

Data Authentication

Prof. Bart Preneel
COSIC – KU Leuven - Belgium
Firstname.Lastname(at)esat.kuleuven.be
<http://homes.esat.kuleuven.be/~preneel>
October 2023

1

1

Goals

- Definitions of hash functions, MAC algorithms and digital signature schemes
- How are hash functions, MAC algorithms and digital signature algorithms constructed
- Why collision resistance is important for digital signatures
- Authenticated encryption
- Secret sharing

2

2

Outline

- introduction
- symmetric authentication
 - hash functions (MDC)
 - MAC algorithms
 - authenticated encryption
- digital signatures
 - RSA
 - ElGamal
 - DSA
- cryptographic protocols and secret sharing

3

3

Symmetric cryptology: data authentication

- the problem
- hash functions without a key
 - MDC: Manipulation Detection Code
- hash functions with a secret key
 - MAC: Message Authentication Code

4

4

Data authentication: the problem

- Bob (recipient) wants to know:
 - the **source** of the information (data origin)
 - that the information has not been **modified**
 - (optionally) **timeliness** and **sequence**
 - (optionally) **the destination**
- problem of replay of messages needs to be addressed at higher layer (e.g. transaction counter in financial systems)
- specific challenges: very long streams, versioning systems, noisy data,....

5

5

Data authentication versus encryption

- encryption provides confidentiality of content
 - prevents Eve from learning information on the cleartext/plaintext
 - but does not protect against modifications (active eavesdropping)
- encryption does not hide
 - length of the message (unless padding)
 - who is talking to whom (traffic analysis)

there are no applications that require encryption **without** data authentication (but this can still be found in legacy applications with as excuse performance)

6

6

Data authentication: hash function

(sometimes called MDC)

- MDC (manipulation detection code)
- protect short hash value rather than long text

This is an input to a cryptographic hash function. The input is a very long string, that is reduced by the hash function to a string of fixed length. There are additional security conditions: it should be very hard to find an input hashing to a given value (a preimage) or to find two colliding inputs (a collision).

h

E835FD4128A198FB3CA345932

(MD5)
(SHA-1)
RIPEMD-160
SHA-2: SHA-256,
SHA-512
SHA-3

7

How to use a hash function in practice

- hash functions move the problem of protecting large strings to that of protecting short strings
- hash the message and
 - either** authenticate the short hash result
 - via a second channel (e.g. read it over the phone)
 - via a digital signature (cf. infra)
 - or** encrypt it using a secret key
 - a MAC algorithm is a better option in this case

8

Hash function definition

- description is public
- takes input of arbitrary size and reduce it to a n-bit result (of fixed length)
- efficient to compute
- preimage resistance: for given y, hard to find input x such that $h(x) = y$ (2^n operations)
- 2nd preimage resistance: hard to find $x' \neq x$ such that $h(x') = h(x)$ (2^n operations)
- collision resistance: hard to find (x, x') with $x' \neq x$ such that $h(x') = h(x)$ ($2^{n/2}$ operations)

9

MDC Security requirements (n-bit result)

preimage

?

h

$h(x)$

2^n

2nd preimage

$x \neq$?

h

h

$h(x) = h(x')$

2^n

collision

?

?

h

h

$=$

$2^{n/2}$

10

Preimage resistance

preimage

?

h

$h(x)$

2^n

- in a password file, one does not store (username, password)
- but (username, hash(password))
- this is sufficient to verify a password
- an attacker with access to the password file has to find a preimage

11

Second preimage resistance

2nd preimage

$x \neq$?

h

h

$h(x) = h(x')$

2^n

- transmit x over a fast but insecure channel
- transmit $h(x)$ over a slow but authenticated channel (e.g., read it over the phone)
- an attacker has access to x but he can only fool the recipient if he finds a second preimage of x
- another example:
 - compute a hash of the files on a USB stick before you lend it to your friend
 - you can store the hash on your laptop or write it down

12

Collision resistance

- in many cryptographic protocols, Alice wants to commit to a value x without revealing it
- Alice picks a secret random string r and sends $y = h(x \parallel r)$ to Bob
- in a later phase of the protocol, Alice reveals x and r to Bob and he checks that y is correct
- if Alice can find a **collision**, that is (x,r) and (x',r') with $x' \neq x$ she can cheat
- if Bob can find a **preimage**, he can learn x and cheat

collision

$x \neq x'$

$h(x) = h(x')$

$2^{n/2}$

13

The birthday paradox

- collision: birthday paradox (or $\sqrt{}$ attack) [Yuval'79]
 - r variations on a correct message
 - r variations on a fraudulent message
- probability of a match is 39% if $r = \sqrt{2^n} = 2^{n/2}$
- infeasible $n \geq 160$ but at least 256 bits necessary for long term security
- efficient implementations: parallel and little memory (see next slide)

14

Brute force attacks in practice (2023)

- (2nd) preimage search
 - $n = 128$: 10 B\$ for 8 billion years
 - $n = 128$: 10 M\$ for 7 years **if one can attack 2^{40} targets in parallel (success probability grows linearly with the number of targets)**
- parallel collision search
 - $n = 128$: 10 M\$ for 15 seconds (or 1 year on 18 GPUs)
 - $n = 160$: 10 M\$ for 11 days
 - need 256-bit result for long term security (25 years or more)

15

Hash function: iterated structure

- apply unambiguous padding rule
- append the input length in bits
- split into blocks of fixed length x_i
- hash them block by block with a compression function f
- output transformation g at the end

16

Hash functions based on a block cipher with n -bit block and key

- 12 secure schemes of rate 1: collision $2^{n/2}$, (2nd) preimage 2^n
- not sufficient for DES, 3-DES or even AES

Matyas-Meyer-Oseas

Davies-Meyer

Miyaguchi-Preneel

17

Permutation (π) based: sponge

absorb

squeeze

if result has n bits, H_1 has r bits (rate), H_2 has c bits (capacity) and the permutation π is "ideal"

collisions	$\min(2^{c/2}, 2^{n/2})$
2 nd preimage	$\min(2^{c/2}, 2^n)$
preimage	$\min(2^c, 2^n)$

18

Many broken hash functions

- X.509 Annex D (1988)
- MD2 (1989)
- Snefru (1990)
- MD4 (1990)
- MD5 (1991, widely used until 2009) – collision attack found in 2004 in time $2^{39} \ll 2^{64}$
- SHA-0 (1993, withdrawn standard)
- SHA-1 (1995, widely used until 2016) – collision attack found in 2005 in time $2^{69} \ll 2^{80}$
- ...
- hash function crisis: 2004-2005

19

19

The complexity of collision attacks

Brute force: 4 million PCs or US\$ 100K hardware (1 year)

Year	MD4	MD5	SHA-0	SHA-1	Brute force
1992	60	60	80	80	60
1994	60	60	80	80	60
1996	20	60	80	80	60
1998	20	60	80	80	60
2000	20	60	80	80	60
2002	20	60	80	80	60
2004	5	40	50	70	65
2006	5	30	40	65	68
2008	5	20	30	60	70

20

20

Hash functions standards (ISO/IEC 10118-3)

- **SHA-1** (1995): NIST FIPS 180-1 (US) (160 bits) – avoid
 - shortcut collision attack published in Aug. '05: 2^{69} steps
 - collision found in Feb. '17: 2^{63} steps
- **RIPEMD-160** (1996): BSI (D) and KU Leuven (B) (160 bits)
- **Whirlpool** (2000): Univ. Sao Paolo (Br) and KU Leuven (B) (256 bits)
- **SHA-2** (2004): set of algorithms SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/256: NIST FIPS 180-2 (US)
- **SHA-3** (2015): set of algorithms based on sponge (Keccak): SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE-128 and SHAKE-256: NIST FIPS 202 (US)

21

21

Data authentication:
Message Authentication Code (MAC)

- Replace protection of authenticity of (long) message by protection of secrecy of (short) key
- Add MAC to the plaintext

- CBC-MAC
- HMAC
- GMAC

This is an input to a MAC algorithm. The input is a very long string, that is reduced by the hash function to a string of fixed length. There are additional security conditions: it should be very hard for someone who does not know the secret key to compute the hash function on a new input.

MAC

7E6FD7198A198FB3C

22

22

MAC algorithms

Clear text

MAC

Clear text

Modify

Clear text

VERIFY

Clear text

23

23

MAC: definition

1. description of h public
2. x arbitrary length \Rightarrow fixed length m (32 . . . 160 bits)
3. computation of $h_K(x)$ “easy” given x and K
4. computation of $h_K(x)$ “hard” given only x , even if a large number of pairs $\{x_i, h_K(x_i)\}$ is known

calculation of $h_K(x)$ without knowledge of secret key: *forgery* (verifiable or not verifiable)

Pseudo-Random function (PRF): the output of all widely used MAC algorithms cannot be distinguished from the output of a random function (stronger requirement than unpredictability)

24

24

MAC: generic attacks

key length = k bits (56..168) and MAC length = m bits (32..96)

1. guess MAC: ± same as for hash function

– on-line verification only

– success probability max (1/2^m, 1/2^k)

2. exhaustive key search: ± same as for block cipher

– # x, h_K(x) pairs ≈ k/m

– # attempts ≈ 2^{k-1}

3. birthday paradox:

internal memory n bits; result m bits

forgery after 2^{n/2} known and ≤ 2^{n-m} chosen texts

25

25

MAC algorithms

• banking: CBC-MAC based on triple-DES

• internet: HMAC and CBC-MAC based on AES

• information theoretic secure MAC algorithms (authentication codes): GMAC/UMAC/Poly1305

– highly efficient but some have rather long keys

– part of the key refreshed per message

– GMAC standard is not very robust (implementation errors can be very costly)

26

26

ANSI Retail MAC based on DES

27

27

CBC-MAC based on AES (LMAC)

28

28

MAC based on a hash function

• secret prefix: h(K₁||x)

prepend length to avoid that one can compute h(K₁||x||y) from h(K₁||x) without knowing K₁

• secret suffix: h(x||K₂)

off-line collision attacks on h

• envelope: h(K₁||x||K₂)

risky: forgery leads to key recovery

• better variant than all of the above: HMAC:

▪ h_K(X) = h(K₂ || h(K₁ || x))

▪ not secure with MD4/MD5

29

29

GMAC: polynomial authentication code (NIST SP 800-38D 2007 + 3GSM)

keys $K_1, K_2 \in GF(2^{128})$

input x: x_1, x_2, \dots, x_t , with $x_i \in GF(2^{128})$

$$g(x) = K_1 + \sum_{i=1}^t x_i \cdot (K_2)^i$$

in practice: compute $K_1 = \text{AES}_{K_2}(n)$ (CTR mode)

properties:

• fast in software and hardware (support from Intel/AMD)

• not very robust w.r.t. nonce reuse, truncation, MAC verifications, due to reuse of K_2 (not in 3G/4G/5G!)

• versions over GF(p) (e.g. Poly1305-AES) seem slightly more robust

30

30

Authenticated encryption

- Encryption only is never sufficient
- Encryption often vulnerable to chosen ciphertext attacks (e.g. padding verification in CBC decryption)

Generic composition [BN'00][NRS'14]

- Encrypt-then-MAC with 2 independent keys
 - IPsec, TLS 1.2, 1.3
- MAC-then-Encrypt with 2 independent keys
 - TLS 1.1 and older, 802.11i WiFi
- MAC-and-Encrypt with 2 independent keys

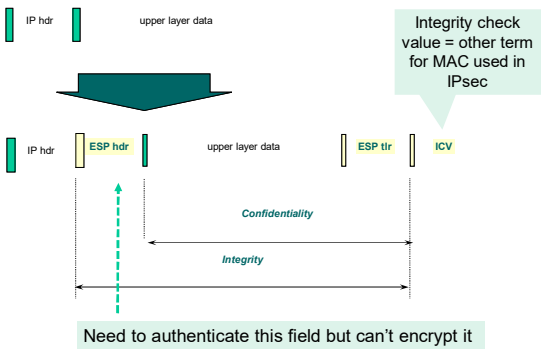
Design “from scratch”

- Integrated authenticated encryption schemes: combined operation with 1 key: see next slide

Authenticated Encryption: properties wish list

- Associated data
- Parallelizable
- Online for encryption
- Security reduction
- Resistance to nonce reuse
- Incremental tags
- Fragmentation
- No release of unverified plaintext
- Flexible implementation sizes
- Performance: speed/size
- Secure implementations: constant time/power analysis/EM analysis/fault attacks...
- Not patented

IPsec - ESP Transport mode



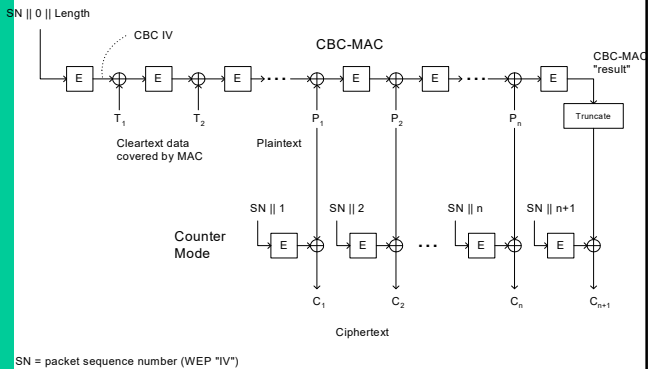
Long history to achieve AE based on block ciphers

- Ad hoc schemes
 - Jueneman ('80s)
 - PCBC (Kerberos)
 - iaPCBC
 - WEP for WiFi 802.11
- 2nd generation
 - CCM
 - **GCM**
 - EAX
 - CWC
- 1st schemes with proofs
 - RPC [Katz-Yung'00]
 - IAPM [Jutla'01]
 - XECB and XCBC [GD'01]
 - OCB1, OCB2, **OCB3** [RBBK'01]
- 3rd generation
 - **ChaCha20-Poly1305**
 - GCM-SIV
 - BTM
 - McOE-G
 - **Argus**

AE: block cipher based

	# passes	//	Online (encr.)	Nonce Misuse	Patented
IAPM	1	✓	✓		✓
XECB	1	✓	✓		✓
OCB	1	✓	✓		✓
CCM	2				
GCM	1*	✓	✓		
EAX	2		✓		
CWC	2	✓	✓		
SIV	2			✓	
BTM	1	✓		✓	
McOE-G	1*		✓	✓	

Example: CCM: CTR + CBC-MAC



OCB = masked ECB encryption + encrypted simple checksum

37

37

Permutation (f) based authenticated encryption

Simple and elegant
Source of figure: Keccak team [Bertoni-Daemen-Peeters-Van Assche]

38

38

Caesar competition for Authenticated Encryption

2013-2019 <https://competitions.cr.yp.to/caesar.html>

	Name	Designers
Lightweight	Ascon	C. Dobraunig, M. Eichlseder, F. Mendel, M. Schl��fer
	ACORN	H. Wu
High speed	Aegis	H. Wu, B. Preneel
	OCB	T. Krovetz, P. Rogaway
Robust	COLM	J. Jean, I. Nikoli��, T. Peyrin, Y. Seurin
	AES-COPA	E. Andreeva, A. Bogdanov, N. Datta, A. Luykx, B. Mennink, M. Nandi, E. Tischhauser, K. Yasuda

Selected from 52 submissions – a 5-year effort
OCB2 has been broken at Crypto 2019 (bug in security proof) – but OCB3 is still ok

39

39

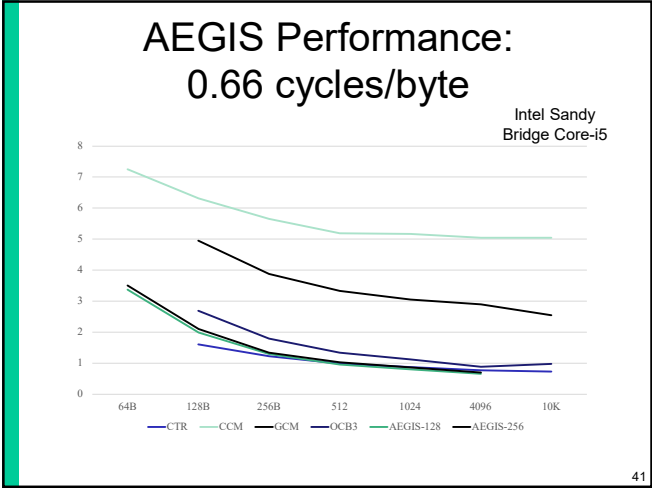
NIST Competition for Lightweight Cryptography

2018-2023 <https://csrc.nist.gov/projects/lightweight-cryptography>

Round 1: 56 candidates (selected from 57 submissions)
Round 2: 32 candidates
Round 3 (19 March 2021): 10 finalists
ASCON, Elephant, GIFT-COFB, Grain128-AEAD, ISAP, Photon-Beetle, Romulus, Sparkle, TinyJambu, Xoodyak
Winner (Feb. 2023): ASCON

40

40



41

Outline

- introduction
- symmetric authentication
 - hash functions (MDC)
 - MAC algorithms
 - authenticated encryption
- digital signatures
 - RSA
 - ElGamal
 - DSA
- cryptographic protocols and secret sharing


42

42

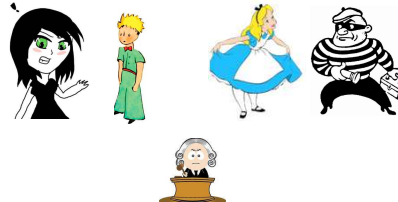
Digital signature schemes

- MAC: requires that Alice and Bob trust each other
- data authentication without shared secret keys
 - works even if Alice and Bob do not trust each other
 - sender and receiver have different capability
 - third party can resolve dispute between sender and receiver

MAC



Digital signature



43

43

Digital signature schemes

- RSA
 - with message recovery (only in textbooks)
 - with appendix
- ElGamal
- DSA
- one can also construct digital signatures based on one-way functions (e.g. SHA-2 or AES)
 - these schemes require large public keys (not explained in this course)
 - considered for post-quantum crypto

44

44

RSA signatures for short messages (message recovery)

signature: $s = m^d \bmod n$

verification: $m = s^e \bmod n$

insecure!

- for small messages there is no reduction and anyone can compute the e^{th} root
 - example: e^{th} root of 1 is 1; e^{th} root of -1 is -1
- everybody can choose a random value s^* and claim that Alice has signed $m^* = s^{*e} \bmod n$ (indeed, e and n are public)
- homomorphism: $\text{RSA}(m_1 \cdot m_2 \bmod n) = \text{RSA}(m_1) \cdot \text{RSA}(m_2)$
 - after two signatures, one can forge a third one

45

45

RSA signatures for short messages (message recovery)

signature:
encode m into \underline{m}
 $s' = \underline{m}^d \bmod n$

verification:
 $\underline{m} = (s')^e \bmod n$
check redundancy in \underline{m}

advantage: low communication overhead because one can recover the message from the signature

disadvantage: only works for short messages (60-80 bytes)

encoding:
ISO 9796-1 was industry standard, but it has been broken
alternative: $00 \parallel 01 \parallel \text{FF} \parallel m \parallel h(m) \parallel h(0 \parallel h(m)) \parallel h(1 \parallel h(m)) \parallel 6$
or with SHA-3 as eXtensible Output Function: $00 \parallel 01 \parallel \text{FF} \parallel m \parallel h(m) \parallel 6$

46

46

RSA signatures for long messages (with appendix)

signature: $s = h(m)^d \bmod n$

verification: $h(m) = s^e \bmod n$

This is an input to a cryptographic hash function. The input is a very long string, that is reduced by the hash function to a string of fixed length.

h

Private key

SIGN

$s=4F80DFD41A198FB3CA3459$

This is an input to a cryptographic hash function. The input is a very long string, that is reduced by the hash function to a string of fixed length.

h

Public key

VERIFY

YES/NO

$s=4F80DFD41A198FB3CA3459$

47

47

Collision resistance revisited:
hash functions used in digital signature schemes

- hacker Alice prepares two versions of a software driver for the O/S company Bob
 - x is correct code
 - x' contains a backdoor that gives Alice access to the machine
- Alice submits x for inspection to Bob
- if Bob is satisfied, he digitally signs $h(x)$ with his private key
- Alice now distributes x' to users of the O/S; these users verify the signature on x with Bob's public key
- however, this signature also works for x' , as $h(x) = h(x')$!

collision

$x \neq x'$

h h

$h(x) = h(x')$

$2^{n/2}$

48

48

RSA signatures for long messages (with appendix)

- formatting in practice: PKCS#1 v1.5

00 01 ff ff ff ff ... ff ff ff 00 HashID h(m)

 - it is not possible to formally prove security
- better solutions exist: PKCS#1 v2.1 – RSA PSS
 - rarely deployed
- EMV (credit card standard) uses hybrid form: partial message recovery
 - security also limited but risk is very small in EMV context

49

49

ElGamal signatures (1985)

key generation: (x,y)

- general parameters: (safe) prime p and generator α
- private key: x ($1 < x < p-1$)
- public key: $y = \alpha^x \bmod p$

signature generation: (r,s)

- generate random k ($1 < k < p-1$) with $\gcd(k, p-1) = 1$
- $r = \alpha^k \bmod p$ (k and r are temporary private/public key)
- s the solution of $\alpha^{h(m)} = \alpha^{x \cdot r} \cdot \alpha^{k \cdot s} \bmod p$
- thus $h(m) = (x \cdot r + k \cdot s) \bmod p-1$ or $s \equiv (h(m) - x \cdot r) \cdot k^{-1} \bmod p-1$ (inverse exists as $\gcd(k, p-1) = 1$)

signature verification

- given m, r, s , check whether $1 < r < p$ and $\alpha^{h(m)} \equiv y^r \cdot r^s \bmod p$
- this holds because $y = \alpha^x$ and $r = \alpha^k$

50

50

ElGamal signatures: properties

- need secure random number generation: insecure if k is reused
 - mistake in Sony PSP3 – k never changed (2011): resulted in leakage of signing key
 - exercise: explain why
- insecure if DLOG is easy
- forgery: solve the verification equation – hard?
 - choose r , find s : $r^s \cdot y^r \equiv m \bmod p$
 - choose s , find r : solve the equation for r (seems even harder)
- if α or p are chosen by a malicious party, forgery is easy [Bleichenbacher'96]

51

51

DSA signatures (1994) – FIPS 186

key generation (x,y)

- general parameters: (safe) prime p
- 256-bit prime q , divisor of $p-1$;
- basis α , element of order q , or $\alpha^q = 1 \bmod p$
- private key: x ($1 < x < p-1$)
- public key: $y = \alpha^x \bmod p$

signature generation: (r,s)

- generate random k ($1 < k < q-1$) with $\gcd(k, q) = 1$
- $r = (\alpha^k \bmod p) \bmod q$
- s the solution of $s \equiv (h(m) + x \cdot r) \cdot k^{-1} \bmod q$ (inverse exists)

signature verification

- compute $w = s^{-1} \bmod q$
- verify whether $r = (\alpha^{h(m)w} \cdot y^{r \cdot w} \bmod p) \bmod q$

52

52

DSA versus ElGamal

- DSA: shorter signatures (512 bits rather than 2048 bits)
- DSA: specifies algorithms to design all the random numbers based on SHA-1
 - this algorithm was patched after an attack [Bleichenbacher01]
- ElGamal requires some extra checks to validate parameters
- ElGamal can be used for encryption as well
 - in 1994, the US government saw this as a disadvantage of ElGamal

ElGamal no longer used in practice

53

53

DSA versus RSA

- discrete log/factoring: \pm equivalent
- advantage DSA: some operations mod q (256 bits!), shorter signature (512 bits), r can be precomputed
- disadvantage DSA: cannot be used for encryption!
- disadvantage DSA: verification of signature slower than signing
 - typically a signature is computed once but verified multiple times
- RSA patent expired in 2000; DSA in 2011
- DSA has elliptic curve variant (ECDSA) which has shorter keys (256 or 512 bits)

54

54

Postquantum signatures

- Lattice-based: Dilithium, Falcon
- Hash-based: Sphincs+, LM, XMSS
- Large signatures and public keys
- More schemes to arrive around 2027

55

55

Public key versus symmetric key

- reduce protection of information to protection of authenticity of public keys
 - confidentiality without establishing secret keys = extremely useful in an **open** environment
 - public key avoids the need of a central database with all symmetric keys of the users
- data authentication without shared secret keys: **digital signature**
 - sender and receiver have different capability
 - third party can resolve dispute between sender and receiver
- large performance disadvantage leads to use of hybrid schemes
 - hash + sign, encrypt symmetric session key with public key encryption
- widely used schemes all depend on small set of mathematical problems from algebraic number theory

56

56

Outline

- introduction
- symmetric authentication
 - hash functions (MDC)
 - MAC algorithms
- digital signatures
 - RSA
 - ElGamal
 - DSA
- cryptographic protocols and secret sharing

57

57

Cryptographic protocols

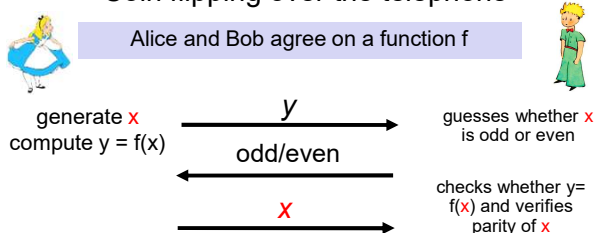
- **coin flipping over the phone**
- **sharing a secret between several persons**
- playing poker over a telephone
- simultaneous exchange of information
- verification of nuclear test-ban treaties and subliminal channels
- electronic money
- anonymous communications
- the millionaires problem
- online auctions
- distance bounding
- ...

58

58

Coin flipping over the telephone

Alice and Bob agree on a function f



Bob wins if he guessed the parity of x correctly

- it should be hard (for Bob) to find out from $f(x)$ whether x is odd or even
- it should be hard (for Alice) to find an even x and an odd x' with $f(x) = f(x')$

59

59

Secret sharing

information theoretic security - no assumptions
is it possible to divide secret information (giving access to a vault) over n persons such that:

- each subset of t or more persons can reconstruct the information
- each subset of $< t$ persons has no information whatsoever on the secret information ...

application: storage of very sensitive keys

60

60

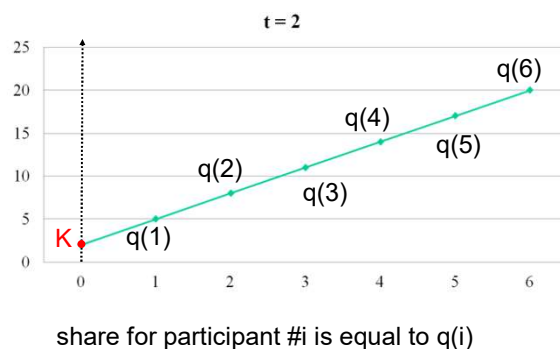
Secret sharing: $t = n$

- secret information K is a v -bit string
- distribution phase
 - $n - 1$ participants get as share an arbitrary v -bit string S_i
 - n th participant gets $S_n = K \oplus_{i=1}^{n-1} S_i$
- reconstruction phase
 - pool all shares and compute $K = \oplus_{i=1}^n S_i$
- no subset of $< n$ participants has any information on K
- but if 1 participant disappears, one can never recover the secret

61

61

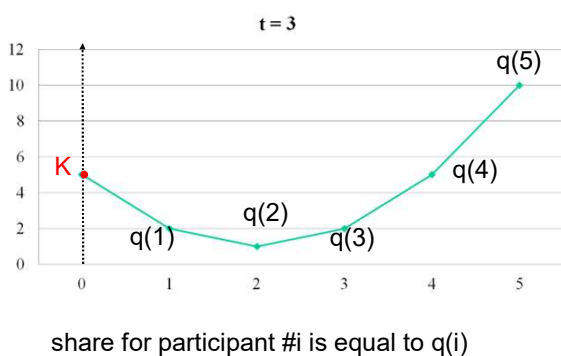
Secret sharing: $t = 2$ and $n = 6$



62

62

Secret sharing: $t = 3$ and $n = 5$



63

63

Secret sharing: $t < n$ - polynomial interpolation [Shamir'78]

- secret information K is a v -bit string
- choose a prime $p \geq \max(2^v, n+1)$
- distribution phase
 - choose $t - 1$ random integers a_1, \dots, a_{t-1} from $[0, p - 1]$
 - $q(x) = K + a_1x + a_2x^2 + \dots + a_{t-1}x^{t-1}$
 - participant i gets share $q(i)$
 - all operations mod p
- reconstruction phase
 - t participants contribute their shares
 - Lagrange interpolation to recover the polynomial
 - find intersection with origin: $q(0)=K$

64

64