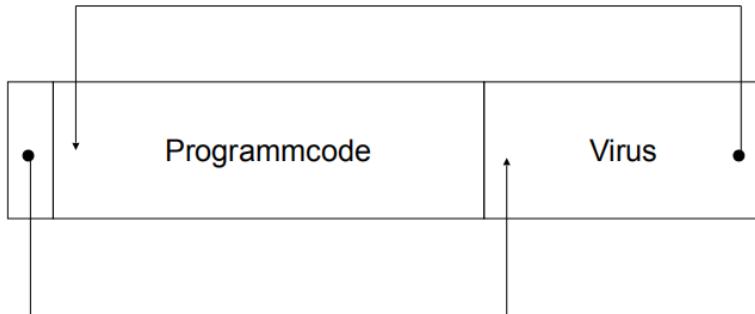
6.15 Viren

- In der **Biologie** ist ein Virus
 - Ein Mikro-Organismus, der auf eine **lebende Wirtszelle** angewiesen ist,
 - Keinen eigenen Stoffwechsel besitzt und
 - Fähig ist, sich **zu reproduzieren**
- Eigenschaften sind direkt auf Computerviren übertragbar: **Computer-virus:** von F. Cohen 1984 eingeführter Begriff
- Historischer Begriff
- **Computervirus:**
 - Nicht Selbstständiges Programm, d.h. es **benötigt Wirt**
 - Beispiel für Wirt: meist ausführbare Dateien
 - Besitzt **Kopierfähigkeit**, ggf. auch mutierend
 - Enthält i.d.R. einen **Schadensteil**, d.h. Code zur Durchführung von Angriffen
 - **Kann Auslöser** enthalten, d.h. Bedingung zur Aktivierung des Schadensteils (logische Bombe) wie z.B. Datum
 - Enthält i.d.R. eine **Kennung**, z.B. Zeichenketten
 - Virus-Code dient häufig **zur** gezielten Angriffs-(Einbruchs)-vorbereitung: u.a.
 - * Infos sammeln, Ports öffnen, Shell-starten, ...

6.15.1 Virentypen

- **Programmviren** (Link-Vieren): infizieren ausführbare Programme (z.B. .exe); Virus-Start mit dem Programm
- **Bootsektor-Viren**: Virus wird resident geladen
- **Makro- und Daten-Viren**: u.a. bei MIME, .ps, .doc, .xls
 - **Interpretierbare Ausführung** von Code. Z.B. Starten eines Dateitransfers (von Festplatte) via FTP
 - Häufig: **Verbreitung über das Netz**, via E-Mail Attachments, Buffer-Overflow-Angriffe, etc.
- **Retro-Viren**: gegen das Immunsystem (z.B. Viren-Scanner), mögliches Angriffsziel: **Deaktivieren** des Viren-Scanners
- **Angriff anderer Systeme**: Smartphone-Viren?
- Viren **benutzen oft Spam**, um sich schneller zu verbreiten (Würmer?)

6.15.2 Funktionsweise: Programmavirus



6.15.3 Gegenmaßnahme: VirensScanner

- "Signatur" eines Virus erkennen
- Dateien im Eingang und Ausgang (z.B. Mail) und vor der Ausführung prüfen
- Aufwendig
- Tiefer Eingriff ins Betriebssystem
- Aktualität der Signaturen ist entscheidend
- Polymorphe Viren erfordern "probeweises Ausführen"
- VirensScanner sind komplex und haben eigene Verwundbarkeiten

6.16 Würmer

- Selbstständig ablauffähiges Programm,
- Nutzt die Infrastruktur eines Netzes, um sich **selbstständig zu verbreiten**
- Ausgangspunkte für Wurmangriff häufig: **Buffer-Overflow-Angriff** auf Systemprozesse, die ständig rechenbereit sind oder in regelmäßigen Abständen aktiviert werden
- **Beispiele:**
 - 1988: **Morris-Wurm** (befiel Unix-Rechner): **Buffer-Overflow**: fin-gerd
 - 2000: **I Love You-Wurm**
 - 2001: **Code Red Wurm**: Buffer Overflow in MS-IIS
 - 2003: **W32/Lovsan/MS Blaster-Wurm**
 - 2008: **Conficker**

6.16.1 Gegenmaßnahme: Firewall

- Nur erwünschte Kommunikation zulassen
 - Windows: zu viele Interna sind von außen zugreifbar; Firewall be-triebsnotwendig
 - Problem: Wie erkennen, was erwünscht ist?
- IDS/IPS: Signaturen von Angriffen erkennen
 - Vgl. VirensScanner
 - Insbesondere: Erkennung von Angriffen, nur wenn Signatur vorhan-den
 - Was ist aber mit unbekannten Angriffen?

6.17 Können wir gewinnen?

- Angreifer braucht nur **eine** Sicherheitslücke
- Verteidiger muss jedes Loch finden und stopfen
- Reale Systeme sind zu komplex, dass sie fehlerfrei sein können
- Eindringen: **erschweren, erkennen, bekämpfen**

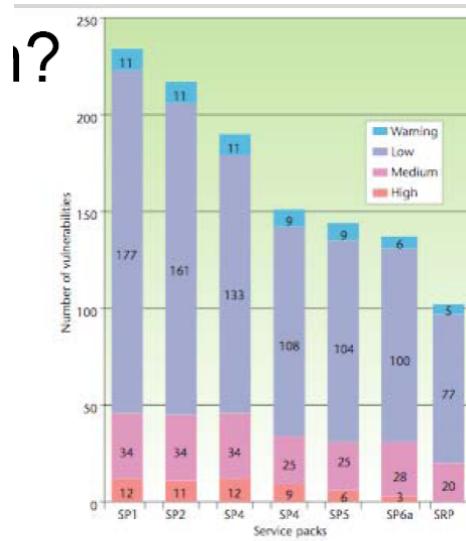


Figure 1. Security flaws. The number of security flaws in Windows NT 4.0 Service Packs 1 through 6a and the post-SP6a Security Rollup Package (SRP) according to risk levels.

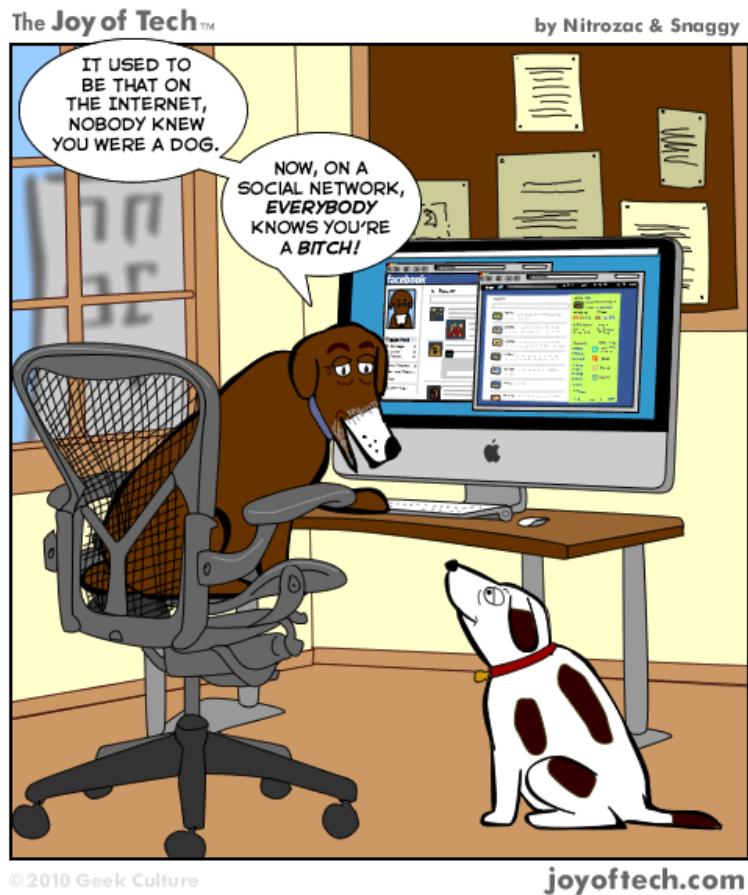
6.18 Die vier Phasen eines Angriffs

- Aufklären (Recon)
- Eindringen (Exploit)
- Spuren verdecken (z.B. rootkit)
- Ausnutzen

- Gute Sicherheitstechnik versucht, alle diese Phasen zu **erschweren, erkennen, bekämpfen**

Chapter 7

Netz-Sicherheit



7.1 Software im Netz

- Server sitzen da und warten auf Verbindungen
 - Meist identifiziert über IP-Adresse und Port-Nummer
- Client stellt Verbindung her, stellt Anfrage
- ”Remote exploit”: Anfrage an Server, die unmittelbar zum unkontrollierten Zugriff führt
- ”Local exploit”: Angriff wird erst nach Authentisierung und Erlangung eigener Zugriffsrechte (Programmausführung) möglich

7.2 Angriffe auf verschiedenen Schichten: Schicht 1

- Ethernet-Repeater: jede Art von Kommunikationsleitung:
 - Abhören (Angriff auf Vertraulichkeit)
 - Verändern, Einschleusen
 - * (allerdings meist einfacher auf höheren Schichten)
- Gegenmittel: Quantenkryptographie
 - Beobachtung verändert Phänomen
 - * Angriff kann erkannt werden
 - Nützlich vor allem zur Übertragung von Schlüsselmaterial

7.3 Angriff auf verschiedenen Schichten: Schicht 2

- Switches: Verunreinigung der Switch-Datenbank
 - Verkehr wird an alle Ports verteilt
- ARP-Spoofing
 - Abhören (Angriff auf Vertraulichkeit), verfälschen
 - Tools: Dsniff, Ettercap
- Vortäuschen einer fremden MAC-Adresse
 - Umgehung Zugriffsschutz (z.B. WLAN)
 - Tools: ifconfig ... ether ...

7.4 Angriffe auf verschiedenen Schichten: Schicht 3

- Router: Verunreinigung des Routing-Protokolls
 - Verkehr wird umgeleitet
 - Abhören (Angriff auf Vertraulichkeit)
 - Verkehr wandert in schwarzes Loch
- Vortäuschen einer fremden IP-Adresse
 - Umgehung Zugriffsschutz (z.B. bei NFS)
 - Einschleusen von Daten (Session hijacking)
- Loose-Source-Routing
 - Paket wird nicht an eigentliche Adresse zurückgeschickt, sondern via umgedrehter Source-Route
 - Hebelt TCP-Handshake aus

7.4.1 Finding victims: Scanning

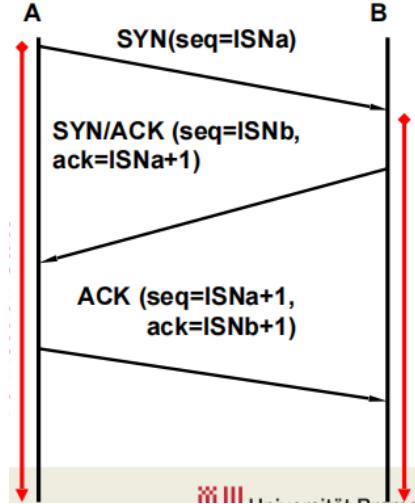
- Reconnaissance: Finding potentially vulnerable Hosts
- IPv4 address space is densely populated
 - Of $\sim 4.3E9$ IPv4 addresses, $\sim 3.7E9$ can be used as unicast addresses; of these $\sim 3.4E9$ are allocated (91%)
 - Of these, $\sim 2.5E9$ are routed globally (67% of the useable, 74% of the allocated address space)
 - $> 0.5E9$ of these have a web server (netcraft.com), which is $> 20\%$!
- IPv6 makes scanning much harder
 - $> 1E34$ addresses are allocated (0.03% of the currently useable space)
 - Enumerating these at 1Gbit/s takes $\sim 1E20$ years
 - However, there are other ways to collect IPv6 addresses, e.g.
 - * DNS analysis
 - * Snooping traffic

7.5 Angriffe auf verschiedenen Schichten: Schicht 4

- RST-Attacks
 - Abbrechen einer TCP-Verbindung
 - Kann Routing-System (BGP) empfindlich stören (DoS)
- SYN-Flooding
 - Aufbau von Zustand
 - Überlastung stört normale Verbindung (DoS)

7.5.1 RN1 Wiederholung: Three-way handshake

- ISN (Initial Sequence Number): 32 bit
- Sequence Number (seq)
- Auswahl der ISN gesteuert jeweils durch jede Seite
- Echo zurück in Acknowledgement Number (ack)



7.5.2 SYN-flood attack

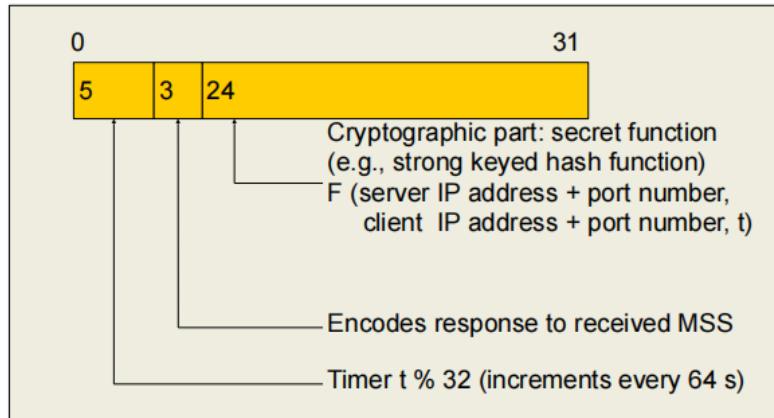
- Objective: clog server
- Bonus: do this without giving hints about identity of attacker
- TCP: protected by three-way-handshake
 - Connection is only completed when peer answers with correct sequence number
 - Cannot easily fake source address
- Idea: send SYN packet only
 - Easy to fake source address
 - Server needs to establish state (Timeout after several minutes)
- $1\text{Gbit/s} \approx 3 \times 10^6 \text{ packets/s} \approx 0.5 \times 10^9 \text{ half-open connections}$

7.5.3 Countering Resource Depletion

- Attackers attempt to bind more resources on the target system that require to mount the attack
 - Make your system perform many and/or expensive computational operations
 - * Particularly relevant with security checks (e.g. signature or certificate validation)
 - Make your system create state information
 - (Make your system transmit data, preferably to somebody else)
- Issue: distinguishing legitimate work from attack
 - There is not necessarily a well-defined user behaviour
 - Example HTTP: Botnet fetching pages, search crawlers, site replication (wget)
 - * Some websites inspect the HTTP User-Agent header and deny access to bots
- General Approach:
 - Avoid creating (much) states early on the server side
 - Make the client work harder (client-side easier to scale anyway)

7.5.4 Example: TCP SYN Cookies

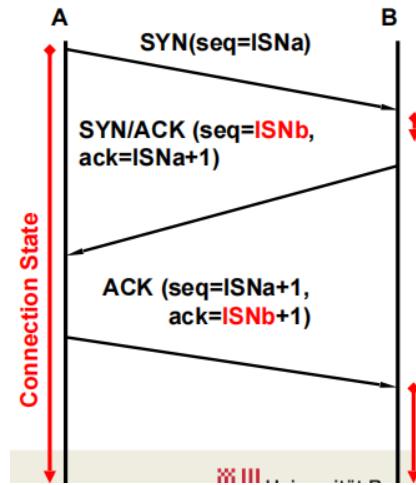
- Normal TCP operation when a SYN packet comes in
 - Create protocol control block (PCB), choose random initial sequence number, set cleanup timeout (in case you never get an ACK), send SYN-ACK back
 - State will last until timeout expires
- TCP SYN Cookie idea (D. J. Bernstein, 1996)
 - Do not create state
 - Encode the local state you would create in 32-bit sequence number
 - * Part of this is protected cryptographically
 - Send SYN-ACK
 - If ACK comes back: recreate state from Acknowledgement number
 - If no ACK comes back: nothing lost except for few CPU cycles
 - Equalizes the burden
 - * Attacker needs to respond and thus bind local resources
 - * Attacker can no longer use random source addresses



- Issues: limits TCP option negotiation capabilities
 - E.g. large windows, SACK

7.5.5 SYN Cookies

- Nur B kann aus den Parametern ISNb ausrechnen
- Erst ACK legt Zustand an
- Weist Empfang der "geheimen" ISNb nach
- Adresse von A authentisiert (return routability)



7.6 Angriffe auf verschiedenen Schichten: Schicht 7

- DNS-Spoofing
 - Verunreinigen der Caches von DNS-Servern
- Email-Spoofing
- Web-Spoofing, Phishing
- Angriffe auf Programme: Buffer Overflows etc.

7.6.1 Gemeinsamkeiten

- On-Path-Angreifer kann abhören
 - Durch gezielte Angriffe kann man "on-path" gelangen
 - Abhilfe: Verschlüsselung (Kryptographie)

- Identitätsbehauptungen können gefälscht werden
 - Abhilfe: Authentisierung
 - Muss resistent gegen Abhören sein
 - * Kryptographische Authentisierung

7.7 The Internet Threat Model

(Dolev-Yao threat model, 1983)

- Assumption: The end-systems are not compromised
 - There are ways to minimize damage even in this case, e.g. perfect forward secrecy
 - Can validate the assumption with *attestation*
- However, the communications channel is completely compromised, i.e. attacker can:
 - Read any PDU
 - Undetectably remove, modify, inject any PDU
 - * Including PDUs that appear to be from "trusted" machine

7.7.1 Types of attacks

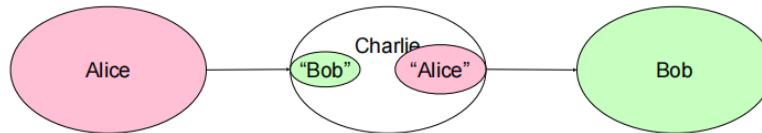
- Passive attacks:
 - Attacker can only read packets ("sniffing")
 - Extremely easy on wireless
 - Relatively easy on shared media such as Ethernet
 - Can only really be excluded by quantum cryptography
- Active attacks:
 - Attacker also injects new packets into the network
 - Source address can be spoofed
 - * Egress/ingress filtering can make this harder
 - Bind attacks: can only write, not read
 - Replay attacks: inject copy of previous packet ("launch rocket now")

7.7.2 Combinations

- Passive followed by active attack:
 - Password sniffing (passive) + login using sniffed password (active)
 - Can be supported by an offline attack, e.g. dictionary attack
 - * If sniffed information can be used offline to determine whether guessed password is correct
- Active attack to facilitate passive attack:
 - Subvert forwarding/routing system to divert traffic via attacker
 - Quite easy at layer 2 (tools: dsniff, ettercap)
 - Subverting routing at layer 3 may be harder
 - Compromised router/switch can be used as tool

7.8 Man-in-the-middle-Angriff = MITM = Janus-Angriff

- Der Man-in-the-middle-Angreifer täuscht jedem Kommunikationspartner vor, der andere zu sein:
 - Nachrichten können auf dem Weg beliebig abgehört oder verfälscht werden



- Abhilfe: Kryptographie (Authentisierung/Verschlüsselung)

7.8.1 On-path vs. off-path attacks

- On-path attacker can easily eavesdrop, spoof, suppress, inject
- Off-path attacker is typically limited to blind attacks
 - Unless topology can be subverted to convert off-path into on-path situation
- Many protocols protect well against off-path attackers, not so well against on-path
 - E.g. TCP random sequence numbers are worthless if overheard by on-path attackers
- (Note that real internet paths are often asymmetric)

7.9 DoS-Angriffe

- Außer Gefecht setzen (Absturz, fehlerhafter Zustand)
- Überwältigen (viel Verkehr)
 - Erleichtert durch Zombies/Botnets (DDoS)
- Amplifier-Angriffe: z.B. Smurf
 - Paket mit gefälschter Absenderadresse (Opfer) an "directed broadcast"
 - Alle Zielsysteme schicken ICMP-Antwort "nicht erreichbar" an Opfer
 - Directed broadcast aus Routern heute praktisch verschwunden
 - Nebeneffekt: ICMP gilt bei vielen Admins als bösartig (vgl. auch "ping of death")
 - * Probleme mit Path MTU discovery

7.10 SQL-Slammer/Sapphire

- 25. Januar 2003: Korea praktisch offline
- UDP-Paket mit 376 Bytes Nutzlast
 - Wird mit maximaler Geschwindigkeit an zufällig generierte IP-Adressen versendet
- Warhol-Wurm: infizierte die meisten der $\geq 75,000$ Opfer innerhalb von 10 Minuten
 - Verdopplung alle 8.5 Sekunden
 - Trotz Bug in PRNG
- Basis: Verwundbarkeit in MSSQL-Server
 - seit 24.07.2002 bekannt
- Abhilfe: Port 1434 (MSSQL) schließen

7.11 Internet Background Radiation

- Würmer wie SQL-Slammer sind immer aktiv
- "Hintergrundstrahlung": Ca. 0.15-2Bytes/s/IP-Adresse
- Ungepatchtes Windows-System ans Internet anschließen?
 - Infektionen innerhalb von Minuten (Sekunden?)

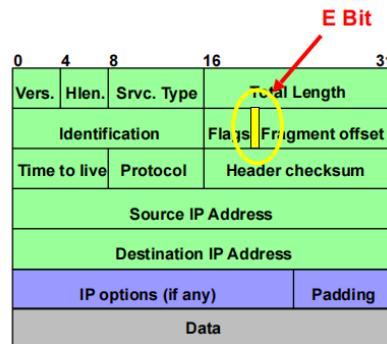
- Kompletter Absturz nach c.a. 30 Minuten
- + Absichtliche Angriffe
- Unternehmensnetze brauchen nicht unbedingt volle Internet-Konnektivität
 - Firewalls

7.12 Geschützte Werkstätten (Unternehmensnetze und Firewalls)

”The primary purpose of **firewalls** has always been to **shield buggy code** from **bad guys**“ - Steve Ballovin, IETF Security AD

7.12.1 Classifying Traffic: The E(vil)-Bit

- Key question: how to identify malicious or other unwanted traffic?
- Potentially intense processing required per packet
 - Source + destination IP addresses and port numbers protocol type
 - Stateful packet inspection even more expensive

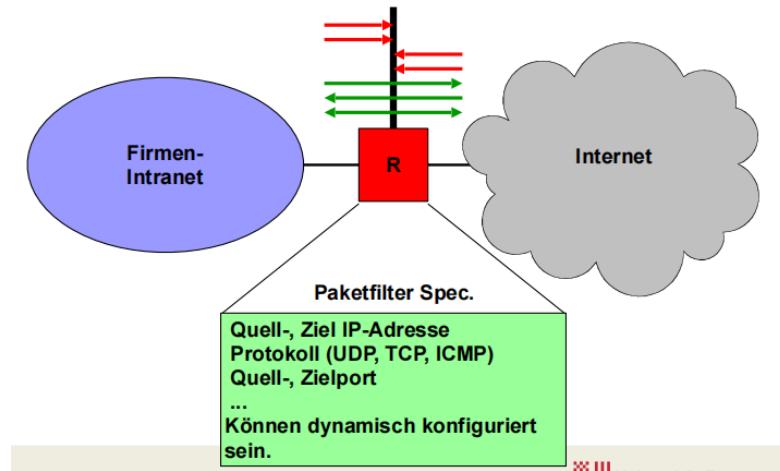


- Solution: RFC 3514 (1 April 2003)
 - ”The Security Bit in the IPv4 Header“
 - Straightforward traffic identification
 - Fail-safe, easy to implement
 - E == 1: packet has evil content
 - E == 0: packet is ok
 - Firewalls simply discard evil packets
 - Extension for IPv6: ”evil strength“

7.12.2 Firewall

- Perimeterschutz: "Crunchy outside, mushy inside"
- Unternehmensnetze und Internet werden nicht direkt verbunden, sondern über **Firewall**
- Nur "erwünschte Kommunikation" wird durchgelassen
 - Problem: wie identifiziert man diese?
- Interne Systeme werden von Angreifern von außen geschützt
 - Sinnvoll: auch umgekehrt
- Gefahr: trügerische Sicherheit
 - Schützt nicht gegen insider!

7.12.3 Paketfilter

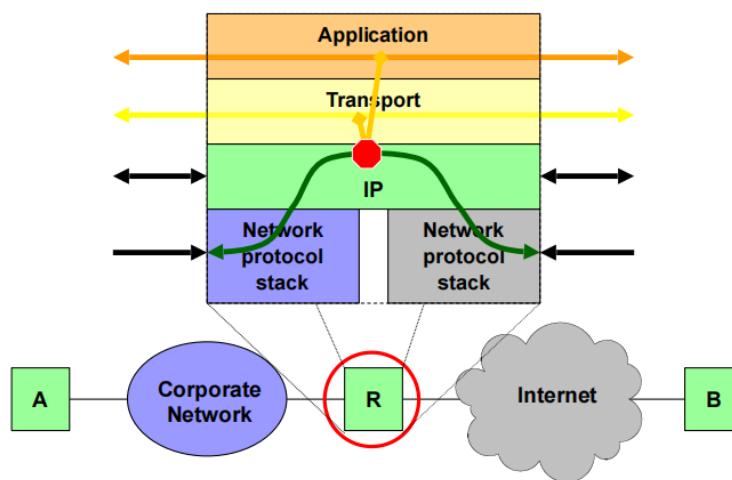


Kriterien für Paketfilter

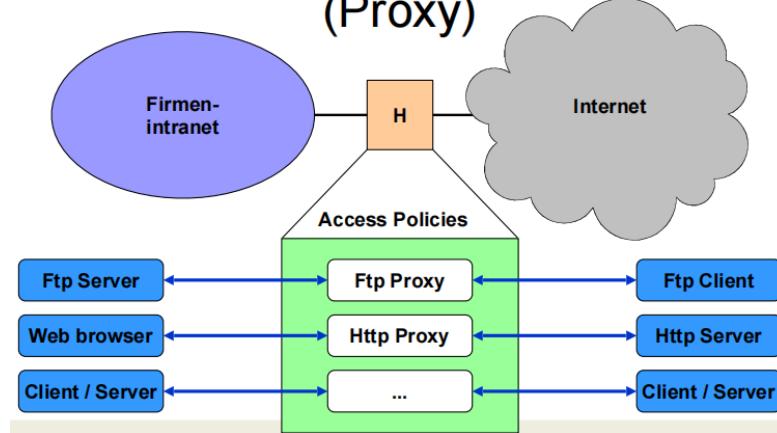
- Keine "merkwürdigen" Pakete, Protokollverletzungen
 - Keine IP-Optionen wie *loose source routing*
 - Evtl. auch Fragmente ausschließen (oder reassemblieren!)
 - Keine Martians/Bogons/Pakete von außen mit innerer IP-Src
- 5-Tupel (IP-Src, IP-Dst, IP-Protokoll, Src-Port, Dst-Port)
 - access-list 115 deny udp any any eq 1434 (SQL-Slammer)

- Besser: explizit nur gewünschte Kommunikation zulassen
- Problem: IPsec verdeckt Port-Nummern (nur noch IP-Src/IP-Dst)
- TCP: Aufbauversuche (SYN ohne ACK) evtl. nur in eine Richtung zu lassen
 - access-list 120 permit tcp any any established

7.12.4 Stateful Packet Inspection

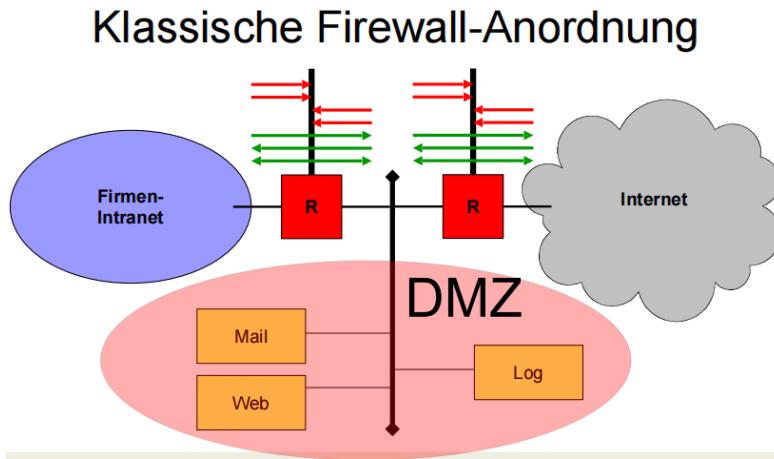


Application Layer Gateway (Proxy)



7.12.5 Aufgaben eines ALG

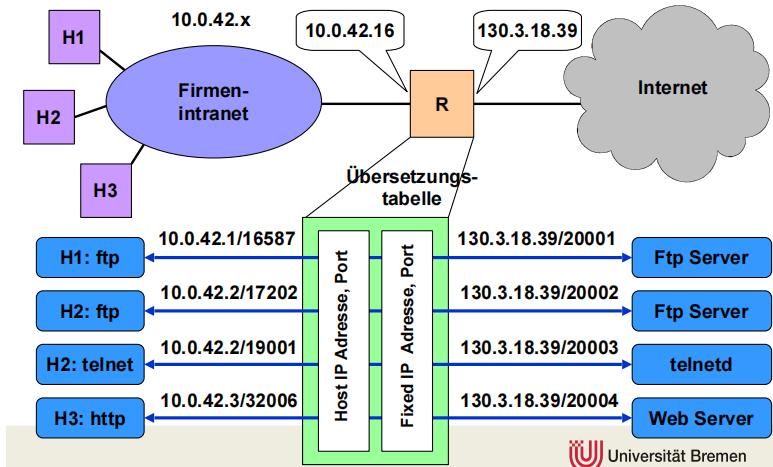
- Kann "tief" in Anwendungsflüsse hinsehen
- Protokollverletzungen nicht durchlassen
- Anwendungsregeln realisieren
 - Kein Zugriff auf bestimmte Websites
 - Kein "unerwünschter Inhalt" (Viren/Trojanische Pferde)
 - Bekannte Angriffe abwehren (Buffer Overflows etc...)
- Anwendungsprotokolle terminieren
 - Z.B. Mail von außen entgegennehmen, dann intern ausliefern (und umgekehrt)
- SPF (stateful packet filter): ALG für Arme



7.12.6 DMZ (Grenznetz)

- Von außen sichtbare Dienste
- Proxies/ALGs
 - "Bastion Hosts"
- Paket-Filter nach außen schützt DMZ
- Paket-Filter nach innen schützt internes Netz
 - Kombination lässt kein Paket durch

Network Address Translator (NAT)



7.12.7 NAT: kein Firewall

- NAT löst Adressraumprobleme
- Nebeneffekt: verhindert kommende Verbindungen
 - Krude Firewall-Funktionalität
 - Kommerzielle Firewalls meist um Paketfilter erweitert (5-Tupel)
- Keine weiteren Analyse-/Logging-Funktionen
- IPv6 (hat keine NATs): RFC 6092 schlägt äquivalente Funktion vor ("Recommended **Simple Security** Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service")

7.12.8 Logging

- Was ist passiert?
- Logfile: Meldungen mitschreiben
 - Ereignisse rekonstruieren
 - Ereignisse erkennen
- Logfile vor Angreifern schützen!

7.12.9 Der Einfluss von Firewalls auf das Internet

- Neue Dienste lassen sich nur Firewall-Upgrade ausrollen
- Hindernisse
 - Neue Software-Versionen erforderlich
 - Schulung der Admins: Fortschreiben der Betriebsregeln, ...
- Ergebnis: Viele Protokolle heute benutzen HTTP, "um durch die Firewalls zu kommen"
 - "Web-Services"
 - SSH over HTTP, IP over HTTP!
- Firewalls müssen nun eine Stufe höher einsetzen

7.12.10 "firewall friendly"

- Neue Protokolle müssen "Firewall-freundlich" sein
- Oberflächlich: Alles über HTTP
- Wirklich: Flüsse müssen selbstbeschreibend sein
 - Politiken müssen allein auf den Flüssen realisiert werden können
 - Trennung Steuerung/Datenflüsse erschwert (vgl. SIP)
- Fallback: Middlebox-Steuerung
 - Z.B. via UPnP, besser: PCP (war: NAT-PMP)
- Fallback: HTTP-Proxy authentication, SOCKS
 - Authentisierter Zugriff auf Proxy

Chapter 8

Sicherheitsprotokolle (Theorie)

8.1 Sicherheitsprotokolle

- Wir haben bisher verschiedene kryptographische Grundbausteine kennen gelernt wie z.B.:
 - Symmetrische und asymmetrische Verschlüsselung
 - Hash, MACs, Digitale Signaturen
 - Zufallswerte, Nonces, Zeitstempel
- Sicherheitsprotokolle (**security protocols, cryptographic protocols**)
 - Basieren auf diesen Grundbaustein
 - Verschiedene Arten von Protokollen:
 - * Protokolle zur Authentisierung (z.B. Challenge Response-Protokolle, Kerberos, Needham-Schroeder)
 - * Schlüsselvereinbarungsprotokolle (z.B. IKE, Diffie-Hellman)
 - * Zusammengesetzt zu E-Commerce-Protokollen (SSL/TLS), E-Government (OSCI)

8.2 Grundlegende Notation

- Ein Protokoll besteht aus einer Menge von Regeln, die den Zustand des Nachrichtenaustauschs zwischen zwei oder mehr Kommunikationsteilnehmern beschreiben:
 - Kommunikationsteilnehmer: Benutzer, Prozesse, Rechner
 - auch "Principals" genannt (siehe Kapitel 2)

- Protokollschrifte

$$n. A \rightarrow B : M$$

- ”**A** sendet **M** an **B** gemäß dem n-ten Protokollschrift.” A,B Principals, M Nachricht
- Nachrichten können strukturiert sein: $M = M_1, \dots, M_n$

- Ein **Protokoll** ist eine Folge von Protokollschriften:

$$\begin{aligned} 1. A_1 &\rightarrow B_1 : M \\ 2. A_2 &\rightarrow B_2 : M \end{aligned}$$

- Protokolldurchläufe
- Berücksichtigung einer feindlichen Umgebung: Fehler, Verlust von Vertraulichkeit/Datenintegrität
 - Angreifer beherrscht Kommunikationskanal, kann aber nicht die grundeliegenden kryptographischen Verfahren brechen (Dolev-Yao-Modell, 1981)

8.3 Challenge-Response-Protokolle

- Einfaches Protokoll: Passwort

$$1. A \rightarrow B : K_{AB}$$

Problem: u.a. Mitschneiden der Passwörter

- Verbesserung: Challenge-Response-Protokoll
 - **Ziel:** B authentisiert sich gegenüber A

$$1. A \rightarrow B : N$$

- A schickt B eine Herausforderung (**challenge**) **N** (nonce)

$$B \rightarrow A : \{N\}_{K_{AB}}$$

- B beantwortet die Challenge mit $\{N\}_{K_{AB}}$ (**response**)

- Mutual Challenge-Response:

$$1. A \rightarrow B : N_A$$

$$2. B \rightarrow A : \{N_A, N_B\}_{K_{AB}}$$

$$3. A \rightarrow B : N_B$$

Bemerkung: Dieses einfache Protokoll ist angreifbar

- Anwendung von Challenge-Response-Protokollen:
 - GSM/GPRS/UMTS
 - PPP (Authentisierungsprotokoll CHAP)
 - Kerberos

8.3.1 Angriff auf das Mutual Challenge-Response-Protokoll

- **Idee:**

- Angreifer B' fängt alle Nachrichten zwischen A und B ab und gibt sich als B aus (vgl. Dolev-Yao-Modell eines Angreifers)
- Angreifer B' startet nun parallel zum Authentisierungsvorgang zwischen A und B eine Session
- Angreifer B' bringt A dazu, die eigene Nonce zu verschlüsseln

Die einzelnen Schritte des Angriffs

$$1. A \rightarrow B' : N_A$$

$$1'. B' \rightarrow A : N_A$$

Angreifer spielt die Nachricht einfach zurück (**reflection attack**)

$$2'. A \rightarrow B' : \{N_A, N'_A\}_{K_{AB}}$$

A meint, dass B einen neuen Durchgang des Protokolls gestartet hat, und antwortet

$$2. B' \rightarrow A : \{N_A, N_B = N'_A\}_{K_{AB}}$$

B' spielt auch **Nachricht 2'** zurück

$$3. A \rightarrow B' : N_B$$

A meint, es handelte sich nun um die Antwort von B aus dem ersten Protokoll-durchgang, und antwortet.

B' authentisiert sich also gegenüber A als B, obwohl B' den Schlüssel K_{AB} nicht kennt!

8.4 Einschub: Reflection-Angriff

- Typisches Angriffsmuster:
Reflection-Angriff:

- Der Angreifer spielt Nachrichten **zurück**
- Systeme zur Freund-Feind-Erkennung (**identity friend-or-foe systems, IFF**) waren angreifbar durch Reflection-Angriffe
- Reflection-Angriffe sind häufig verbunden mit der Vermischung verschiedener Protokollläufe

8.4.1 Lösung des Problems

- Lösungsmöglichkeiten:
 - 1. Für beide Richtungen verschiedene Schlüssel verwenden (K_{AB} bzw. K_{BA})
 - 2. Den Namen des Absenders in Schritt 2 mit angeben, also:

$$B \rightarrow A : \{B, N_A, N_B\}_{K_{AB}}$$

- Wichtige Designregeln für Protokolle
 - Nicht denselben Schlüssel für verschiedene Zwecke verwenden
 - Soviel explizite Informationen wie möglich verwenden (hier den Namen explizit erwähnen)

8.5 Das Denning-Sacco-Protokoll

- D.E Denning und G.M. Sacco, 1982
- Schlüsselaustauschprotokoll mittels Public-Key-Verfahren und Zertifikaten
- **Ziel:**
 - Alice besitzt nach Durchführung der Protokollschrifte einen gemeinsamen symmetrischen Schlüssel K_{AB} mit Bob
 - Dieser Schlüssel soll frisch, also keine Wiedereinspielung (**replay**) sein
- **Voraussetzung:** Es existiert eine vertrauenswürdige Certification Authority S:
 - Zertifikate CA und CB für Alice und Bob
 - Diese Zertifikate binden den öffentlichen Schlüssel für die Verschlüsselung und den Verifikationsschlüssel für digitale Signaturen an den Namen des jeweiligen Besitzers

8.5.1 Die einzelnen Schritte des Denning-Sacco-Protokolls

Bezeichnungen: K_x^{-1} bezeichne den Signaturschlüssel von X, K_x den Public-Key von X.

$$1. A \rightarrow S : A, B$$

$$2. S \rightarrow A : CA, CB$$

A erhält die entsprechenden Zertifikate von S.

$$3. A \rightarrow B : CA, CB, \{T_A, K_{AB}, \{T_A, K_{AB}\}_{K_A^{-1}}\}_{K_B}$$

- Vertraulichkeit durch Verschlüsselung
- Authentisierung durch Zertifikat und digitale Signatur
- Replay-Angriffe werden durch Zeitstempel verhindert

8.5.2 Der Angriff

- Bob kann sich gegenüber anderen Kommunikationspartnern als Alice ausgeben:
 - 1. Bob besorgt sich ein Zertifikat für Charlie,
 - 2. entschlüsselt die Nachricht 3,
 - 3. schneidet die Signatur aus und
 - 4. erzeugt eine Nachricht an Charlie:

$$B \rightarrow C : CA, CC, \{T_A, K_{AB}, \{T_A, K_{AB}\}_{K_A^{-1}}\}_{K_C}$$

- Charlie meint wegen der digitalen Signatur von Alice, dass die Nachricht von Alice stammt
- Der Fehler wurde erst 1994 von Abadi aufgedeckt!

8.5.3 Problembehebung

- Lösung:
 - Einfügen des Namens von Bob in dem signierten Teil der Nachricht, also:

$$\{T_A, B, K_{AB}\}_{K_A^{-1}}$$
 - Regel: soviel explizite Informationen wie möglich verwenden (hier den Namen explizit erwähnen)

8.6 Zusammenfassung

- Sicherheitsmechanismen werden oft über die Protokolle und nicht über unzureichende kryptographische Algorithmen gebrochen
- Scheinbar einfachste Protokolle können subtile Sicherheitslücken besitzen
- Dieses Problem ist immer noch aktuell, da viele neue bzw. spezielle Protokolle entworfen werden:
 - Nicht nur Authentisierungs- und Schlüsselaustausch-Protokolle, sondern auch:
 - * E-Purse-Protokolle (z.B. bei der Geldkarte), E-Commerce-Protokolle

- * OSCI-Protokoll (Ist dieses Protokoll überhaupt schon einmal von Sicherheitsexperten analysiert worden?)
- Lösungsmöglichkeiten:
 - * Sich an Designregeln halten
 - * Einsatz formaler Methoden

8.7 Protokollanalyse mit Formalen Methoden

- Verschiedene Ansätze wie z.B.:
 - Modellprüfer (**model checker**):
 - * Protokolle werden als Finite State Matchings (FSM) wie z.B. Kripke-Strukturen modelliert
 - * Untersuchung aller Ausführungspfade in der FSM
 - * Aber: State Explosion Problem
 - * Werkzeuge für die Verifikation von Security-Protokollen: Casper-Werkzeug von Gavin Lowe, CAPSL, SPIN
 - Theorembeweisen
 - * Induktive Methode von Paulson
 - * Spezielle Security-Logiken: BAN-Logik(Burrows, Abadi, Needham)
 - Kein Allheilmittel, oft unzuverlässige Abstraktion
 - Lars Knudsen: **If it's proveably secure, it probably isn't!**

Chapter 9

Sicherheitsprotokoll: TLS (SSL)

9.1 Transport Layer Security: SSL/TLS

- Entwickelt von Netscape für "E-Commerce" im Web
 - Online-Shops: Schutz von persönlichen Daten, Kreditkartennummern etc.
 - "HTTPS" = HTTP über TLS (RFC 2818)
- Secure Sockets Layer (SSL)
 - SSL 2.0 war unsicher
 - letzte Version unter dem Namen SSL 3.0 (in 2014 auch hier Lücken gefunden)
- Transport Layer Security (TLS, RFC 2246+3546/4346+4366/5246→8446): leicht abgewandelter Nachfolger von SSL (= "SSL 3.1")
 - Allgemein verwendbar, nicht nur HTTPS (z.B. POP3, IMAP, SMTP)
 - DTLS: RFC 6347

9.2 SSL/TLS

9.2.1 Grundlegende Idee

- Hauptziele:
 - **Vertraulichkeit** (Secrecy) zwischen dem Kunden (Web-Browser) und einem Web-Server
 - **Authentisierung** (des Web-Servers)

- **Integrität** der übertragenen Daten
- Authentisierung des Clients laut Spezifikation auch möglich, wird aber selten genutzt (warum?)
- SSL/TLS nutzt hybrides Verschlüsselungsverfahren:
 - Schlüsselvereinbarung mit Public-Key-Verfahren
 - * Öffentlicher Schlüssel des Web-Servers wird dem Web-Browser per X.509-Zertifikat bekannt gemacht
 - Anschließend: symmetrisches Verfahren zum Datenaustausch

9.2.2 Teilprotokolle

- SSL/TLS Teilprotokolle:
 - Handshake-Protokoll (Schlüsselvereinbarung, Aushandlung der Parameter für die Verschlüsselung, MAC-Bildung, ...)
 - Record-Protokoll (verschlüsselte Daten, Bildung von MACs)
 - Alert-Protokoll (aufgetretene Fehler während des Protokollverlaufes)
- Wir betrachten das Handshake-Protokoll in zwei Verfahren
 - 1. Auf RSA basierender Schlüsselaustausch
 - 2. Diffie-Hellman-Schlüsselvereinbarung

9.2.3 SSL/TLS-Handshake mit RSA: Vorausgesetzte Notationen und Bezeichnungen

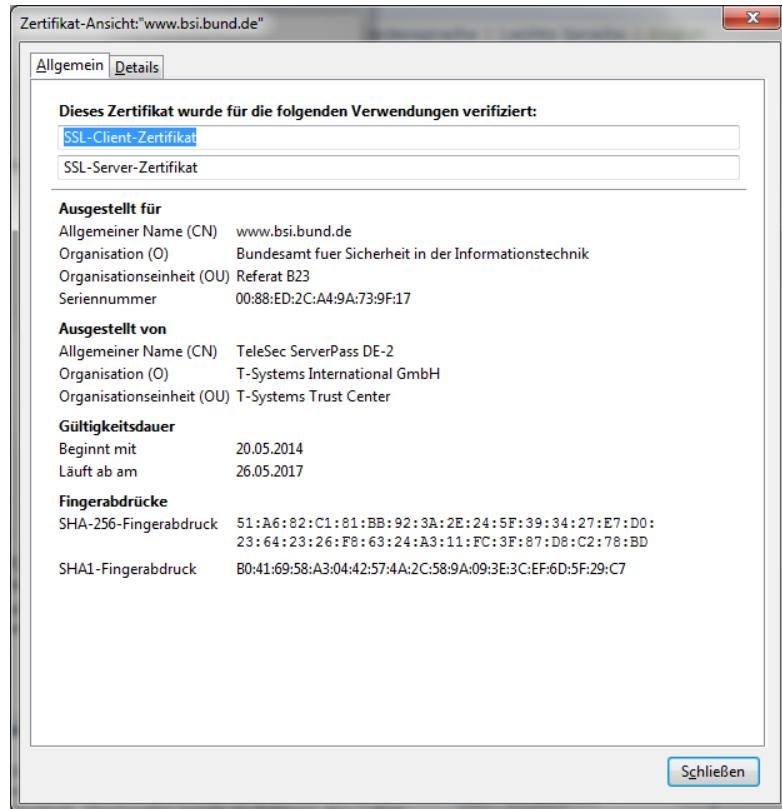
- Kommunikationspartner C (**client**) und S (**server**)
- CS: Serverzertifikat

$$CS = Cert_{CA}(S, K_S, T, L)_{K_{CA}^{-1}}$$

- K_S öffentlicher Schlüssel des Servers
- K_{CA}^{-1} privater Schlüssel der CA
- **Vorbemerkung:** Es wird im Folgenden eine vereinfachte Fassung des Handshake=Protokolls vorgestellt
- Session-ID lassen wir weg, ebenso Details zu Zufallszahlen **Random_X**

Serverzertifikat

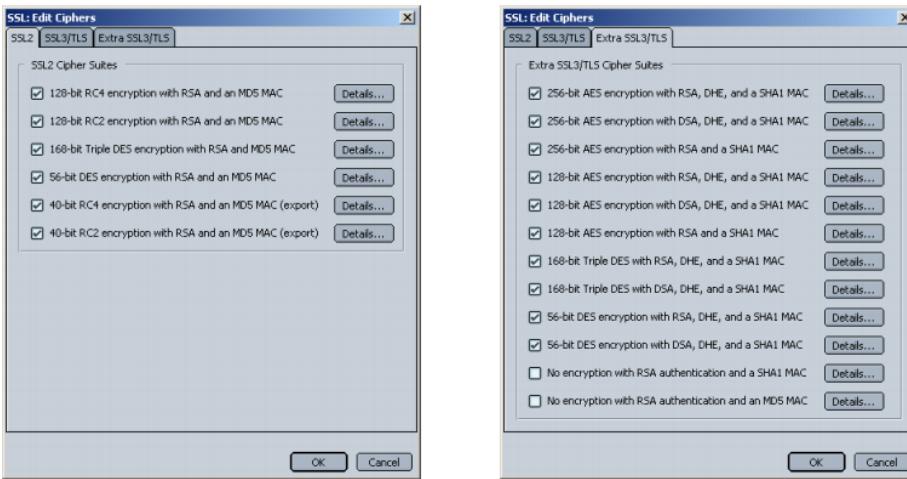
- Bindet den **Public Key** des Servers an den **DNS-Namen** des Servers



9.2.4 SSL/TLS: Handshake mit RSA

- 1. C → S: Random_C , Supported_Ciphers ClientHello
 - Supported_Ciphers sind die Cipher-Suites, die der Client unterstützt
 - Können im Mozilla-Browser ausgewählt werden
 - Bsp.: TLS_RSA_WITH_AES_128_CBC_SHA256
 - Random_C wird später bei der Schlüsselerzeugung benötigt: verhindert **Replay-Angriffe** (Wiedereinspielen von alten Nachrichten)

Kryptographische Suiten



Crypto Agility: Ciphersuites

- **Cipher suite** [|swit|]
Sehr gut zusammenpassender kryptographischer Algorithmen
- **Cipher suit** [|sut|]: "Iron Man" - Anzug

9.2.5 SSL/TLS: Handshake mit RSA

- **2. S → C: Random_S, CS, Chosen_Cipher** ServerHello Certificate
 - **Chosen_Cipher:**
 - * Auswahl aus **Supported_Ciphers**
 - * Auswahl des vom Client präferierten Verfahrens aus der Liste (aber nicht vorgeschrieben)
 - **CS** (Server-Zertifikat): Zur vertrauenswürdigen Verteilung des öffentlichen Server-Schlüssels K_S (RSA-Key)
 - **Random_S**: wie **Random_C** zusätzlicher Schutz vor Replay-Angriffen
- **3. C → S: { K_0 } K_S** (vgl. hybride Verschlüsselung)
 K_0 - **Premaster Secret**: 48 Bytes lang
 - Berechnung eines Master Secrets K_1 aus K_0 (durch beide Kommunikationspartner unabhängig voneinander)

$$K_1 = PRF(K_0, \text{"master_secret"}, Random_c + Random_S),$$

”+” bedeutet hier Konkatenation
PRF - Pseudo Random Function zur Erzeugung von diversen Schlüsseln

- \mathbf{Random}_C und \mathbf{Random}_S verhindern Replay-Angriffe
- **S** und **C** können dann aus dem Master Secret K_1 das eigentliche Schlüsselmaterial berechnen
 - * Wieder unter Verwendung von PRF

9.2.6 Einschub: PRF

- In TLS, in SSL 3.0 ähnliche Funktion
- Basiert auf HMAC unter Verwendung von verschiedenen Hashfunktionen
 - Nun keine Festlegung mehr auf SHA-1 und MD5 (unsicher), Verwendung vom sicheren SHA-256 möglich
 - Verkettung von Hashfunktionen
- Berechnet verschiedenes Schlüsselmaterial
Berechnung eines Schlüssels pro Kommunikationsrichtung:
 - für die symmetrische Verschlüsselung (K_{CS}, K_{SC})
 - für MACs
 - für IVs (z.B. für CBC-Mode)
- PRF liefert diese sechs Schlüssel in einem Schlüsselblock zurück
- PRF(secret,label,seed)

Beispelaufrufe:

$$\text{keyblock} = \text{PRF}(K_1, \text{"key_expansion"}, \text{Random}_S + \text{Random}_C)$$

$$K_1 = \text{PRF}(K_0, \text{"master_secret"}, \text{Random}_C + \text{Random}_S)$$

$$\text{P_hash}(\text{secret}, \text{seed}) = \text{HMAC_hash}(\text{secret}, A(1) + \text{seed}) +$$

$$\text{HMAC_hash}(\text{secret}, A(2) + \text{seed}) +$$

$$\text{HMAC_hash}(\text{secret}, A(3) + \text{seed}) + \dots$$

$$\text{PRF}(\text{secret}, \text{label}, \text{seed}) = \text{P_} < \text{hash} > (\text{secret}, \text{label} + \text{seed})$$

- **Beispiel:**

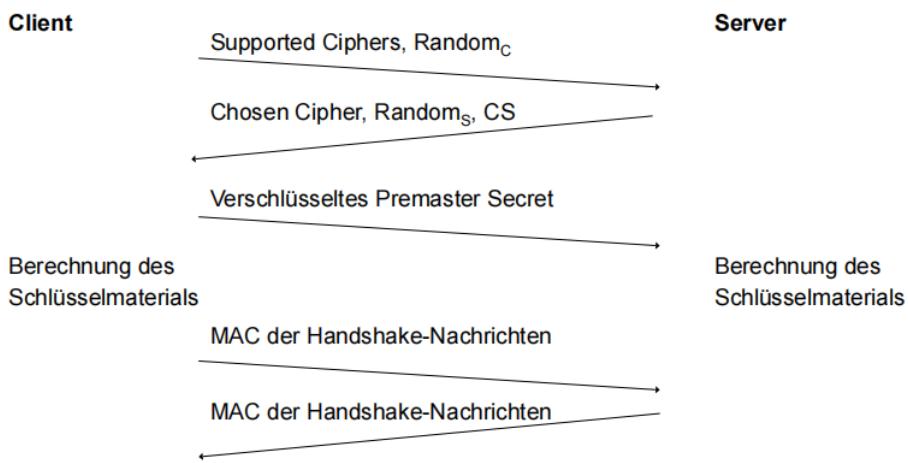
$$\text{keyblock} = \text{PRF}(K_1, \text{"key_expansion"}, \text{Random})$$

$$= \text{P_SHA-256}(K_1, \text{"key_expansion"} + \text{Random})$$

9.2.7 SSL/TLS mit RSA: Handshake (Part 2)

- 4. $C \rightarrow S: \{finished, MAC(K_1, \text{alle bisher gesendeten Nachrichten})\}_{K_{CS}}$
 - **finished** bestätigt das Ende des Handshake-Vorganges des Clients
- 5. $S \rightarrow C: \{finished, MAC(K_1, \text{alle bisher gesendeten Nachrichten})\}_{K_{SC}}$
 - **finished** bestätigt das Ende des Handshake-Vorganges des Servers
 - Warum Verschlüsselung der Nachrichten 4. und 5.?
 - Erst **nach** dieser Nachricht können verschlüsselte und durch einen MAC gesicherte Daten gesendet werden wie z.B. Kreditkartennummern (Record-Protokoll)

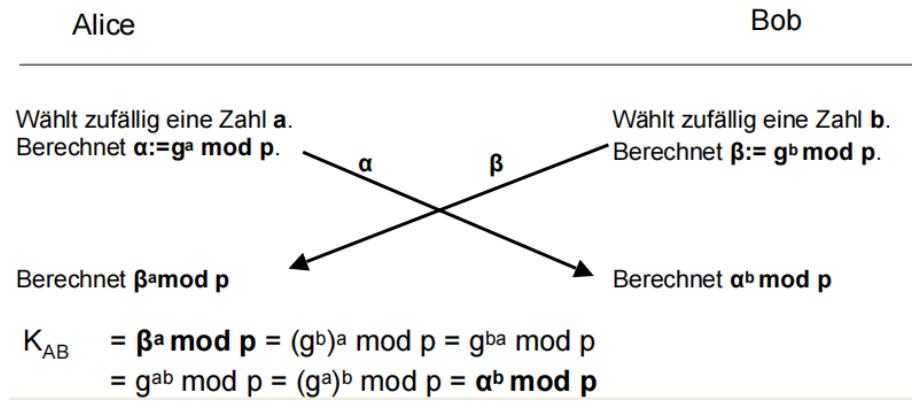
9.2.8 SSL/TLS-Handshake: Überblick



9.2.9 Ergänzung: SSL/TLS-Handshake mit DH

- Schlüsselvereinbarung nach Diffie und Hellman
- Verwendung von DSA (ECDSA) zum digitalen Signieren
- Ansonsten: ähnliche Voraussetzungen wie bei der RSA-Variante

Wiederholung: Schlüsselvereinbarung nach Diffie und Hellman



9.2.10 SSL/TLS: Handshake mit DH

- 1. C → S: **Random_C, Supported_Ciphers** ClientHello
 - Wie bei der RSA-Variante
- 2. S → C: **Random_S, CS, Chosen_Ciphers, {p, g, α}_{K_S^{-1}}** ServerHello
Certificate ServerKeyExchange
 - Ähnlich wie bei der RSA-Variante
 - **Aber:** CS enthält jetzt den Public Key von S zur Signaturverifikation (DSA)
 - Mit DSA digital signierter DH-Public Key: {p, g, α}_{K_S^{-1}}
- 3. C → S: β ClientKeyExchange
 $\beta^a \text{ mod } p = \alpha^b \text{ mod } p = K_0$ **Premaster secret**
 Aus K_0 kann das Master Secret berechnet werden (wie im RSA-Fall):
 - $K_1 = PRF(K_0, \text{"master_secret"}, Random_C + Random_S)$
 - ...
- 4. C → S: {finished, MAC(K_1 , alle bisher gesendeten Nachrichten)}_{K_CS}
- 5. S → C: {finished, MAC(K_1 , alle bisher gesendeten Nachrichten)}_{K_SC}
 - Erst **nach** dieser Nachricht können verschlüsselte und durch einen MAC gesicherte **Daten** gesendet werden, wie z.B. Kreditkartennummern (Record-Protokoll)

9.2.11 Record-Protokoll

- Nach erfolgreichem Handshake können die Daten verschlüsselt und durch einen MAC gesichert gesendet werden
 - z.B. Kreditkartennummern
 - Schutzziele: Vertraulichkeit (Secrecy), Datenintegrität
- Aufspaltung der Nachricht in Blöcke
- Unverschlüsselte Header-Informationen pro Block
- Bei Blockschiffen eventuell Padding erforderlich
- Verschlüsselung und Sicherung durch MAC pro Block

$$C \rightarrow S : \{data, MAC(K_{CS-MAC}, data)\}_{K_{CS}}$$

9.2.12 Alert-Protokoll

- Ziel: den Kommunikationspartner über Ausnahmebedingungen informieren, z.B.:
 - Handshake failure
 - Close notify
 - Unknown CA
 - Certificate expired
 - Certificate unknown
 - No certificate
- Unterscheidung zwischen Warn- und Fehlermeldungen
 - Fehlermeldungen führen immer zum Verbindungsabbruch

9.2.13 RFC 8446: TLS 1.3

- Ziele:
 - Höhere Sicherheit durch heftiges Aufräumen
 - Privacy-Ziele angehen
 - Performance steigern (0-RTT)
- Ciphersuites entbündeln
 - Ciphersuite beschreibt nur noch Record Protocol
 - * Nur noch AEAD
 - Key Exchange (Diffie-Hellman) separat aushandeln
 - * Nur noch PFS
 - Authentication (RSA, ECDSA) separat aushandeln
 - (Pre-Shared-Key separat aushandeln)

Chapter 10

Drahtlose Sicherheit

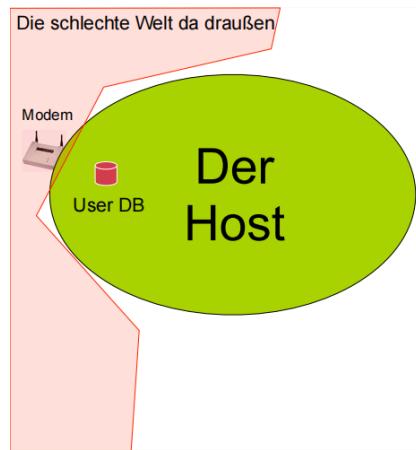
Sicherung des Zugangsnetzes - Network Access Security

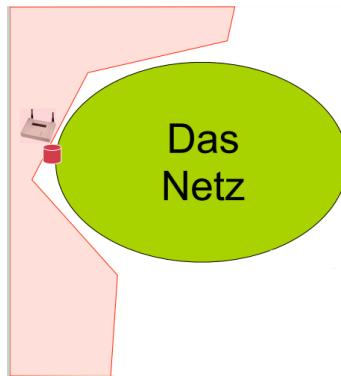
10.1 Überblick

- Traditionelle Sicherung des Zugangsnetzes
- WLAN-Sicherheit
- WLAN Roaming (nicht hier)

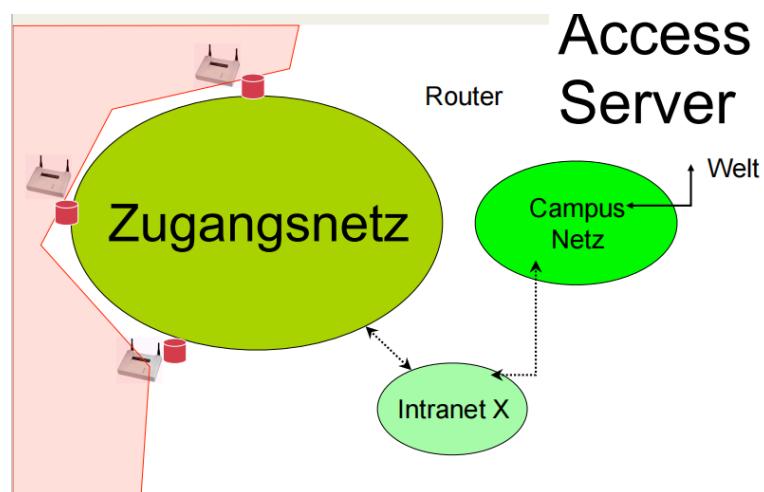
10.2 Traditionelle Sicherung des Zugangsnetzes

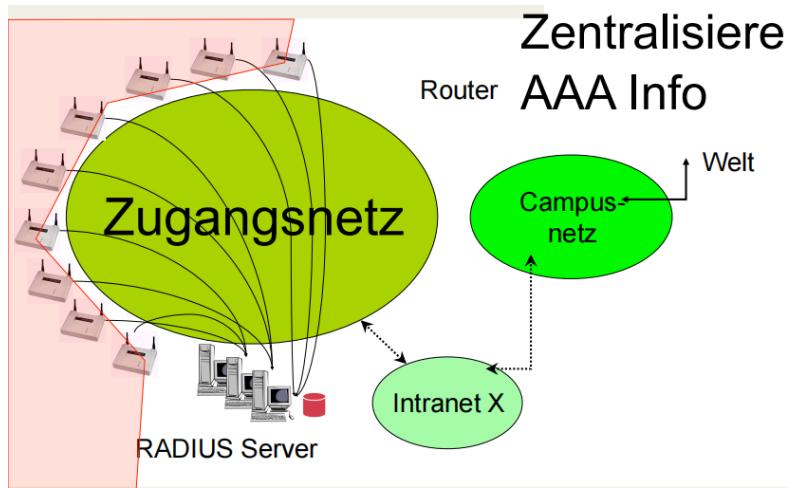
Die Alte Welt



Access Server

- Basisprotokoll: PPP
- Authentisierungsprotokolle:
 - PAP (Passwörter)
 - CHAP (Challenge/Response)
 - EAP (Plugin-fähig)





10.3 Elemente der traditionellen Sicherheit für den Netzzugang

- Behandle die Sicherung des Zugangsnetzes in Schicht 2
 - PPP und die dazugehörigen Authentisierungsprotokolle (PAP, CHAP, **EAP**)
 - Hauptsächlich Benutzer-/Passwort-basiert
- RADIUS als "Backend Protokoll"
 - Access Devices (PEPs, **Policy Enforcement Points**) bleiben "dumm"
 - RADIUS-Server ist der PDP (**Policy Decision Point**)
- NAIs und RADIUS Proxying
 - **Network Access Identifier:** cab0@tzi.de
 - Verwende den Teil hinter @, um den Home RADIUS Server zu identifizieren

10.4 802.1X: Sicherung des Zugangsnetzes im Ethernet

- Vor 802.1X: Jeder konnte sich an einen Switch "hängen" und auf diese Weise Zugang zum Netz erhalten
- 802.1X: EAP over LAN

- Supplicant: Client-Gerät, das versucht, Zugang zum Netz zu erhalten
- Authenticator (PEP): Switch
- Authentication Server (PDP): RADIUS Server
- Switch kann Entscheidungen treffen, die auf Informationen beruhen, die vom RADIUS Server zurückgeliefert werden (Wie z.B. die VLAN-Zuordnung)

10.5 Was ist zu schützen?

- Eher Früher: Knappe Netzressourcen
 - Dial-in Pool, etc.
- Netzsicherheit
 - Häufig war die Sicherung des Zugangsnetzes die einzige Sicherheit für das gesamte Netz!
 - * Heutzutage wegen des Internets keine Option mehr
 - Privilegierte IP-Adressen
 - Zugriff hinter dem Firewall

10.6 WLANs sind anders

- WLANs sind funk-basiert
 - Jeder kann alles hören
 - Vertraulichkeitsanforderungen
- Keine "Leitungen" mehr
 - Unerwünschte Access-Points können Informationen hinzufügen oder manipulieren
- Keine hohen Ressourceanforderungen
 - WLAN ist "schnell"
 - ISM-Band Funk kann sowieso nicht geschützt werden

10.6.1 WLAN-Sicherheit: Anforderungen (Sicht Universität)

- **Vertraulichkeit (Privacy):**
 - Fremder Datenverkehr kann nicht verstanden werden
 - Angriffe von Insidern genauso Wahrscheinlich wie Angriffe von außen

- **Accountability:**

- Möglichkeit, herauszufinden, wer was getan hat
- Voraussetzung: **Authentisierung**

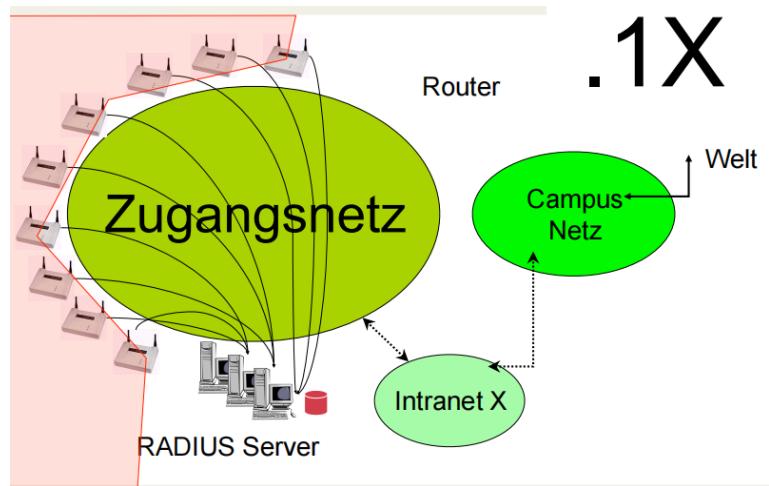
10.6.2 WLAN-Sicherheit: Ansätze

- **AP-basierte Sicherheit:** AP ist die Grenze des Netzes

- WEP (gebrochen), WEP Fixes
- **802.1X** (EAP-Varianten + RADIUS) + 802.11i ("WPA-Enterprise")

- **Netzbasierte Sicherheit:** starke Sicherheit

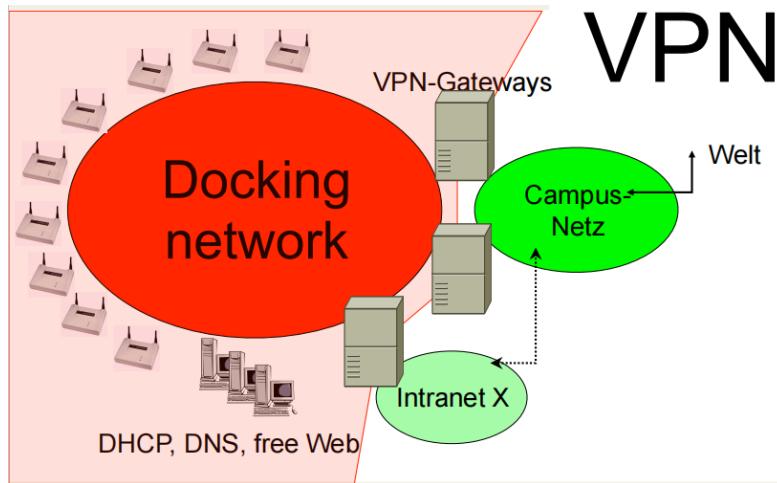
- **VPNs** werden von mobilen Benutzern sowieso verwendet
 - * SSH, PPTP, IPsec
- Alternative: **Web Diverter** (temporäres Filtern von MAC-/IP-Adressen)
 - * Allerdings keine Vertraulichkeit



10.6.3 WLAN-Zugangskontrolle: Warum 802.1X besser ist

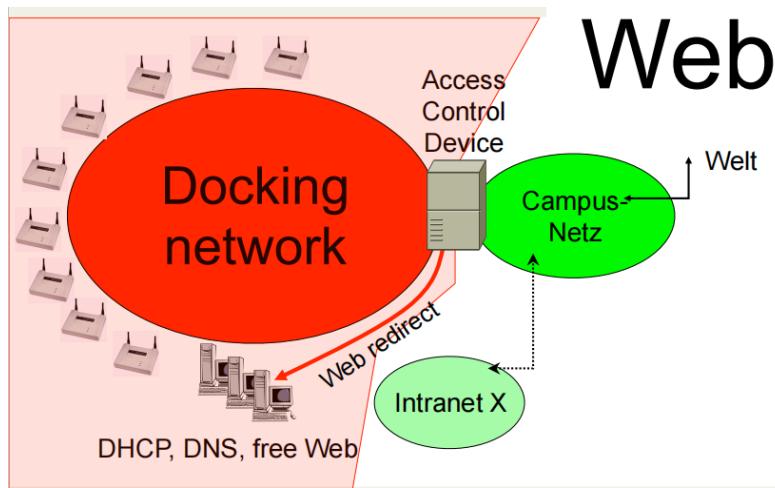
- 802.1X hat sich weitergehend durchgesetzt
- Mehrere EAP/XYZ-Varianten sind inzwischen weit verbreitet
- Wird von immer mehr Systemen unterstützt (Windows 2000 aufwärts)
- Aufwendige Kryptographie wird auf viele APs verteilt

- Angreifer werden so früh wie möglich abgewehrt
 - Roaming: mehr Kontrolle durch Administratoren der Gastgeber-Sites
- Vor allem aber: it just works!



10.6.4 WLAN-Zugangskontrolle: Warum ein VPN besser ist

- Aus historischen Gründen sollte man der Sicherheit in **Schicht 3** mehr als der Sicherheit in Schicht 2 trauen:
 - IPsec ist eingehend auf Sicherheitsprobleme hin analysiert worden
- Man kann billige/”dumme”/nicht vertrauenswürdige APs benutzen
- Auf fast allen Systemen (Windows 98, PDA etc.) **verfügbar**
- Leicht für **mehrere Sicherheitskontakte** anzupassen
 - Sogar mit einer Infrastruktur ”von vor 2003”
 - Die Daten sind nun auf der Luftschnittstelle und bis zum VPN Gateway gesichert
- Aber vor allem: it just works!



10.6.5 WLAN-Zugangskontrolle: Warum Web-basiertes Filtern besser ist

- Keine Software auf dem Client benötigt (jeder hat ja einen Browser)
- Passt gut zu den bereits existierenden Benutzer/Passwort-Schemata
- Kann einfach für **Gastbenutzer** eingerichtet werden
 - Die Hotspots benutzen Web-basiertes Filtern, so dass Gastbenutzer dies schon kennen
 - Passt auch auch gut zu Hotspot-Föderationen
- Privacy ist hier nicht sicherzustellen (verwende TLS und SSH)
- Accountability ist kaum herzustellen
- Aber vor allem: it just works!

10.6.6 Eingebaute Sicherheitsmechanismen

- Service Set Identifier (SSID)
- Wird zur Unterscheidung von APs verwendet
- SSID wird alle paar Sekunden in *Beacon Frames* per Broadcast versendet.
- Beacon Frames werden im Klartext verschickt!
- Erste Sicherheitsschicht
- Stealth Mode - Testanfrage (probe request)

Was man in Bezug auf SSIDs tun und was man nicht tun sollte

- Default SSIDs sind weit verbreitet (Linksys AP Default ist *linksys*, Cisco Default ist *tsunami* etc.): Verwirrung
 - → Default-SSID ändern!
- ? Ändere die Einstellungen im AP, so dass dieser nicht mehr die SSID in einem Beacon Frame per Broadcast versendet
- Warum?

10.6.7 Verbergen der SSID

- Wie bereits erwähnt, wird die SSID standartmäßig alle paar Sekunden per Broadcast versendet
- + Wenn man dieses Feature abschaltet, dann kann man schwerer herausfinden, dass man es überhaupt mit einer drahtlosen Verbindung zu tun hat
- - Lesen von Rohpaketen (raw packets) offenbart die SSID
 - Selbst wenn WEP verwendet wird, ist die SSID im Klartext!
- - Schwieriger in Betrieb zu nehmen
 - Windows scheint hier durcheinander zu kommen

10.6.8 Filtern von MAC-Adressen

- Durch Filtern von MAC-Adressen dürfen sich nur bestimmte Geräte mit einem AP assoziieren
- - Management in großen Netzen undurchführbar
- - Selbst bei Verschlüsselung der Daten kann man mit Hilfe eines Paket-Sniffers sehr leicht eine gültige MAC-Adresse finden und dann über das Betriebssystem die eigene so ändern, dass diese verwendet wird
- → Micky-Maus-”Sicherheit” für sehr kleine Netze, die sich nur von zufälligen Angriffen schützen müssen

10.6.9 Verbinden mit dem AP

- Access Points haben zwei Möglichkeiten, eine Verbindung mit einem Client zu initiieren:
 - Shared Key- oder Open Key-Authentisierung

- **Open Key-Authentisierung** erlaubt jedem Client, eine Verbindung mit dem AP aufzubauen
- **Shared Key-Authentisierung** soll zusätzliche Sicherheit bringen, indem sie Ausweisinformationen (credentials) erfordert, wenn man sich verbindet

Wie die Shared Key-Authentisierung funktioniert

- Challenge Response:
 - Client sendet eine Anforderung für eine Verbindung an den AP
 - AP antwortet mit einer unverschlüsselten Challenge
 - Client verschlüsselt die Challenge mit Hilfe eines korrekten WEP-Schlüssels und sendet diese zum AP zurück
 - Wenn die Challenge korrekt verschlüsselt worden ist, dann erlaubt der AP die Kommunikation mit dem Client

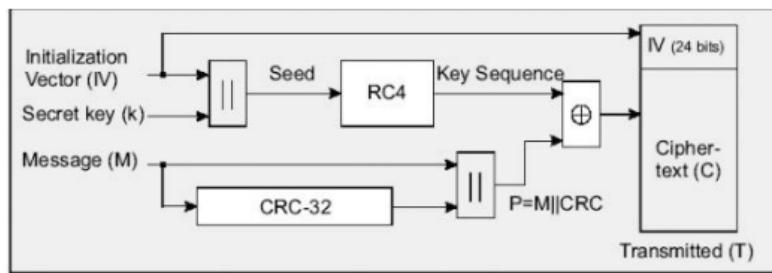
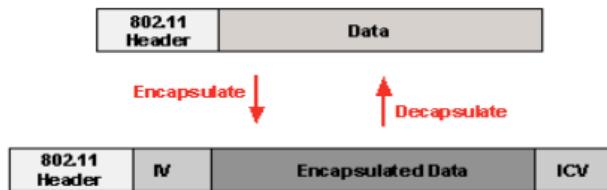
Ist die Open Key- oder Shared Key-Authentisierung sicherer?

- Merkwürdigerweise ist die Open Key-Authentisierung sicherer!
- Mit Hilfe von passivem Sniffen kann man zwei der drei Parameter abfangen, die für die Shared Key-Authentisierung benötigt werden, nämlich: Challenge und die verschlüsselte Challenge
 - Kann man durch einen Dis-Assoziierungsangriff (**disassociation attack**) erhalten

10.6.10 Wired Equivalent Privacy (WEP)

- Ursprüngliche Sicherheit für das 802.11-Protokoll
- Sollte drahtlose Netze genauso sicher machen wie das herkömmliche Netz
- Sollte "Vertaulichkeit", "Integrität" und "Authentisierung" sicherstellen
- Verwendet 40-Bit-RC4-Verschlüsselung
- Leider hat sich diese Art, RC4 zu verwenden, seit der Annahme des 802.11-Standarts als unsicher erwiesen:
 - Das 802.11 Protokoll ist verwundbar gegenüber unterschiedlichsten Arten von Angriffen

WEP-Verschlüsselung



Probleme mit WEP

- Schlüssellänge ist zu kurz (40 Bits!)
 - Brute Force-Angriff ist möglich
- Keine Spezifikation für die Schlüsselverteilung
 - Keine Skalierbarkeit
- → Ein statischer Schlüssel!
 - Keine noch so gute Verschlüsselung ist stark, wenn ein Schlüssel für immer verwendet wird
- Verwenden von CRC32 als ICV (integrity check value)
 - Bit flipping-Angriff:

$$\text{CRC}(\text{msg} \text{ XOR } \delta) = \text{CRC}(\text{msg}) \text{ XOR } \text{CRC}(\delta)$$
 - Bits können zwar nicht gezielt gesetzt oder gelöscht, dafür aber umgedreht werden
- Kein Schutz vor Replay-Angriffen
- Falsche Anwendung von RC4
 - Protokoll spezifiziert nicht, wie IVs verwendet werden sollen
 - Es existieren zwei Angriffe

10.6.11 Angriff auf die Länge des IVs

- IVs sind nur 24 Bits lang und somit gibt es nur 16,777,216 mögliche Werte für die IVs
- Ein Netz mit viel Datenverkehr wird die IVs also oft wiederholen
- Indem der verschlüsselte Verkehr mitgeschnitten wird und Duplikate der IVs eingesammelt werden, kann man den Klartext wiedergewinnen

10.6.12 FMS-Angriff (Weak IV-Angriff)

- Manche IVs sind für RC4 unbrauchbar!
- Wenn man eine Formel nutzt, kann man diese IVs verwenden, um Teile des WEP-Schlüssels abzuleiten
 - 5% Wahrscheinlichkeit, richtig zu raten
- Es gilt wieder: Eine passive Überwachung des Netzes ein paar Stunden lang kann bereits ausreichen, genug schwache IVs einzusammeln, um den WEP-Schlüssel zu ermitteln
- 4M ~ 6M Pakete, um einen 40-Bit-WEP-Schlüssel zu ermitteln
- Die Zeit, die man zur Durchführung des Angriffs benötigt, ist **proportional** zur Schlüssellänge
 - 104-Bit-Schlüssel sind nur 2.6 mal sicherer als 40-Bit-Schlüssel
- Fluhrer, Mantin, Shamir: Weaknesses in the key scheduling algorithm of RC4. In 8th Annual Workshop on Selected Areas of Cryptography, 2001

10.6.13 Zusammenfassung: WEP

- Vertraulichkeit
 - FMS-Angriff
- Integrität
 - Bit-flipping-Angriff
- Authentisierung
 - Nicht wirklich
- WEP ist gebrochen, und es gibt keine einfache Möglichkeit, WEP zu reparieren
- Angriffe auf WEP sind passiv möglich und extrem schwer zu erkennen

WEP NICHT MEHR VERWENDEN!

10.7 Virtual Private Networks (VPN)

- Ein sicheres VPN über ein drahtloses Netz kann die Sicherheit der Daten substantiell erhöhen
- Die Idee ist hier, dass ein drahtloses Netz genauso wie ein unsicheres Netz (das Netz) behandelt wird
- Das "docking network" führt nur zu den VPN-Gateways

10.7.1 Probleme des VPN-Ansatzes

- Erheblicher Aufwand bei der Inbetriebnahme
- Performance skaliert nicht mit der Anzahl der eingerichteten APs
 - PC: Krypto-Geschwindigkeiten liegen bei ungefähr 500 MBit/s, stark parallelisierbar
- Anfällig für DoS-Angriffe
 - Zum Beispiel gegen DHCP/DNS im Docking Netz
- PCs im Docking-Netz sind verwundbar
- Anfällig für Angriffe auf das jeweils gewählte VPN
 - Begründete Annahme: Kann schnell in Ordnung gebracht werden (VPNs haben eine Schnittstelle zum Internet!)
 - (aber PPTP mit MSCHAPv2 ist ziemlich anfällig für Wörterbuch-Angriffe)

Zurück zu Lösungen auf der Schicht 2

- 802.1x
 - Authentisierung für jeden einzelnen Nutzer
 - Mechanismus für die Schlüsselverteilung
- 802.11i "RSN" (Robust Security Network)
 - 802.1x mit EAP + AES + CCM
- WPA
 - Teilmenge von 802.11i
 - Zwei Arten
 - * Enterprise mode: 802.1x mit EAP + TKIP (enthält MIC)
 - * Personal mode: Pre-shared Schlüssel + TKIP (enthält MIC)

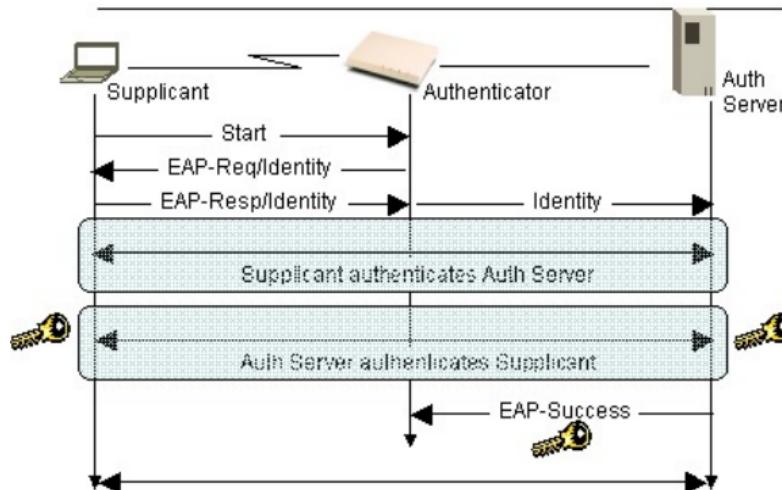
10.8 802.1X-Authentisierung

- 802.1X ist port-basiert, Rahmenwerk zur Authentisierung in IEEE 802-Netzen auf Schicht 2 (MAC-Adressen-Schicht)
- Nicht nur anwendbar in 802.11-Netzen
- Verwendet EAP für die Implementierung
- 802.1X ist keine Alternative zu WEP: Es arbeitet mit dem 802.11-Protokoll zusammen, um die **Authentisierung** von WLAN Clients zu regeln
 - Erzeugt auch temporäre Schlüssel für die Verschlüsselung und zur Sicherstellung der Datenintegrität

Wie die Authentisierung durchgeführt wird

- Ein Client verlangt Zugriff auf den AP
- Der AP fragt nach einer Menge von Ausweisinformationen (**credentials**)
- Der Client schickt die Ausweisinformationen an den AP, der seinerseits die Ausweisinformationen an den Authentisierungsserver weiterleitet
- Das genaue Verfahren, um die Ausweisinformationen zu verteilen, wird in 802.1X selbst nicht festgelegt!
 - Verwendet EAP über LAN (EAPOL)

802.1x-Authentisierung



10.9 Extensible Authentication Protocol (EAP)

- 802.1X verwendet EAP für sein Rahmenwerk zur Authentisierung
- Flexibel: Einmalpasswörter, Zertifikate, Chipkarten, eigene EAP-Protokolle, etc.
- Kein Overhead für ein Paket
- Kosteneffizient
 - 802.1X passt gut zu anderen offenen Standards wie RADIUS
 - RADIUS ist der De-facto Standard für ein Backend-Protokoll zur Authentisierung an einem Network Access Server

10.10 EAP-MD5

- EAP-MD5 ist ein einfaches EAP-Protokoll (ähnlich CHAP)
- Verwendet einen MD5-Hash des Benutzernamens, eine Challenge für den Server und ein Passwort, das zum RADIUS-Server gesendet wird
 - Verwundbar gegen Wörterbuch-Angriffe
- Authentisiert nur in einer Richtung
 - Middle-Person-Attack
- Keine Schlüsselerzeugung

10.11 LEAP (Cisco Wireless)

- Wie MD5 verwendet LEAP ein Login-/Passwort-Schema, welches an den RADIUS-Server gesendet wird
- Jeder Nutzer erhält einen dynamisch erzeugten Einmalschlüssel beim Login
- Authentisiert Client am AP und vice versa
- Kann auch mit RADIUS Session Time Out Feature verwendet werden, um Schlüssel an dynamisch festgelegten Intervallen zu erzeugen
- Funktioniert nur mit Cisco Wireless Clients
- Gebrochen - ASLEAP von Joshua Wright
 - Wörterbuch-Angriffe zu leicht durchzuführen

10.12 EAP-TLS

- Anstelle eines Benutzername-/Passwort-Schemas verwendet EAP-TLS eine auf Zertifikaten beruhende Authentisierung
- Dynamische Erzeugung von Einmalschlüsseln
- Zwei-Wege-Authentisierung
- Verwendet TLS (Transport Layer Security), um die PKI-Information an den RADIUS Server weiterzutragen
- Mit vielen Betriebssystemen kompatibel
- Schwerer zu implementieren und "auszurollen", weil Schlüssel/Zertifikate für Clients erzeugt werden müssen

10.13 EAP-TTLS (Bob Funk), PEAP von Microsoft und Cisco

- Sehr ähnlich zu EAP-TLS; allerdings brauchen die Clients sich nicht selbst mit einem Zertifikat beim Server zu authentisieren
 - Phase 1: Es kann eine gefälschte Identität (bogus identity) vom Client verwendet werden (muss aber gut genug sein, um den Authentisierungsserver zu finden); nur der Server authentisiert sich in dieser Phase
 - Phase 2: Der durch TLS geschützte Kanal kann für ein einfaches Login-/Passwort-Schema verwendet werden (z.B. MSCHAPv2 verwenden)
- Leichter einzurichten, erfordert nicht notwendigerweise eine PKI
- PEAPv0 arbeitet ursprünglich mit Windows XP SP1, aber andere Plattformen beginnen es nun auch zu unterstützen; EAP-TTLS wird immer mehr von Opera Source Software unterstützt

EAP-Arten

	Offen / Proprietär	Gegen-seitige Auth.	Auth. Client	Auth. Server	Nutzername im Klartext
MD5	Offen	Nein	Ben./ Pass.	Kein	Ja
TLS	Offen	Ja	Zertifikat	Zertifikat	Ja
TTLS	Offen	Ja	Ben./ Pass.	Zertifikat	Nein
PEAP	Offen	Ja	Ben./ Pass.	Zertifikat	Nein
LEAP	Proprietär	Ja	Ben./ Pass.	Kein	Ja

10.14 WPA-Schritte (Enterprise)

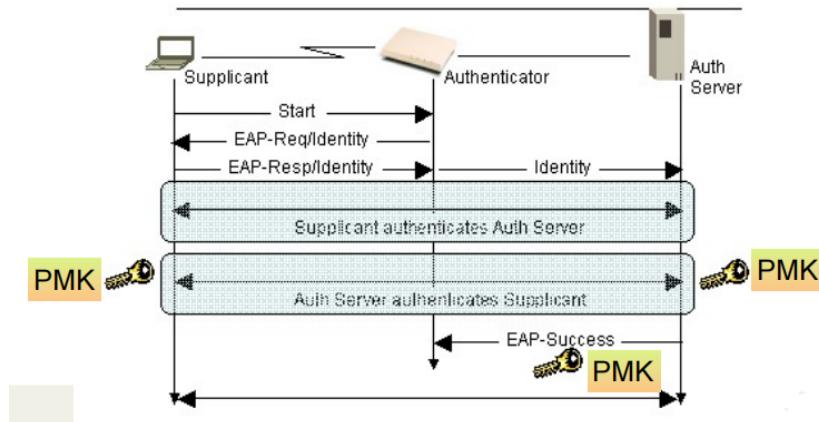
- 802.1x Authentisierung und Erzeugung von PMK
- 4-Way-Handshake und PTK-Installation
- Installation von Gruppenschlüsseln (GTK)
- Verschlüsselung mit TKIP (WPA) oder AES/CCM (WPA2)

10.14.1 802.1x Authentisierung + PMK

Pairwise Master Key (PMK)

- Authentisierungsprozess nutzt einen sicheren Kanal
- Dieser kann auch für die Erzeugung des PMKs verwendet werden
- PMK ist die Basis für die Erzeugung eines temporären WEP-Schlüssels in der nächsten Phase
- PMK wird erzeugt, indem das Ergebnis der Authentisierung des Benutzers zugrundegelegt wird

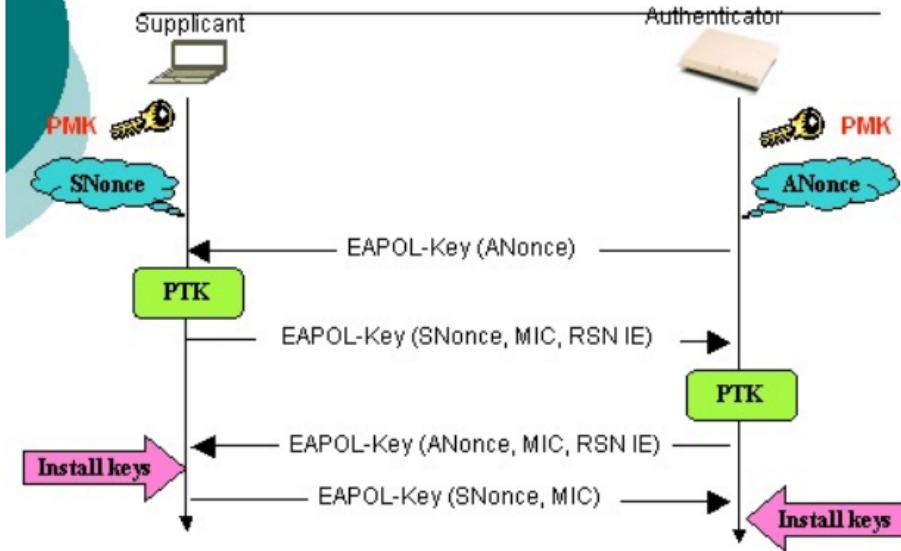
802.1x Authentisierung + PMK



10.14.2 Situation nach erfolgreicher Durchführung von EAP

- Supplicant (Station) und Authentisierungsserver haben sich gegenseitig authentisiert, sie kennen beide den PMK
- Authentisierungsserver sendet dem AP (Authentikator) den PMK
- Die Station und der AP müssen sich nun gegenseitig beweisen, dass sie den PMK kennen
 - Dieses Handshake erzeugt auch den PTK: ANonce (authenticator nonce) und SNonce (supplicant nonce) stellen sicher, dass der PTK frisch ist

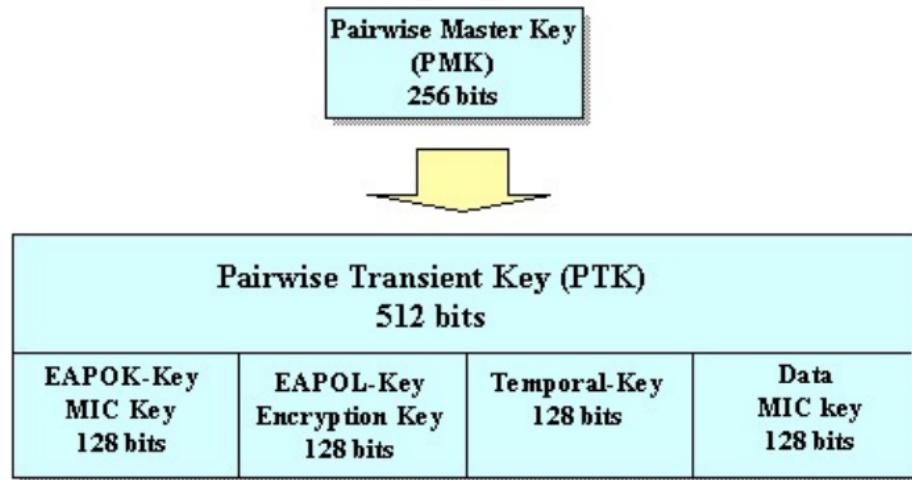
4 Way Handshake und PTK



10.14.3 4-Way-Handshake und PTK

- Verwende PMK nicht direkt für die Krypto
- Erzeuge Pairwise Transient Key PTK (512 bits) aus dem PMK und Nonces
- Aufgeteilt in 4 Teile, jeder einzelne ist 128 bit lang:
 - Verschlüsselung der Daten, Datenintegrität, EAPOL-Key Verschlüsselung, EAPOL-Key Integrität
- Ein Teil des PTKs wird verwendet, um den Schlüssel für die Verschlüsselung (äquivalent zu WEP) in der nächsten Phase zu erzeugen

4 Way Handshake und PTK



10.14.4 KRACK: Key Re-installment AttaCKs

- IV ist vorhersagbar → wiederholte Benutzung eines Schlüssels deckt Nutzdaten auf
- KRACK: Client dazu überlisten, einen (korrekt ausgehandelten!) Schlüssel zweimal zu installieren (und damit beim IV wieder vorne anfangen)
- N.B.: Das 4-way-handshake-Protokoll ist "formal verifiziert"!

10.14.5 Gruppenschlüssel

- Problem: Broadcasts (AP zu Stationen) können keine paarweisen Schlüssel verwenden
 - Broadcast-Pakete von den Stationen werden hingegen zu den APs zunächst per Unicast verschickt - für diesen Teil des Weges wird der PTK genutzt
 - Separater Group Transient Key (GTK)
 - Wird gesendet, nachdem paarweise sichere Verbindungen eingerichtet worden sind
 - Muss nach jeder Dis-Assozierung neu erzeugt werden!
 - * WEP Key-ID Feld wird wiederverwendet, um einen nahtlosen Übergang (transition) zu gewährleisten

10.14.6 Hole 196

- S. 196: GTK ist allen Stationen bekannt, also kein Schutz gegen Ein-spielung von Broadcast/Multicast

NOTE—Pairwise key support with TKIP or CCMP allows a receiving STA to detect MAC address spoofing and data forgery. The RSNA architecture binds the transmit and receive addresses to the pairwise key. If an attacker creates an MPDU with the spoofed TA, then the decapsulation procedure at the receiver will generate an error. GTKs do not have this property.

- Ethernet-Modell: jeder kann Broadcast sehen
- Entwicklungen bei Basisstationen: proprietäre Hacks (gegen ARP-Poisoning, ra-guard, etc.) unterdrücken BC/MC
- Veränderte Sicherheitsanforderungen → S. 196 ist plötzlich eine Sicherheitslücke

10.14.7 TKIP (Temporal Key Integrity Protocol)

- Problem: Alte Hardware ist nicht leistungsfähig genug, um AES-CCMP zu unterstützen; RC4 wird also weiter verwendet

TKIP:

- Vergrößert IV-Raum (24 → 48 Bits)
- IV-Sequenz wird festgelegt
 - TSC (TKIP sequence counter) bietet Schutz vor Replay-Angriffen
- Mixing Function erzeugt für jedes Paket den 40-Bit- (104-Bit-) Anteil
 - Man kann auch mit Legacy Hardware arbeiten, die eine 24+40-Struktur erwartet
 - Auch MAC-Adresse beim Mischen berücksichtigen, um die Wiederverwendung des IV zwischen Systemen zu vermeiden
- MIC (message integrity code): Michael
 - Sehr billige Integritätsprüfung für MAC-Adressen und Daten

10.14.8 Der MIC-Tradeoff

- Die meisten guten Verfahren zur Überprüfung der Integrität von Nachrichten sind zu teuer
- Micheal ist schnell und billig
 - Aber nur begrenzte Widerstandsfähigkeit

- Wird zum WEP ICV (CRC) hinzugefügt, der immer noch auf der MPDU-Ebene angewendet wird
- Michael wird auf der MSDU-Ebene angewendet
- Angriffe würden Millionen von Paketen benötigen
 - ”Gegenmaßnahmen“ (60-Sekunden-Blackout), wenn ein Angriff entdeckt wird
- Erzeugt das altbekannte DoS-Problem
 - Es gibt allerdings einfachere Möglichkeiten für DoS-Angriffe in drahtlosen Netzen

10.14.9 WPA-PSK

- Geeignet für zuhause/Einsatz im Bereich Small Office bzw. Home Office (SOHO)
- Entfernt 802.1X-Authentisierung
 - Pre-shared Key (”PSK“) wird aus einer Passphrase via Password-based Key Derivation Function PBKDF2 (RFC2898) erzeugt
- WPA-PSK = Pre-shared Key + TKIP
- Verwundbar gegenüber passivem Wörterbuchangriff
 - Wähle lange, komplexe PSKs
- Immer noch viel, viel besser als WEP

10.14.10 WPA3

Fixes für WPA2:

- ”Simultaneous Authentication of Equals“ (RFC 7664)
- Offline-Angriffe auf Passwort schwieriger
- Forward secrecy
- Opportunistic Wireless Encryption (RFC 8110)
 - → WiFi Enhanced Open
- WiFi Protected Setup → WiFi Easy Connect
- Protection of management frames (802.11w) erforderlich
 - Verhindert disassociation attack

10.15 Was ist beim Ausrollen eines WLANs zu berücksichtigen?

- Verstecke SSID (??)
- Verwende **nicht** WEP
- Verwende WPA mit 802.1x falls möglich
- Oder verwende WPA wenigstens mit einem sehr komplexen Pre-shared Key
- Und/oder verwende VPNs

10.16 Fazit

- ”If you **compromise** on security, your security will be compromised”
- Führe ein Review der Sicherheit schon **früh** während des Entwicklungsprozesses durch
- Verteilen von sicherheitskritischen Funktionen in Millionen von nicht-änderbaren Hardwaregeräten **wird** früher oder später zu einem Problem führen
- *With sufficient thrust, pigs fly just fine*
 - *However, this is not necessarily a good idea. It is hard to be sure where they are going to land, and it could be dangerous sitting under them as they fly overhead.* (RFC 1925: Fundamental truths of networking, 1. April 1996)

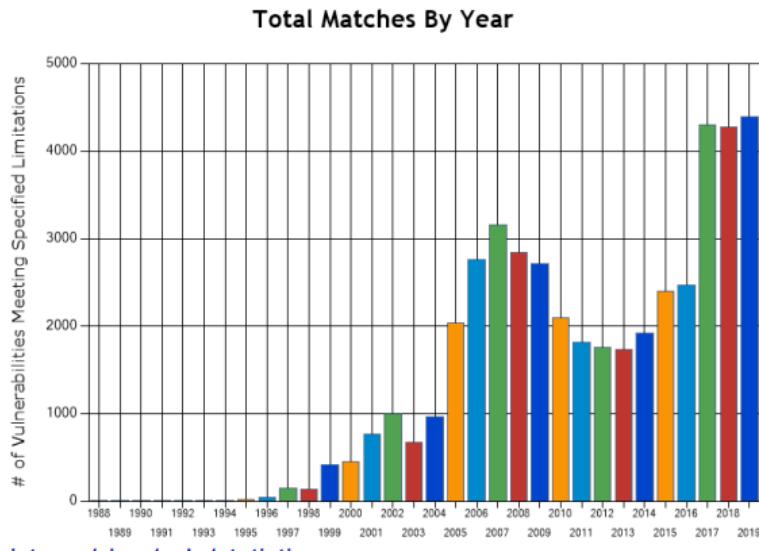
Chapter 11

Software Security

11.1 Sicherheitslücken in Software: Heartbleed Bug

- Sicherheitslücke in OpenSSL 1.0.1 bis 1.0.1f
 - April 2014
 - ab 1.0.1g behoben
- Angreifer kann speziell fabrizierte Nachrichten an Server schicken und dort unerlaubte Arbeitsspeicherzugriffe machen (Länge unter Kontrolle des Angreifers, Arraygrenzenfehler)
- Server sendet einfach Speicherinhalte an den Angreifer zurück
 - Mit privaten Schlüsseln (z.B. zum Server-Zertifikat gehörig)
 - Mit Sitzungsschlüsseln (z.B. für TLS-Kommunikation)
- OpenSSL kryptographische Bibliothek, die u.a. im Software-Stack der meisten Web-Server weltweit enthalten ist
 - → Hohe Angriffsfläche, Rücknahme des Schlüssel aufwendig

Schwerwiegende Lücken



11.2 Trinity of Trouble

- Gary McGraw: Software Security, Addison Wesley, 2006
- "Trinity of Trouble"
 - Steigende Komplexität (Windows 8 bis zu 80 Mio. Lines of Code?)
 - Wachsende Vernetzung (SOA, Industrie 4.0, Internet der Dinge, ...)
 - Erweiterbarkeit von Systemen (Nachladen von Apps, Plugins für Browser)

11.3 Software Security als eigene Disziplin

- Gängige Sicherheitsmechanismen wie z.B. Firewalls, Antiviren-Software oder Intrusion Detection Systeme sind reaktiv
- Ursache für Sicherheitsprobleme: Lücken in Software
- Werkzeuge und systematische Vorgehensweisen zur Verbesserung der Software-Security
 - Security Development Lifecycle (SDL)

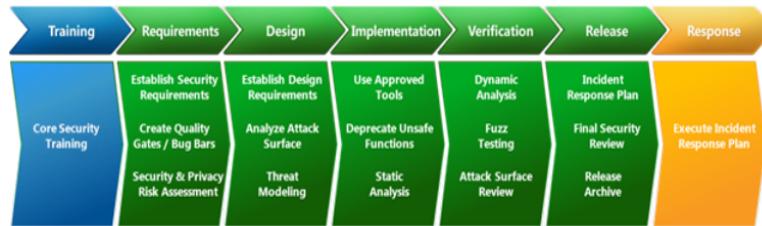
11.4 SDL - Security Development Lifecycle

- Viele große Software-Hersteller (z.B. Microsoft, SAP, Adobe, Siemens) setzen Prozesse zur Entwicklung sicherer Software ein
- Orientierung des SDL an Software-Entwicklungsprozessen
 - Wasserfall
 - Iterative Software-Entwicklung
 - Agile Software-Entwicklung
- Umfasst alle Phasen der SW-Entwicklung und injiziert dort Security-relevante Schritte

Beispiele für SDLs

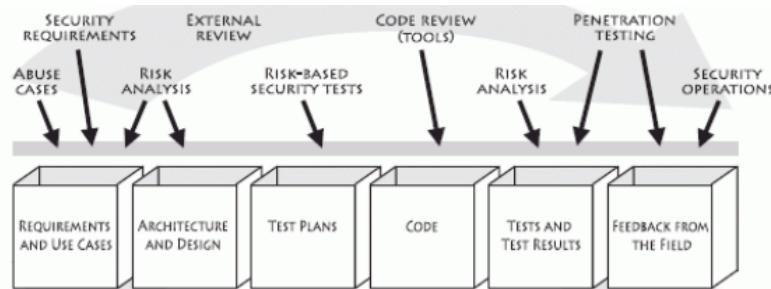
- Microsoft SDL: <https://www.microsoft.com/en-us/sdl/>
- Seven Touchpoints, Gary McGraw: <http://www.swsec.com/resources/touchpoints/>
- Beide Prozesse umfassen ähnliche Aktivitäten, die z.T. nur anders benannt sind

11.4.1 SDL von Microsoft



Die Schritte werden in Abhängigkeit des SW-Entwicklungsmodell wiederholt: Zyklus!

11.4.2 Seven Touchpoints nach McGraw



11.4.3 Wichtige Aktivitäten innerhalb eines SDLs

- 1. Code Review mit Werkzeugen (vor allem statische Code-Analyse)
- 2. Architekturelle Risikoanalyse/Thread Modeling
- 3. Penetration Testing

11.4.4 Wiederholung: Buffer-Overflows

- Häufigste Einbruchsmethode in Server (insbesondere in Web-Server)
- Altbekanntes Problem (schon in den 60er Jahren bekannt)
- "Attack of the decade" (Bill Gates)
- Die meisten Viren/Würmer nutzen Buffer-Overflows aus (Morris-Wurm, Code Red, Blaster)
- **Ziel:** Einschleusen von Code
- Ausnutzen von Programmierfehlern

```
void trouble() {
    char line[128];
    gets(line); /* lies von stdin */
}
```

```
void trouble1(char* input) {
    char line[128];
    strcpy(line, input);
}
```

11.4.5 Integer Overflows

Beispiel aus Chess, West: Secure Programming with Static Analysis, Addison-Wesley, 2007

```
unsigned int readamt;
readamt = getstringsize();
if(readamt > 1024) return -1;
readamt--;//don't allocate space for '\n'
buf = malloc(readamt);
```

11.4.6 Probleme mit der Speicherverwaltung

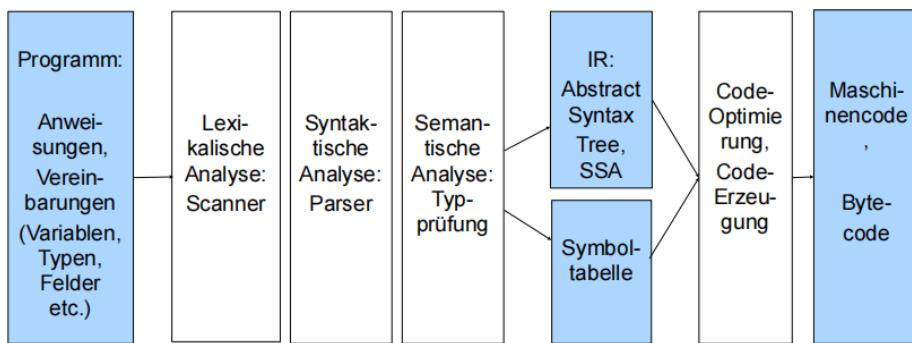
```
double_free(int i) {
    int *p;
    p = (int*) malloc(42*sizeof(int));
    if(i <= 0) free(p);
    p[0] = 2;
    p[1] = 3;
}
```

Ähnliche Fehler: Arraygrenzenfehler, Speicherleaks, ...

11.5 Code Review mittels statischer Programmanalyse

- Sicherheitsanalyse des Quelltextes von Programmen
- Aufdecken von gängigen Programmierfehlern wie z.B.:
 - Buffer-Overflows, Heap Overflows, Integer Overflows
 - SQL-Injection-Verwundbarkeiten
 - Cross-Site-Scripting-Verwundbarkeiten
- Automatisierte Analyse

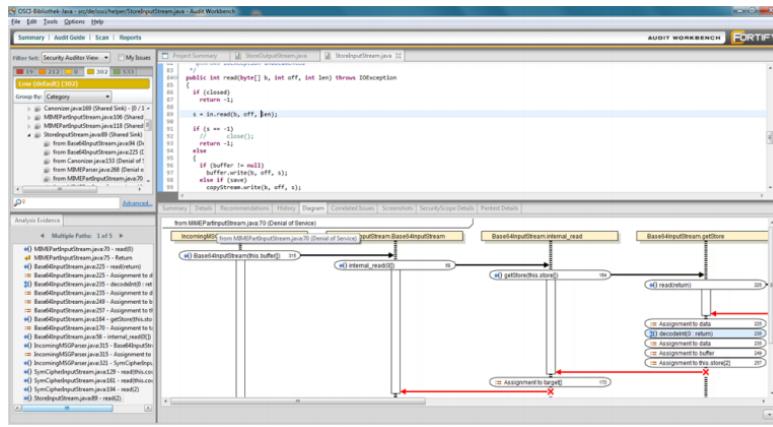
11.5.1 Phasen eines Compilers



11.5.2 Statische Programmanalyse

- Einsatz von Compilerbau-Techniken
 - Zwischen (IR) des Programmes z.B. durch Abstract Syntax Trees, **Static Single Assignment** (SSA)
 - Daten- und Kontrollflussanalysen auf Zwischendarstellungen
- **False Positives** (Fehlalarme), **False Negatives** (übersehene Fehler) aufgrund von Nicht-Entscheidbarkeit
- Gängige kommerzielle Werkzeuge: Fortify SCA (Java-Code), IBM AppScan, Veracode, Coverity Prevent (für C/C++ Code), Checkmarx, SonarQube
- Einfache statische Analysen eingebaut in Compiler wie gcc oder Clang

Screenshot Fortify



11.5.3 Statische Programmanalyse: SonarQube

- Statisches Analysewerkzeug zur Bewertung der Code-Qualität im Allgemein
- Auch Unterstützung von Security Rules: z.B. SQL-Injection- und XSS-Verwundbarkeiten, OWASP Top 10 (Liste mit gängigen SW-Schwachstellen, OWASP - Open Web Application Security Project)
- Frei verfügbare Fassung, aber auch kostenpflichtige Commercial Edition
- In der Praxis weit verbreitet
- Unterstützte Sprachen: u.a. Java, Python, Groovy, JavaScript, PHP, TypeScript, HTML, C#
- <https://www.sonarqube.org/features/security/>

11.5. CODE REVIEW MITTELS STATISCHER PROGRAMMANALYSE 167

Screenshot SonarQube

The screenshot displays two side-by-side code review interfaces from SonarQube. Both panels show Java code with various annotations and error markers.

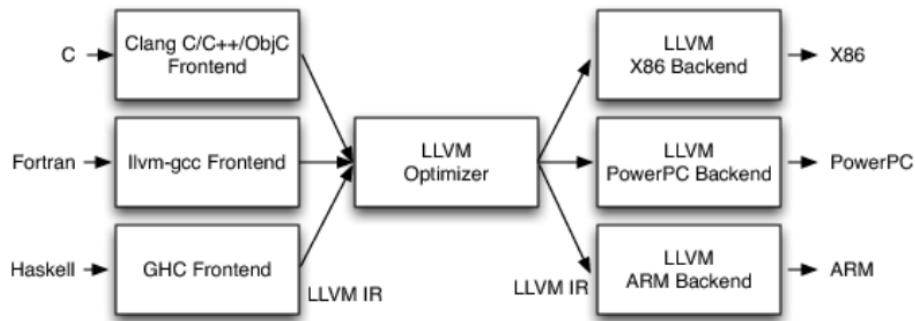
Top Panel: The code is located at `src/main/java/devoxo/security/injection/Servlet.java`. It contains several annotations (e.g., `@TaintedRequest`, `@URLDecoder`) and a try-catch block. Red annotations highlight parts of the code, and a tooltip for one annotation reads: "Refactor this code to not construct the OS command from tainted, user-controlled data." Other annotations include `SQLInjectionVulnerability`, `CommandInjectionVulnerability`, and `RegexInjectionVulnerability`.

Bottom Panel: The code is located at `src/main/java/devoxo/security/injection/CommandInjectionVulnerability.java`. It shows a conditional statement and a call to `runtime.exec(command)`. A specific line of code (`command[command.length - 1] = path;`) is highlighted in red. A tooltip for this line reads: "Refactor this code to not construct the OS command from tainted, user-controlled data." Other annotations include `Blocker`, `Confirmed`, `Not assigned`, `30min effort`, and `Comment`.

11.5.4 Statische Programmanalyse: Clang

- Statische Sicherheitsanalysewerkzeuge oft teuer
- Einsatz von frei verfügbaren statischen Analysewerkzeugen
- LLVM: Compiler Infrastructure
- Unterstützt verschiedene Frontends: C/C++, Objective-C, Haskell, Fortran, (Java)
- Clang Static Analyser von LLVM: <http://clang-analyzer.llvm.org/>

Design von LLVM



11.5.5 Clang Static Analyzer

- Clang: C/C++ Frontend von LLVM
- Clang Static Analyzer: Statisches Analysewerkzeug für Clang: <http://clang-analyzer.llvm.org/>
- Unterstützt Analysen (http://clang-analyzer.llvm.org/available_checks.html), u.a.:
 - Nullpointer-Dereferenzierung
 - Unsichere Verwendung von C/C++ Funktionen wie z.B. gets(), strcpy()
 - Arraygrenzfehler
- Manche Analysen noch experimentell, recht eingeschränkter Regelsatz

Beispiel für eine zu prüfene Sicherheitsregel

Regel: alpha.security.ArrayBoundV2 (C): Warn about buffer overflows (newer checker)

```
void test() {
    char *s = "";
    char c = s[1]; // warn
}
```

```
void test() {
    int buf[100];
    int *p = buf;
    p = p + 99;
    p[1] = 1; // warn
}
```

11.5.6 Statische versus dynamische Programmanalyse

- **Statische Analyse:** Durchführung der Programmanalyse auf Quelltext oder Zwischendarstellung (IR) ohne Ausführung des Programms, z.B. zur Compilezeit
- **Dynamische Analyse:** Durchführung der Analyse im laufenden Betrieb der Software. (Beispiel mittels Code-Instrumentierung)

11.5.7 Dynamische Analyse mittels Code-Instrumentierung

- Idee: Compiler setzt **zusätzliche Sicherheitsprüfungen** beim Verwenden von kritischen Funktionen wie strcpy(), sprintf(), strcat(), malloc(), free() in den Code ein
- Fachbegriff dafür: Code-Instrumentierung
- Vorteil: Keine oder **wenige False Positives**, Analyseergebnis genauer
- Nachteil: Code ist größer und weniger effizient
- Anderes Beispiel für Code-Instrumentierung: Canaries
 - Einfügen von Canaries in den Unterprogrammstack durch Compiler

Beispielwerkzeug: AddressSanitizer

- Auf Basis von Code-Instrumentierung: Google AddressSanitizer (oder ASan); to sanitize - reinigen: <https://github.com/google/sanitizers/wiki/AddressSanitizer>

- Memory Error Detector für C++
- Durchschnittliche Verlangsamung des instrumentierten Programmes: Faktor 2
- Run-time library ersetzt die malloc() und free() Funktionen.
- Integration in LLVM/Clang (ab Version 3.1) und auch gcc (ab Version 4.8)
- Aufruf: -fsanitize=address
- Unterstützte Checks, z.B.:
 - Use after free (dangling pointer dereference)
 - Heap buffer overflow
 - Stack buffer overflow
 - Use after return
 - Use after scope
 - Initialisation order bugs
 - Memory leaks

Fehlerhaftes Programm: Zugriff auf Variable nach Speicherfreigabe

```
#include <stdlib.h>
int main() {
    char *x = (char*)malloc(10 * sizeof(char*));
    free(x);
    return x[5];
}

% ..../clang.build.Linux/Release+Asserts/bin/clang -fsanitize=address -O1 -
fnoomit-frame-pointer -g tests/use-after-free.c
```

Ausgabe von AddressSanitizer

```
% ./a.out
==9901==ERROR: AddressSanitizer: heap-use-after-free on address 0x60700000dfb5 at pc 0x45917b bp 0x7fff4490c700 sp 0x7fff4490c6f8
READ of size 1 at 0x60700000dfb5 thread T0
#0 0x45917a in main use-after-free.c:5
#1 0x7fce9f25e76c in __libc_start_main /build/buldd/eglibc-2.15/csu/libc-start.c:226
#2 0x459074 in _start (a.out+0x459074)
0x60700000dfb5 is located 5 bytes inside of 80-byte region [0x60700000dfb0,0x60700000e000)
freed by thread T0 here:
#0 0x4441ee in __interceptor_free projects/compiler-rt/lib/asan/_asan_malloc_linux.cc:64
#1 0x45914a in main use-after-free.c:4
#2 0x7fce9f25e76c in __libc_start_main /build/buldd/eglibc-2.15/csu/libc-start.c:226
previously allocated by thread T0 here:
#0 0x44436e in __interceptor_malloc projects/compiler-rt/lib/asan/_asan_malloc_linux.cc:74
#1 0x45913f in main use-after-free.c:3
#2 0x7fce9f25e76c in __libc_start_main /build/buldd/eglibc-2.15/csu/libc-start.c:226
SUMMARY: AddressSanitizer: heap-use-after-free use-after-free.c:5 main
```

11.6 Nicht nur Bugs, sondern auch Flaws

- Designfehler (Flaws) 50% aller Sicherheitsprobleme von Software
- Beispiele
 - Rollenbasierte Zugriffskontrolle inkonsistent über mehrere Schichten einer Multi-Tier-Anwendung hinweg
 - Falscher Einsatz von Krypto (Schlüssellänge zu klein)
 - Schutz der Integrität von Daten, wenn eigentlich Vertraulichkeit benötigt
- Lösung: Führe Architekturelle Risikoanalyse durch

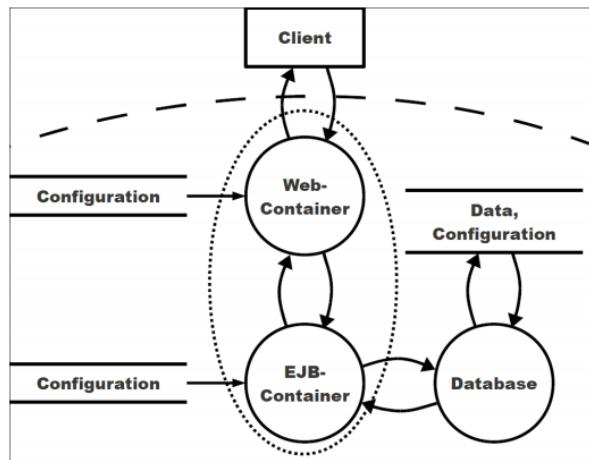
11.7 Architekturelle Risikoanalyse

- Aktivität in der Designphase von Software
 - Später gefundene Lücken teuer zu behandeln
- MS-SDL: Threat Modeling, G. McGraw: Architekturelle Risikoanalyse (ARA)
- Grobe Idee:
 - 1. Erstelle Diagramme der Software
 - * Forest-Level-Übersicht, nicht Tree Level
 - 2. Diskutiere anhand des Bildes systematisch Design-Schwächen (manueller Prozess!)

11.7.1 Weitere Schritte der architekturellen Risikoanalyse nach McGraw

- 1. Durchsuchen der Literatur nach **bekannten architekturellen Sicherheitsproblemen** (z.B. CWE - Common Weakness Enumeration, CAPEC - Common Attack Pattern Enumeration and Classification)
- 2. Identifizierung von Sicherheitsproblemen **verwendeter Frameworks**
- 3. Zusammensetzen in einer Gruppe, um **unbekannte Risiken** zu identifizieren
 - Parallelisierung
 - Forest-Level-Abbildung hilfreich

Threat Modeling/ARA mit Datenflussdiagrammen



11.8 Penetrationstests

- Integration in SW-Entwicklungsprozess
 - Ergebnisse der architekturellen Risikoanalyse nutzen - zielgerichtete Pen-Tests
 - Ergebnisse der Pen-Tests in den SW-Entwicklungszyklus einbringen;
 - Nicht nur das offensichtliche Sicherheitsproblem beheben:
 - * Declare victory and go home
- Nicht als erste Sicherheitsüberprüfung im SW-Entwicklungsprozess, letzter Test vor dem Deployment!
- Werkzeuge nutzen

11.8.1 Software Penetrationtests - Werkzeuge nutzen

- Reverse Engineering mittels Disassembler (z.B. IDA, www.hex-rays.com/idapro/) und Debugger
- Testen von Web-Anwendungen
 - OWASP Zed Attack Proxy Project (https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project)
 - Burp-Suite (<https://portswigger.net/burp/>); dynamisches Testen von Web-Anwendungen inklusive MITM-Angriffen
 - HP WebInspect - dynamisches Testen von Web-Anwendungen
 - Fuzzer

11.9 Fuzzing

- Wichtiges Hilfsmittel für Penetrationstests
- **Idee:** Zufällig fehlerhafte Daten erzeugen und dann (Web-) Applikation automatisiert damit testen
- Man kann Fuzzer auch für spezielle Anwendungen und Protokolle schreiben
 - Sogar für Automobilelektronik
 - (in Masterarbeit an der Uni Bremen) für App-gesteuerte Alarmanlagen
- Liste von Fuzzern auf den OWASP-Webseiten: <https://www.owasp.org/index.php/Fuzzing>

11.9.1 Beispiel: american fuzzy lop

- **Problem:** Wie Codeabdeckung eines Fuzzers erhöhen?
- Lösung für C/C++-Programme: american fuzzy lop, Open Source: <https://lcamtuf.coredump.cx/afl/>
- Entwickler: Michal Zalewski (Google)
- Ansatz:
 - Erzeugt aus einer kleinen Menge von Testsamples automatisiert viele Fälle
 - Testfälle **passen zur erwarteten Eingabe** des Analysekandidaten
 - Hierzu: Instrumentierung des Analysekandidaten (Quellcode, Binaries), Genetische Algorithmen zur Testfallerzeugung (Kontrollfluss des Analysekandidaten berücksichtigen)
- Generierte Testfälle können auch als Eingabemenge für andere Fuzzer verwendet werden

Animation: american fuzzy lop

- <https://lcamtuf.coredump.cx/afl/rabbit.gif>
- Zum Ausprobieren
- Hintergrund:
 - Ziel: Testen des Programmes ImageMagick
 - Startpunkt das afi-Logo (als GIF)
 - Darstellung aller hieraus generierten Testfälle als Animation
 - Testfälle werden von ImageMagick akzeptiert

Chapter 12

Sicherheitskriterien

12.1 Motivation: Sicherheitskriterien

- Viele IT-Produkte müssen bestimmte Sicherheitsziele erfüllen:
 - Betriebssysteme, Anwendungen, Datenbanken, Chipkarten, ...
- **Zentrale Frage:** Wie kann man die "Sicherheit" solcher Produkte bewerten/evaluieren?
- Allgemeine Anerkennung der Evaluationsergebnisse
- Möglichkeit des Vergleichs von IT-Produkten mit ähnlicher Funktionalität
- Einführung einer Evaluationsmethodik

12.2 Kriterienkataloge

- TCSEC (Trusted Computer Evaluation Criteria, 1980)
- ITSEC (Information Technology Security Evaluation Criteria, Europa, 1991)
- CTCPEC (Canadian Trusted Computer Product Evaluation Criteria, 1993)
- CC (Common Criteria, 1996; V3.1 vom September 2006)

12.3 Trusted Computer Evaluation Criteria - TC-SEC

- US National Computer Security Center (Teil der NSA), 1980
- Auch "Orange Book" genannt

- Älteste Kriterien zur Bewertung der Sicherheit von IT-Produkten
- Sieben **Hierarchiestufen** zur Sicherheitsstufen zur Klassifikation eines IT-Systems
 - Vier Hauptstufen A bis D
 - Fünf Unterstufen (C1, C2, B1, B2, B3)
 - Niedrigste Stufe ist D

12.3.1 Die Hierarchiestufen von TCSEC

- **Stufe D:** Minimaler Schutz (Im Grunde gar keine Aussage über Schutz)
- **Stufe C1:** Benutzerbestimmbare Zugriffskontrolle (DAC) für Gruppen:
 - Benutzer können Rechte nur grobgranular an Gruppen vergeben
 - Beispiel: gängige Unix-Systeme
- **Stufe C2:** Benutzerbestimmbare Zugriffskontrolle (DAC) für einzelne Benutzer und Audit
 - Benutzer können Rechte auch an einzelne Benutzer vergeben
 - Mitprotokollieren der Aktionen einzelner Nutzer
 - Schutz der Audit- und Authentisierungsinformationen
 - Beispiel: Windows NT (ohne Netzanbindung!), RACF-OS von IBM, div. speziell für C2 umgerüstete Unix-Varianten
- **Stufe B1:** Systembasierte Zugriffskontrolle (MAC)
 - Wie C2, aber zusätzlich Einführung von Sicherheitsmarkierungen (vgl. Bell-LaPadula)
- **Stufe B2:** Strukturierter Schutz
 - Wie B1, u.a. zusätzlich
 - * Formales Sicherheitsmodell (**security model**)
 - * Die **Trusted Computing Base** (TCB) muss strukturiert sein: Trusted Computing Base = die Komponenten eines IT-Systems, die korrekt arbeiten müssen, damit die Security Policy nicht verletzt wird
 - * Analyse von verdeckten Kanälen
 - * Strikte Tests müssen durchgeführt werden
- **Stufe B3:** Security Domains
 - Wie B2, aber TCB muss **minimal**, ausführlich getestet und gegen unbefugte Zugriffe geschützt sein

* Vgl. Principle of Economics of Mechanism

- Echtzeit-Monitoring und Mechanismen zur Benachrichtigung

- **Stufe A:** Verifiziertes Design

- Keine funktionalen Erweiterungen von B3, aber es wird ein formaler Nachweis gefordert, dass die TCB-Spezifikation und das zugrunde liegende Sicherheitsmodell äquivalent sind
- Beispiel: MLS LAN Version 2.1 der Firma Boeing

12.3.2 Bewertung: TCSEC

- Nur auf Betriebssysteme beschränkt
- Betonung von Bell-LaPadula-Policies, Vernachlässigung anderer Schutzziele wie der Integrität und der Verfügbarkeit
- Nur auf USA beschränkt
- Keine Trennung von **Sicherheitsfunktionalität** und der **Qualität**, mit der die Sicherheitsfunktionen erbracht werden

12.4 ITSEC

- Europäische Kriterien (Großbritannien, Frankreich, Deutschland, Niederlande), 1991
- Einführung von **Funktionalitätsklassen F**
 - Teilweise Orientierung an den Stufen des Orange Books (F-C1, F-B1, ...)
 - Einführung weiterer Funktionsklassen: bspw. F-IN besondere Anforderungen an die Integrität
- Sieben verschiedene **Evaluationsstufen** E0, ..., E6 (Qualitätsstufen)
 - E0: Einstiegsstufe
 - E6: formale Spezifikation der Sicherheitsanforderungen
- **Weitere Neuerung:** Der Produkthersteller muss die Kosten für Zertifizierung übernehmen
- Gegenseitige Anerkennung der Zertifizierungsergebnisse in europäischen Staaten, bis auf Stufe E6

12.5 Die Common Criteria

- Common Criteria V1.0, 1996
- Aktuelle Version: Common Criteria V3.1, September 2006: <http://www.commoncriteriaportal.org/>
- Zusammenführung von CTCPEC, TCSEC und ITSEC
- Im Wesentlichen Orientierung an ITSEC
- **Weltweit einheitlicher** Kriterienkatalog
 - u.a. unter Beteiligung von Deutschland, Frankreich, Großbritannien, Kanada, den Niederlanden und den USA
- Internationaler Standard: ISO/IEC 15408
- Ziel: Evaluation von sicherheitsrelevanten Produkten
 - **EVG-Evaluationsgegenstand (target of evaluation, TOE)**: ein IT-Produkt oder -System, das Gegenstand einer Prüfung ist

12.5.1 Zusammenhang mit ITSEC

- Weiterentwicklung von ITSEC
- Trennung von Sicherheitsfunktionalität und Qualitätskriterien
- CC-Evaluierung wird von national **akkreditierten Prüfstellen (commercial licensed evaluation facility, CLEF)** durchgeführt
 - derzeit 7 Prüfstellen in Deutschland (z.B. T-Systems International GmbH, TÜV Informationstechnik GmbH, DFKI); im Jahr 2013 noch 15
 - Prüfstellen werden in Deutschland vom BSI lizenziert
 - Akkreditierung in den USA von der NSA und dem NIST
- Die Kosten der Zertifizierung werden wie bei ITSEC vom Produkthersteller übernommen!

12.5.2 Struktur der CC-Dokumente

- Drei Teile:
 - Teil 1: Einführung in die CC und allgemeines Modell
 - * Glossar, Grundlagen der Evaluation, Schutzprofile, Sicherheitsvorgaben
 - * Verständliche Einführung
 - Teil 2: Funktionale Sicherheitsanforderungen

- * Kataloge von Empfehlungen für Sicherheitsanforderungen
 - Teil 3: Anforderungen an die Vertrauenswürdigkeit (**assurance**)
- Sowie: CEM - Evaluations-Methodologie (ISO/IEC 18045, V3.1)

12.5.3 Funktionale Sicherheitsanforderungen

- Definiert in Teil 2 der CC:
 - Sicherheitsprotokollierung (FAU)
 - Kommunikation (FCO)
 - **Kryptographische Unterstützung** (FCS)
 - Schutz der Benutzerdaten (FDP)
 - Identifikation und Authentisierung (FIA)
 - Sicherheitsmanagement (FMT)
 - Privatheit (FPR)
 - Schutz der EVG-Sicherheitsfunktionen (FPT)
 - Betriebsmittelnutzung (FRU)
 - EVG-Zugriff (FTA)
 - Vertrauenswürdiger Pfad/Kanal (FTP)
- Funktionale Sicherheitsanforderungen an einen Evaluationsgegenstand (Auszug)
 - Klasse FCS - Kryptographische Unterstützung
 - * FCS_COP.1 Kryptographischer Betrieb ("Hashen")
FCS_COP1.1 ... müssen Hashfunktion gemäß SHA256 durchführen
...
 - * FSC_COP.2 Kryptographischer Betrieb ("Verifizieren")
FCS_COP2.1 ... müssen Verifikationsalgorithmus gemäß RSA (2048 Bit) durchführen ...
 - * FSC_CKM.1 Schlüsselerzeugung
FCS_CKM1.1 ... müssen die Schlüssel für das RSA mit der Größe [1024 Bit, 2048 Bit] gemäß Standard PKCS #1 erzeugen

12.5.4 Vertrauenswürdigkeits-anforderungen

- Definiert in Teil 3 der CC:
 - Konfigurationsmanagement (ACM)
 - Auslieferung und Betrieb (ADO)
 - Entwicklung (ADV)
 - Handbücher (AGD)

- Lebenszyklus-Unterstützung (ALC)
- Testen (ATE)
- Schwachstellenbewertung (AVA)
- Erhaltung der Vertrauenswürdigkeit (AMA)

- Prüfung und Bewertung des **Schutzprofils** (APE)
- Prüfung und Bewertung der **Sicherheitsvorgabe** (ASE)

Beispiele für Vertrauenswürdigkeitsanforderungen

- Entnommen aus Teil 3, CC

ATE_DPT.3 Testing: modular design

Developer action elements:

ATE_DPT.3.1D The developer shall provide the analysis of the depth of testing.

Content and presentation elements:

ATE_DPT.3.1C The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test documentation and the TSF subsystems and modules in the TOE design.

ATE_DPT.3.2C The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have been tested.

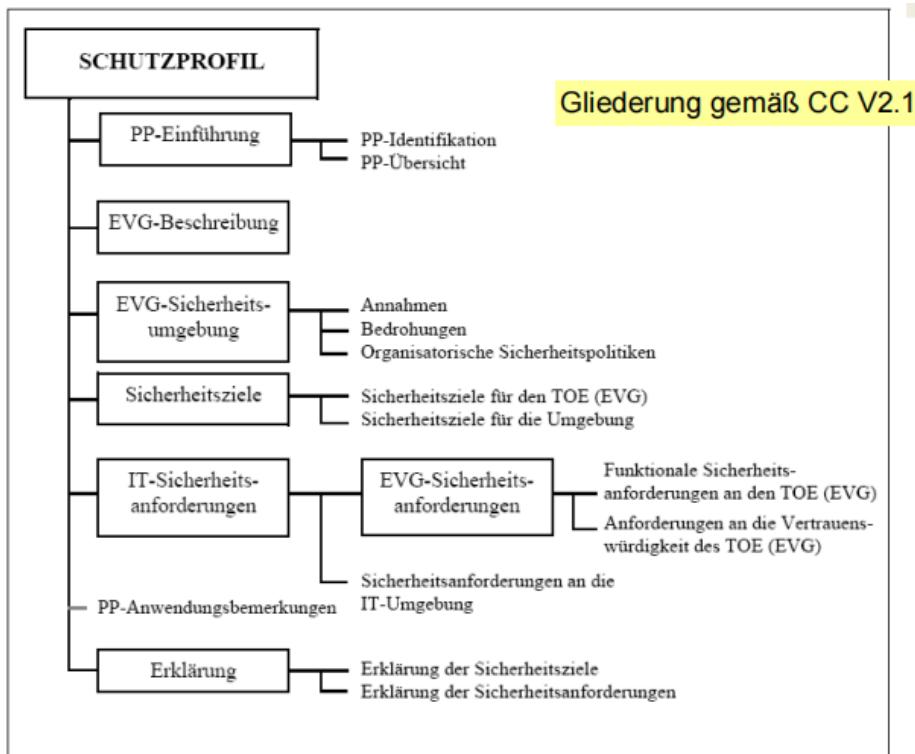
ATE_DPT.3.3C The analysis of the depth of testing shall demonstrate that **all** TSF modules in the TOE design have been tested.

12.5.5 Vertrauenswürdigkeitsstudien

- Vertrauenswürdigkeitsstudie (**Evaluation Assurance Level**, EAL), Bewertungsskala
 - EAL1: funktionell getestet
 - EAL2: strukturell getestet
 - EAL3: methodisch getestet und überprüft
 - EAL4: methodisch entwickelt, getestet und durchgesehen
 - EAL5: semiformal entworfen und getestet
 - EAL6: semiformal verifizierter Entwurf und getestet
 - EAL7: formal verifizierter Entwurf und getestet
- EALs sind Sammlungen von "Vertrauenswürdigkeitsanforderungen" aus Teil 3 der CC
- EALs entsprechen im Wesentlichen den Evaluationsstufen von ITSEC:
 - EAL2 entspricht E1, ..., EAL 7 entspricht E6
 - Kein Pendant zu E0
 - EAL1 wurde neu eingeführt.

12.6 Schutzprofil

- Ein **Schutzprofil** (*protection profile*) enthält die Sicherheitsanforderungen für eine **Klasse von Produkten**
- Darstellung in einer **implementierungsunabhängigen** Form
- Beispiele:
 - Firewalls (Paketfilter/Application Gateways)
 - Rollenbasierte Zugriffskontrolle
 - Krankenhausinformationssysteme
 - Chipkarten, elektronische Ausweise
 - Heilberufsausweis für Ärzte
 - Weitere Profile siehe <http://www.commoncriteriaportal.org/pp.html>



12.6.1 Bestandteile eines Schutzprofils

- 1. **Bedrohungen (threats): Beispiel:** T.ArchivierungWahlgeheimnis
Eine Person, die nach der Phase "Wahldurchführung inkl. der Stimmabzählung" Zugriff auf die im EVG gespeicherten Daten hat und ggf. Zusatzdaten wie beispielsweise Entschlüsselungskeys kennt, kann an Hand der im EVG gespeicherten Daten eine Zuordnung zwischen dem Wähler und seiner Stimme (im Klartext oder in verschlüsselter Form) herstellen.
- 2. **Sicherheitsziele (security objectives): Beispiel:** O.ArchivierungWahlgeheimnis
Die nach Feststellung des Wahlergebnisses noch auf dem Wahlserver gespeicherten Daten lassen keine Zuordnung zwischen dem Wähler und seiner Stimme (im Klartext oder in verschlüsselter Form) zu. Eine Zuordnung darf insbesondere nicht über die Reihenfolge und/oder den Zeitpunkt der Speicherung der Stimmdatensätze in der Ume geschehen.
- 3. Sammlung von **Sicherheitsanforderungen** (die, wenn möglich, schon in den CC-Dokumenten vordefiniert sein sollen)
- 4. **Erklärung (rationale)**, die u.a. nachweisen, dass die Sicherheitsziele und Bedrohungen sowie die Sicherheitsziele und die Sicherheitsanforderungen einander entsprechen
- 5. Optional eine EAL (fast immer!)

12.6.2 Erklärung - Rationale

- Sicherheitsziele müssen die Bedrohungen abdecken
- Tabelle: Sicherheitsziele versus Bedrohungen

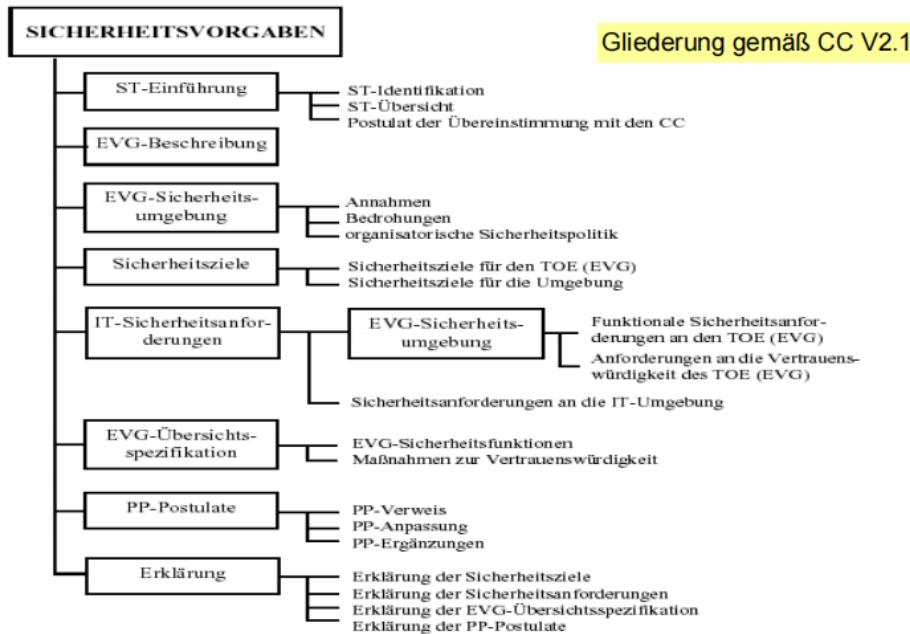
	T.UnbefugterWähler	T.Beweis	T.IntegritätNachricht	T.GeheimNachricht	T.AuthentizitätServer	T.ArchivierungIntegrität	T.ArchivierungWahlgeheimnis
O.StimmberechtigterWähler	X						
O.Beweis		X					
O.IntegritätNachricht			X				
O.Wahlgeheimnis				X			
O.GeheimNachricht				X			
O.AuthentizitätServer			X	X	X		
O.ArchivierungIntegrität						X	
O.ArchivierungWahlgeheimnis							X

- Erklärung, dass die Sicherheitsziele und die Sicherheitsanforderungen einander entsprechen

O.ArchivierungWahlgeheimnis Die Komponente **FPR_ANO.1** stellt sicher, daß nach Wahlende eine Zuordnung zwischen Wähler und seiner Stimme nicht mehr möglich ist. Durch die Verwendung der Komponente **FPR_UNL.1B** wird darüber hinaus sichergestellt, daß über die Reihenfolge und/oder den Zeitpunkt der Speicherung der Stimme in der Urne keine Zuordnung zwischen Wähler und Stimme möglich ist. Beides wird durch die Komponente **FDP_RIP.1A** unterstützt, die gewährleistet, daß keine zwischengespeicherten Stimmzettel erhalten bleiben, die das Wahlgeheimnis gefährden könnten.

12.6.3 Sicherheitsvorgaben

- **Sicherheitsvorgabe (security target)**: eine Menge von Sicherheitsanforderungen und Sicherheitsspezifikationen, die als Grundlage für die Prüfung und Bewertung eines angegebenen Evaluationsgegenstands dienen
- Eine Sicherheitsvorgabe kann eine Verfeinerung (Erweiterung) eines Schutzprofils sein
 - Es gibt aber auch Sicherheitsvorgaben, die nicht auf einem Schutzprofil beruhen
- **Implementierungs- und herstellerabhängigkeit**
- Für jeden EVG muss eine Sicherheitsvorgabe existieren
 - Gegen diese Sicherheitsvorgabe wird evaluiert



12.6.4 Vorgehensweise zur Evaluation

- Standardvorgehensweise
 - 1. Schutzprofil erstellen (falls ein solches noch nicht existiert)
 - 2. Schutzprofil auf Vollständigkeit, Konsistenz und Korrektheit hin evaluieren
 - 3. Sicherheitsvorgabe erstellen
 - 4. Gegebenfalls überprüfen, dass die Sicherheitsvorgabe eine korrekte Verfeinerung eines Schutzprofils ist
 - 5. Evaluationsgegenstand (EVG) gegen die Sicherheitsvorgabe evaluieren

12.6.5 Zertifizierte Produkte

- Liste mit zertifizierten Produkten: <http://www.commoncriteriaportal.org/products.html>
- Z.B.: MICARDO V4.0 R1.0 eHC v1.2 von Morpho Cards GmbH (evaluiert von SRC Security Research and Consulting GmbH)
- Es gibt zertifizierte Produkte mit der Evaluationsstufe EAL 5+, z.B.:
 - Infineon Smart Card IC (Security Controller) SLE66CX322P with RSA 2048/m1484 a24/ m1484a27 and m1484b14

- GemXplore Xpresso V3 Java Card Platform Embedded Software V3 (Core)
- Weltweite Anerkennung nur bis EAL4, **in Europa bis EAL7**

Beispiel: Protection Profile for electronic Health Cards (eHC)



https://www.bsi.bund.de/SharedDocs/Zertifikate_CC/PP/aktuell/PP_0020+V2+V3.html

12.6.6 Abschließende Bemerkungen und Bewertung

- **Vorteile** der CC:

- Weltweit anerkannter Kriterienkatalog
- Anwendbar auf verschiedene Klassen von IT-Produkten (nicht nur auf Betriebssysteme)
- Viele IT-Produkte sind bereits zertifiziert worden (aber viele aktuelle Main-Stream-Produkte auch nicht)
- Sicherheitsvorgabe/Schutzprofil nützliche Hilfsmittel zur Entwicklung eines sicheren Systems
- Hersteller können Zertifikate als Wettbewerbsvorteil nutzen:
 - * Z.B.: Ausschreibungen bezüglich Gesundheitskarten

- **Probleme** der CC:

- CC enthalten keine Evaluationsmethodik (Wie wird von den Prüfstellen evaluiert?)
- * Aber: zusätzliche Einführung der **Common Evaluation Methodology** (CEM)

- Aufwendiger und teurer Prozess, Rezertifizierung bei veränderter Version?
- Keine Usability-Aspekte
- Nicht jede Prüfstelle ist für jedes Produkt kompetent
- Abhängigkeit der Prüfstellen von den (nationalen) Behörden: Es wird so zertifiziert, wie es offiziell gewünscht wird.
- Teilweise bekannte Sicherheitsprobleme ignoriert:
 - * Beispiel: Schutzprofil für Krankenhausinformationssysteme setzt voraus, dass alle internen Benutzer ehrlich sind!

Sicherheitsmanagement

12.7 Informationssicherheitsmanagement

- Etablierung eines **Sicherheitsprozesses** in einer Organisation/Unternehmen
- Systematische Ermittlung und Umsetzung von Sicherheitsmaßnahmen
- Bewertung von Risiken:
 - Manche Risiken kann man tragen
- Auf Standards basierende Vorgehensweise für ein Sicherheitsmanagement gewünscht

12.8 BS 7799

BSI = British Standards Institute

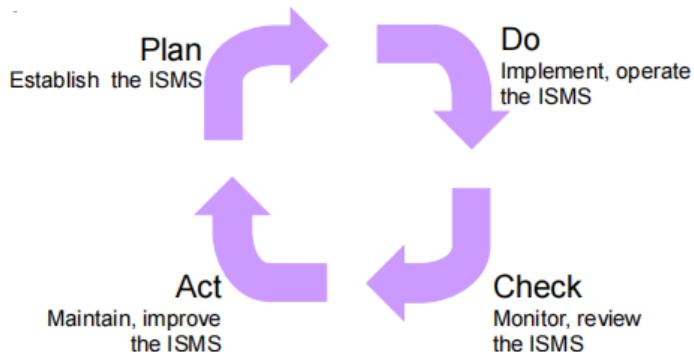
- Britisch Standard (BS) 7799
 - Ziel: Aufbau eines IT-Sicherheitsmanagements; Verankerung in der Organisation
 - **Management-orientiert**, nicht technisch
 - Keine detaillierten Umsetzungshinweise, sondern übergreifende Anforderungen
 - Zertifizierung gemäß BS 7799 - Teil 2 möglich (durch lizenzierte Auditoren)
- BS 7799 - Teil 1 ist Basis für ISO/IEC 17799
 - **Best Practice-Verfahren** und -Methoden
 - **Keine** Empfehlung für **konkrete** Sicherheitslösungen
 - **Keine** Hilfestellung zur **Bewertung** existierender Sicherheitsmaßnahmen
 - Keine Zertifizierung nach BS 17799 Teil 1 (Soll-Maßnahmen, nicht verpflichtend)

12.8.1 Themenbereiche BS 7799

- Zehn Themenbereiche:

- 1. Security policy - Provides management direction and support for information security
- 2. Organisation of assets and resources - To help you manage information security within the organisation
- 3. Asset classification and control - To help you identify your assets and appropriately protect them
- 4. Personnel security - To reduce the risks of human error, theft, fraud or misuse of facilities
- 5. Physical and environmental security - To prevent unauthorized access, damage and interference to business premises and information
- 6. Communications and operations management - To ensure the correct and secure operation of information processing facilities
- 7. Access control - To control access to information
- 8. Systems development and maintenance - To ensure that security is built into information systems
- 9. Business continuity management - To counteract interruptions to business activities and to protect critical business processes from the effects of major failures or disasters
- 10. Compliance - To avoid breaches of any criminal and civil law, statutory, regulatory or contractual obligations, and any security requirement

12.8.2 ISO 27001 / BS 7799



- Information Security Management System (ISMS)

- "ISO 9000 für Security"
- Plan-Do-Check-Act (aus QM)
- Zertifizierbar (teuer!)

12.9 Security Management nach BSI-Grundschutz

- **Ziel:** Etablierung eines Informationssicherheitsmanagements (ISMS) in einer Organisation wie z.B. Unternehmen, Behörde (IT-Verbund)
- **BSI** - Bundesamt für Sicherheit in der Informationstechnik
- **Grundschutzkompaktum** (früher: Grundschutzkataloge, Grundschutzhandbuch - GSHB)
 - Webseite des BSI zum Thema "IT-Grundschutz": <https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutz.html>
 - Online-Kurs IT-Grundschutz: <https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutz.html>
 - CD mit Grundschutzkatalogen (kostenlos) ; 3000 Seiten!
 - Zusammengefasste Darstellung u.a. im Buch C. Eckert: "IT-Sicherheit"
- GSTool: Hilfestellung bei der Anwendung von IT-Grundschutz (nur noch bis Version 4, Version 5 eingestellt, jetzt: Tools kommerzieller Anbieter)

12.9.1 BSI-Grundschutzkompaktum

- Sehr weit im Einsatz in Unternehmen mit niedrigem bzw. geringem Schutzbedarf
- Mehr als 50 **Bausteine** beschreiben verschiedene Aspekte der IT-Sicherheit (Server, PC, Firewall, E-Mail, WLAN, SAP, ...)
- Organisatorische, personelle, infrastrukturelle und technische Standardsicherheitsmaßnahmen
- Gefährdungs- und Maßnahmenkataloge

12.9.2 Schichtmodell

- Gruppierung der Bausteine in Schichten
 - 1. Übergreifende Aspekte: IT-Sicherheitsmanagement, Organisation, Personal, ...
 - 2. Infrastruktur: Gebäude, Verkabelung, Räume, ...
 - 3. IT-Systeme: Windows Clients, Unix Clients, Unix Server, TK-Anlage, ...
 - 4. Netze: Heterogene Netze, Remote Access, Netzmanagement, Modem
 - 5. IT-Anwendungen: E-Mail, Webserver, Datenbanken, ...

**Beispiel für einen Baustein aus den Grundschutzkatalogen: B 3.203
Laptop****Beschreibung**

Unter einem Laptop oder Notebook wird ein PC verstanden, der aufgrund seiner Bauart transportfreundlich ist und mobil genutzt werden kann. Ein Laptop hat eine kompaktere Bauform als Arbeitsplatzrechner und kann über Akkus zeitweise unabhängig von externer Stromversorgung betrieben werden. Er verfügt über eine Festplatte und meist auch über weitere Speichergeräte wie ein Disketten-, CD-ROM - oder DVD -Laufwerke sowie über Schnittstellen zur Kommunikation über verschiedene Medien (beispielsweise Modem, ISDN , LAN, USB, Firewire, WLAN). Laptops können mit allen üblichen Betriebssystemen wie Windows oder Linux betrieben werden. Daher ist zusätzlich der betriebssystemspezifische Client-Baustein zu betrachten. (...)

Die Einrichtungen zur Datenfernübertragung (über Modem, ISDN-Karte, etc.) werden hier nicht behandelt (siehe Baustein B 4.3). Für den Laptop wird vorausgesetzt, dass er innerhalb eines bestimmten Zeitraums nur von einem Benutzer gebraucht wird. Ein anschließender Benutzerwechsel wird berücksichtigt.

Gefährdungslage

Für den IT-Grundschutz eines Laptops werden folgende typische Gefährdungen angenommen:

Gefährdungslage für den Baustein (Auszug)

Höhere Gewalt

- G 1.2 Ausfall von IT-Systemen
- G 1.15 Beeinträchtigung durch wechselnde Einsatzumgebung

Organisatorische Mängel

- G 2.7 Unerlaubte Ausübung von Rechten**
- G 2.8 Unkontrollierter Einsatz von Betriebsmitteln
- G 2.16 Ungeordneter Benutzerwechsel bei tragbaren PCs

Menschliche Fehlhandlungen

- G 3.2 Fahrlässige Zerstörung von Gerät oder Daten
- G 3.3 Nichtbeachtung von Sicherheitsmaßnahmen
- G 3.6 Gefährdung durch Reinigungs- oder Fremdpersonal
- G 3.8 Fehlerhafte Nutzung von IT-Systemen
- G 3.38 Konfigurations- und Bedienungsfehler
- G 3.76 Fehler bei der Synchronisation mobiler Endgeräte

Technisches Versagen

- G 4.9 Ausfall der internen Stromversorgung
- G 4.13 Verlust gespeicherter Daten
- G 4.22 Software-Schwachstellen oder -Fehler
- G 4.52 Datenverlust bei mobilem Einsatz

Vorsätzliche Handlungen

- G 5.1 Manipulation oder Zerstörung von Geräten oder Zubehör

Beispiel: Gefährdungslage

G 2.7 Unerlaubte Ausübung von Rechten

Rechte wie Zutritts-, Zugangs- und Zugriffsberechtigungen werden als organisatorische Maßnahmen eingesetzt, um Informationen, Geschäftsprozesse und IT -Systeme vor unbefugtem Zugriff zu schützen. Werden solche Rechte an die falsche Person vergeben oder wird ein Recht unautorisiert ausgeübt, kann sich eine Vielzahl von Gefahren für die Vertraulichkeit und Integrität von Daten oder die Verfügbarkeit von Rechnerleistung ergeben.

Beispiele:

- Der Arbeitsvorbereiter, der keine Zutrittsberechtigung zum Datenträgerarchiv besitzt, entnimmt in Abwesenheit des Archivverwalters Magnetbänder, um Sicherungskopien einspielen zu können. Durch die unkontrollierte Entnahme wird das Bestandsverzeichnis des Datenträgerarchivs nicht aktualisiert, die Bänder sind für diesen Zeitraum nicht auffindbar.
- Ein Mitarbeiter ist erkrankt. Ein Zimmerkollege weiß aufgrund von Beobachtungen, wo dieser sein Passwort auf einem Merkzettel aufbewahrt und verschafft sich Zugang zum Rechner des anderen Mitarbeiters...

Maßnahmenempfehlungen für den Baustein (Auszug)

Planung und Konzeption

M 2.36 (B)	Geregelte Übergabe und Rücknahme eines tragbaren PC
M 2.218 (C)	Regelung der Mitnahme von Datenträgern und IT-Komponenten
M 2.309 (A)	Sicherheitsrichtlinien und Regelungen für die mobile IT-Nutzung
M 4.29 (Z)	Einsatz eines Verschlüsselungsproduktes für tragbare IT-Systeme

Beschaffung

M 2.310 (Z)	Geeignete Auswahl von Laptops
-------------	-------------------------------

Umsetzung

M 5.91 (A)	Einsatz von Personal Firewalls für Clients
M 5.121 (B)	Sichere Kommunikation von unterwegs
M 5.122 (A)	Sicherer Anschluss von Laptops an lokale Netze

Betrieb

M 1.33 (A)	Geeignete Aufbewahrung tragbarer IT-Systeme bei mobilem Einsatz
M 1.34 (A)	Geeignete Aufbewahrung tragbarer IT-Systeme im stationären Einsatz
M 1.35 (Z)	Sammelaufbewahrung tragbarer IT-Systeme
M 1.46 (Z)	Einsatz von Diebstahl-Sicherungen
M 4.3 (A)	Einsatz von Viren-Schutzprogrammen
M 4.27 (A)	Zugriffsschutz am Laptop
M 4.28 (Z)	Software-Reinstallation bei Benutzerwechsel eines Laptops...

Beispiel einer Maßnahmenbeschreibung (Auszug)

M 4.27 Zugriffsschutz am Laptop

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT

Verantwortlich für Umsetzung: Benutzer

Jeder Laptop sollte mit einem Zugriffsschutz versehen werden, der verhindert, dass dieser unberechtigt benutzt werden kann. Bei Laptops sollte als Minimalschutz, wenn kein anderer Sicherheitsmechanismus vorhanden ist, der BIOS -Bootschutz aktiviert werden, wenn dessen Nutzung möglich ist. Erst nach Eingabe des korrekten Bootpasswortes wird der Rechner dann hochgefahren. Die im Umgang mit Passwörtern zu beachtenden Regeln sind in M 2.11 Regelung des Passwortgebrauchs aufgeführt worden.

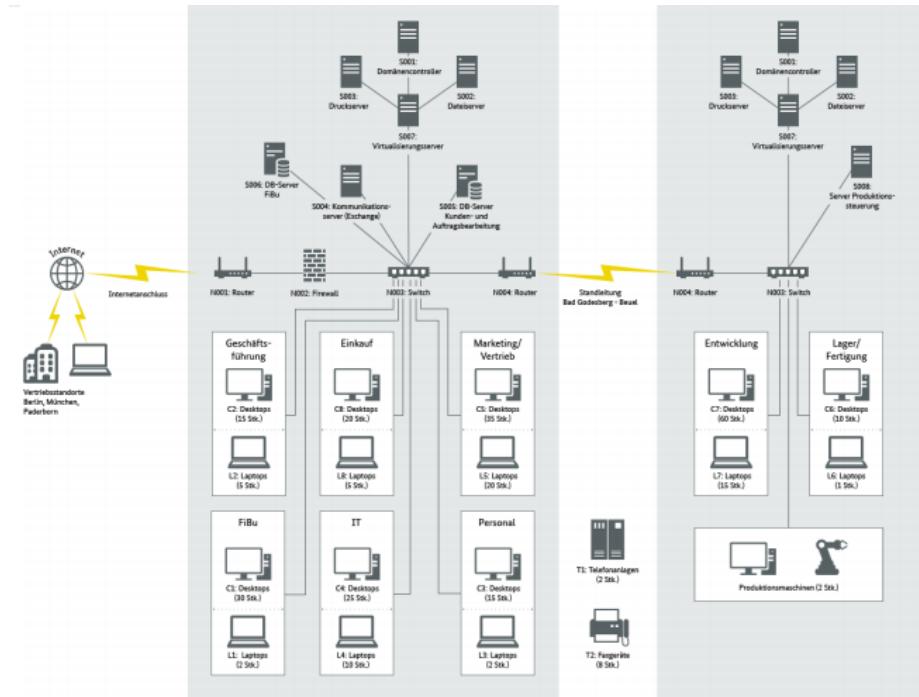
Außerdem bieten nahezu alle Betriebssysteme die Möglichkeit, Anmeldepasswörter einzurichten und diese mit geeigneten Restriktionen zu versehen (z. B. Mindestlänge, Lebensdauer, etc.). Da diese Bordmittel nur eine begrenzte Sicherheit bieten, empfiehlt es sich bei Laptops, auf denen sich schnell große Mengen sensibler Daten sammeln, zusätzliche Sicherheitshard- oder -software einzusetzen. Dazu gehören beispielsweise ...

12.9.3 Vorgehensweise nach IT-Grundschutz

- Durchführung der folgenden Schritte
 - 1. Infrastrukturanalyse
 - 2. Schutzbedarfsfeststellung
 - 3. Modellierung der Maßnahmen
 - 4. Basis-Sicherheitscheck
 - 5. Ergänzende Risikoanalyse (falls erforderlich)
 - 6. (Planung der) Realisierung
- Wiederholung der Schritte in regelmäßigen Abständen: **Prozess**; kompatibel zu ISO 27001?

12.9.4 Infrastrukturanalyse

- Festlegung des IT-Verbundes (z.B. Unternehmen, Abteilung)
- Erstellung von (vereinfachten) Netzplänen des IT-Verbundes
- Aufnahme der Anwendungen, IT-Systeme (Clients, Server, Netzkomponenten, TK-Anlagen, ...)



12.9.5 Schutzbedarfsfeststellung

Schutzbedarfskategorien	
Niedrig - mittel	die Schadensauswirkungen sind begrenzt und überschaubar
Hoch	die Schadensauswirkungen können beträchtlich sein
Sehr hoch	die Schadensauswirkungen können ein existentiell bedrohliches, katastrophales Ausmaß erreichen

- Einordnung der Anwendungen, IT-Systeme und Räume gemäß den Schutzbedarfskategorien
- Und zwar für die Schutzziele Vertraulichkeit, Integrität und Verfügbarkeit
- Einordnung mit Hilfe des zu zertifizierenden Unternehmens

12.9.6 Modellierung nach IT-Grundschatz

- Auswahl der geeigneten Bausteine der Grundschatzkataloge auf Basis der Ergebnisse der IT-Infrastrukturanalyse und der Schutzbedarfsfeststellung

- Wie bekommt man die geeigneten Bausteine? Intellektuelle Leistung?
- Unterstützung durch Werkzeuge (keine neue Version des GS-Tools mehr!)
 - Ergebnisse sollte man schon noch überprüfen

12.9.7 Basis-Sicherheitscheck

- Idee: Vergleiche die bereits umgesetzten Maßnahmen mit denen, welche die Modellierung ergeben hat
- Soll-Ist-Vergleich
- Geht nur für normalen Schutzbedarf (niedrig-mittel)
 - → ergänzende Risikoanalyse

12.9.8 Ergänzende Risikoanalyse

- Was ist, wenn man IT-System bzw. eine Anwendung hat, für die kein Baustein in den Grundschutzkatalogen existiert?
 - Beispiele: Krankenhausinformationssystem, Bankanwendungen, Fabriksteuerung
- Was ist, wenn man einen hohen oder sogar einen sehr hohen Schutzbedarf hat?
 - Verwaltung von Patientendaten
 - Militärische Daten
- ⇒ Durchführung einer ergänzenden Risikoanalyse
 - BSI macht Vorschlag zu einer vereinfachten, qualitativen Risikoanalyse

12.9.9 Planung der Realisierung

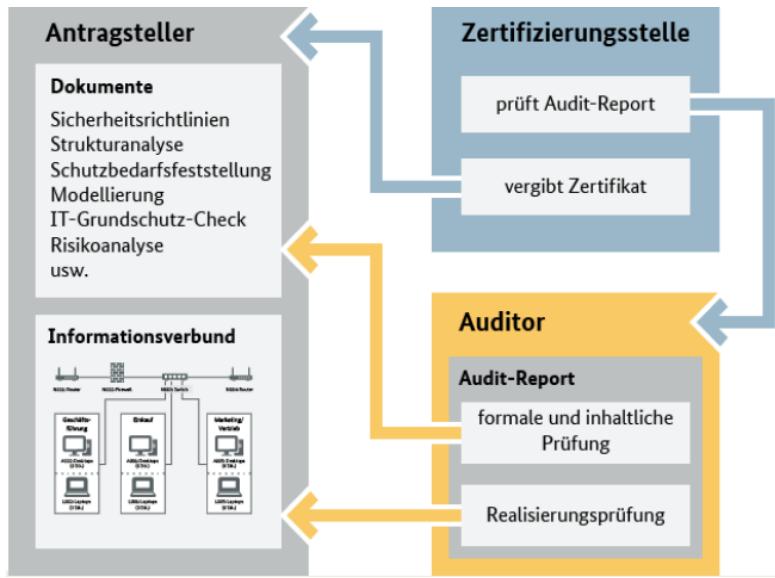
- Nach Bestimmung aller Maßnahmen: Planung der Realisierung
- Umsetzen fehlender Maßnahmen
- Priorisierung der Maßnahmen
- Abschätzung der Kosten der Maßnahmen
- Bestätigung durch Zertifikat möglich, wenn lizenzierte Auditoren den IT-Verbund überprüfen (stichprobenartig)

12.9.10 Zusammenfassung: Vorgehensweise nach GS-Grundschutz



12.9.11 Grundschatz-Zertifizierung

- Zertifizierung des ISMS für einen IT-Verbund:
 - ISO 27001-Zertifikat auf Basis von IT-Grundschutz
- Evaluierung durch lizenzierten Auditor
- Vergabe des Zertifikates durch den BSI



12.9.12 Einordnung

- Pro:

- Für Standard IT-Systeme bekommt man relativ schnell geeignete und **konkrete** Sicherheitsmaßnahmen
- Keine Neuerfindung des Rades: Standardvorgehensweise mit ausgereifter Methodik
- Möglichkeit der Zertifizierung

- Contra:

- Gesamter Prozess etwas schwerfällig
- „Box-Ticking“-Mentalität
- Keine offizielle internationale Anerkennung, aber laut BSI kompatibel zu ISO 27001: ISO/IEC 27001-Zertifikat auf Basis von IT-Grundsatz

12.10 Vergleich CC/BSI GS/BS 7799

- CC:

- Abstrakt
- Auf verschiedene **IT-Produkte** anwendbar
- Nur Technik, keine organisatorischen Maßnahmen
- Internationaler Standard

- BSI GS
 - Konkrete Maßnahmen
 - Auf ganze **IT-Verbünde**, nicht aber auf einzelne IT-Produkte anwendbar
 - Sowohl technische als auch organisatorische Maßnahmen
 - Kein internationaler Standard
- BS 7799
 - Auf ganze **IT-Verbünde**, nicht aber auf einzelne IT-Produkte anwendbar
 - Nur organisatorische/prozessorientierte Maßnahmen
 - Internationaler Standard