UNIK4250 Security in Distributed Systems
UNIK University Graduate Center
Spring 2012

Lecture 3

Public-key Cryptography and Message Authentication

Leif Nilsen
Ed 1.0

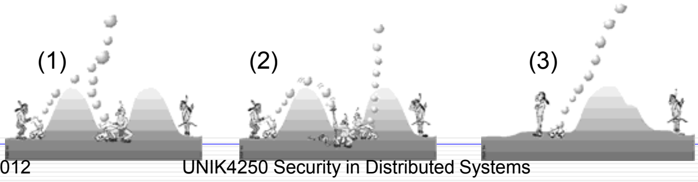
UNIK

Outline

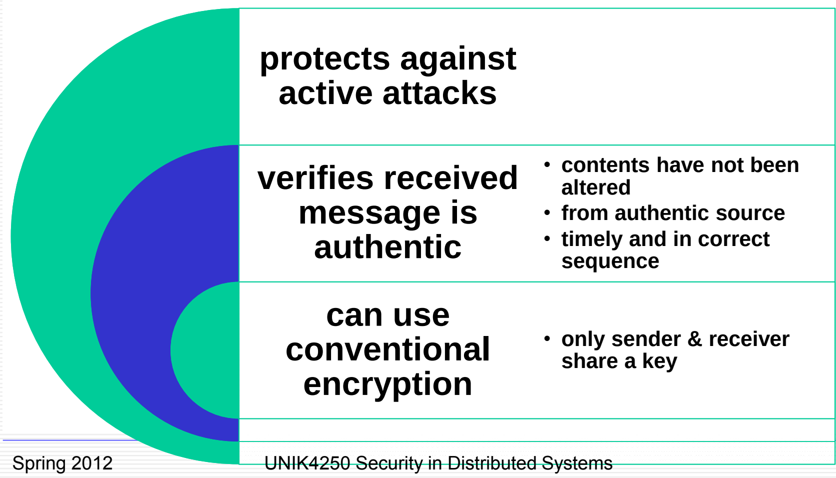
- Message authentication
- Message Authentication Codes (MAC)
- Cryptographic hash functions
- Asymmetric crypto
- Digital signatures
- Elliptic Curve Cryptography

Message Authentication

- message authentication is concerned with:
  - protecting the integrity of a message (1)
  - validating identity of originator (2)
  - non-repudiation of origin (dispute resolution) (3)
- the three alternative functions used:
  - hash function
  - message encryption
  - message authentication code (MAC)



Message Authentication



Message Authentication Codes

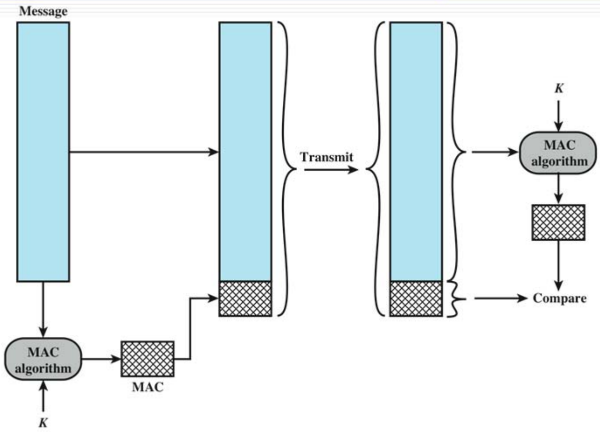


Figure 2.4 Message Authentication Using a Message Authentication Code (MAC). The MAC is a function of an input message and a secret key.

The Aftenposten puzzle

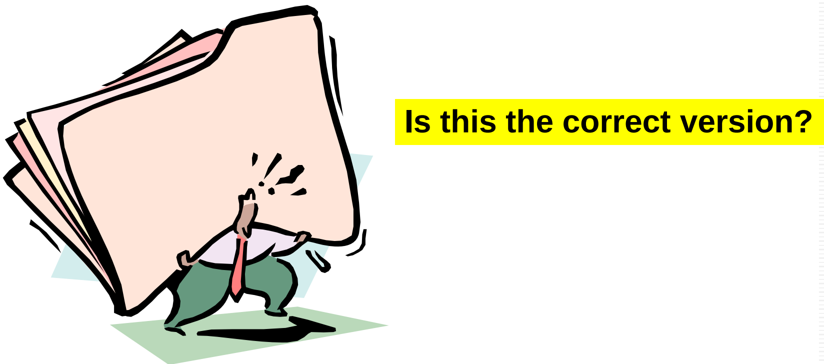
Aftenposten
Mandag 28. september 2009

▶ Kunngjøringer

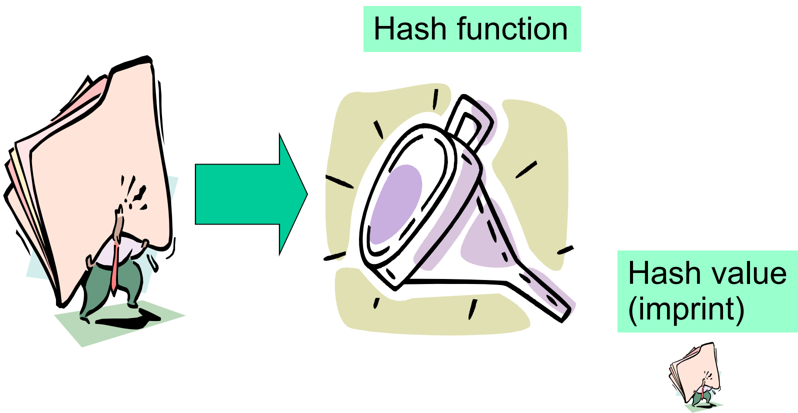
Conax As presenterer:
SHA256(x) =
0x2d12e807bc4880db21a2981
bdfc8b5a0de3786036f0519d8
dfe8f1cc5f3c160f.

Har bo
Lag di
www.af
eller rin
senter p

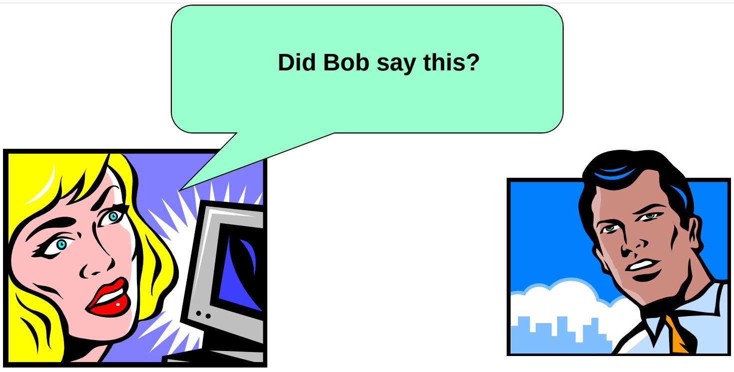
Motivation (1)



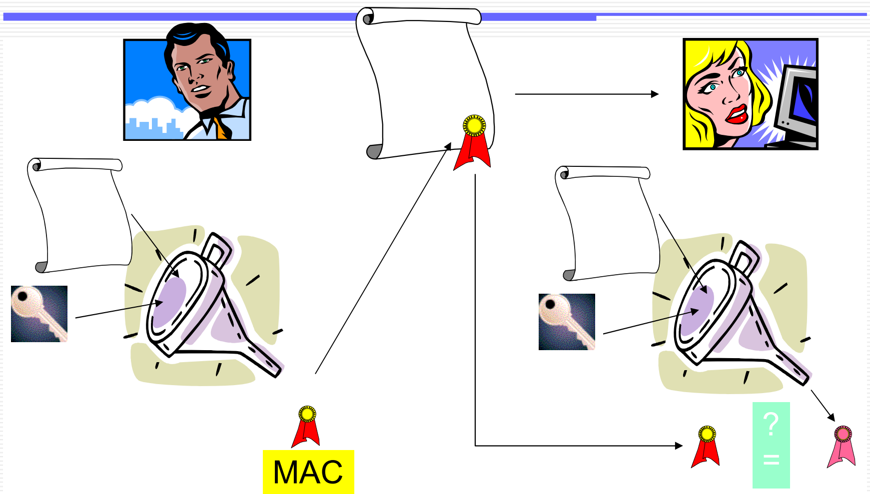
Solution (1)



Motivation (2)



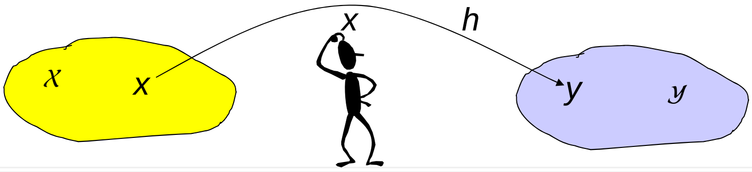
Solution(2)



Hash function property(1)

Problem 1: Preimage
Instance: A hash function $h : \mathcal{X} \to \mathcal{Y}$ and an element $y \in \mathcal{Y}$
Find: $x \in \mathcal{X}$ such that $h(x) = y$

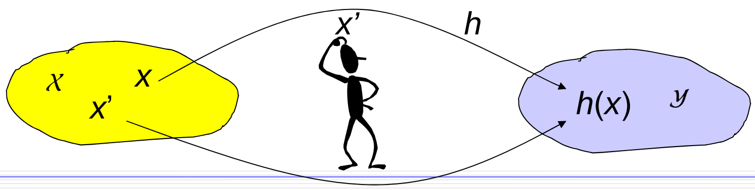
A *oneway* hash function is a hash function where the Preimage problem does not have any efficient solution (*preimage resistant*)



Hash function property(2)

Problem 2: Second Preimage
Instance: A hash function $h : \mathcal{X} \to \mathcal{Y}$ and an element $x \in \mathcal{X}$
Find: $x' \in \mathcal{X}$ such that $x \neq x'$ and $h(x') = h(x)$

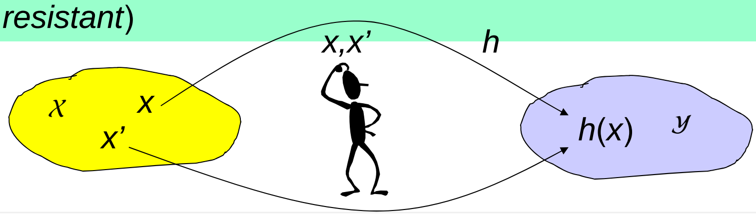
A hash function where Second Preimage does not have any efficient solution is called *second preimage resistant*



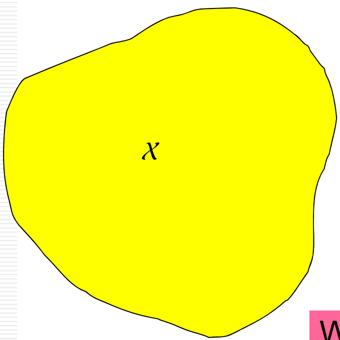
Hash function property(3)

Problem 3: Collision
Instance: A hash function $h : \mathcal{X} \rightarrow \mathcal{Y}$
Find: x, $x' \in \mathcal{X}$ such that $x \neq x'$ and $h(x') = h(x)$

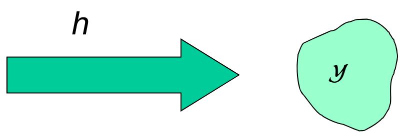
A *collisionfree* hash funktion is a hash function where Collision does not have any efficient solution (*Collision resistant*)



Does hash functions exist ?

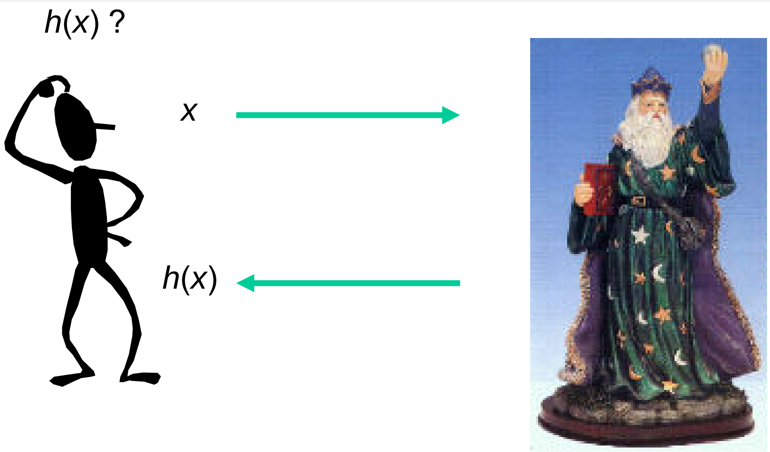


Since | $\mathcal{X}$ | >> | $\mathcal{Y}$ | they are “mathematical impossibilities”, that can only survive in a complexity restricted world !



We often consider a hash function as an instance of a random oracle!

Random oracle



Security level

- The generic security levels for a strong cryptographic hash functions which outputs *n* bits are:
  - For preimage attacks: 2^n
  - For second preimage attacks: 2^n
  - For second preimage attacks: $2^{n/2}$
- A hash functions of hash lengths 128 offers at best 64 bits security against a collision attack

The birthday paradox

Given a group of at least 23 persons. The probability for the event that two persons have the same birthday is at least 50% (1/2).

The birthday paradox

- Given a hash function $h : \{0, 1\}^* \rightarrow \{0, 1\}^M$. The probability for no collisions after q hash computations is:

$$p = \prod_{i=1}^{q-1}(1-\frac{i}{M}) \approx \prod_{i=1}^{q-1} e^{\frac{-i}{M}} = e^{-\sum_{i=1}^{q-1}\frac{i}{M}} = e^{\frac{-q(q-1)}{2M}}$$

Set the probability for at least one collision to1/2, we then have the following approximation for q :

$$q \approx 1.17\sqrt{M}$$

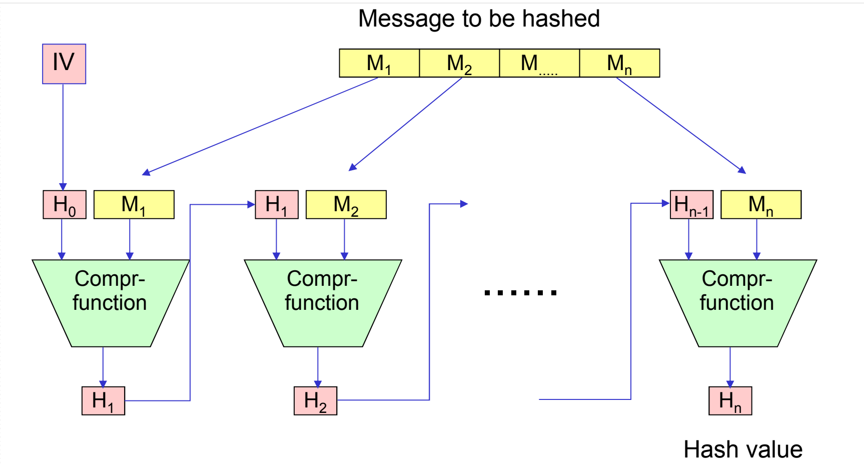
Hash Function Requirements

- can be applied to a block of data of any size
- produces a fixed-length output
- H(x) is relatively easy to compute for any given x
- one-way or pre-image resident
  - computationally infeasible to find x such that H(x) = h
- second pre-image resistant or weak collision resistant
  - computationally infeasible to find $y \neq x$ such that H(y) = H(x)
- collision resistant or strong collision resistance
  - computationally infeasible to find any pair (x, y) such that H(x) = H(y)

Security of Hash Functions

- **there** are two approaches to attacking a secure hash function:
  - cryptanalysis
    - exploit logical weaknesses in the algorithm
  - brute-force attack
    - strength of hash function depends solely on the length of the hash code produced by the algorithm
- SHA most widely used hash algorithm
- additional secure hash function applications:
  - passwords
    - hash of a password is stored by an operating system
  - intrusion detection
    - store H(F) for each file on a system and secure the hash values

Itererated hash functions



Known hash functions

- **MD4 (128)**
  - **Desiged by Ron Rivest 1990. Many attacks.**
- **MD5 (128)**
  - **Desiged by Ron Rivest 1990. Widely used. Today trivially to find collisions.**
- **RIPEMD-160**
  - **Developed by European project RIPE as an alternative to MD4 and MD5.**
- **SHA-1**
  - **Developed by NIST 1995 (Modified SHA). Hash value 160 bits. Collision attack of complexity 2^{63}**
- **SHA-2**
  - **Developed by NIST. Variants SHA-224, SHA-256, SHA-384 and SHA-512**

Technology status

- **MD5 is totally insecure for use in digital signature schemes. It is urgent to terminate such use.**
- **SHA-1 does not obtain the targeted security level (63 bits rather than 80)**
- **NIST recommends replacing SHA-1 for applications where collisions resistance is needed before2010**
- **Current alternative: use SHA-2**

New standard for secure hash functions

- **NIST has initiated a new project to develop a new family of secure hash functions – SHA-3**
  - **51 candidates out of 64 submissions were accepted for the first round 31. October 2008**
  - **14 candidate made it to the second round 24. July 2009**
  - **5 finialists published December 10, 2010**
  - **Target for new standard 2012!**

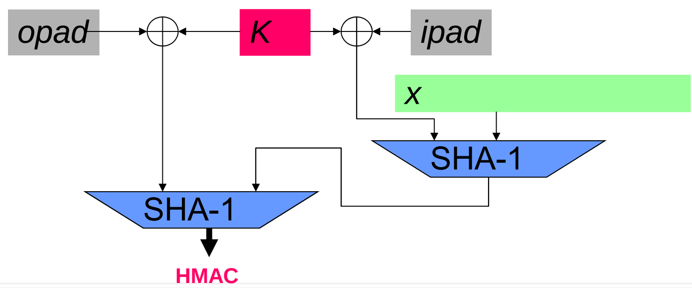
SHA-3 finalists

Hash Name	Principal Submitter	Best Attack on Main NIST Requirements	Best Attack on other Hash Requirements
BLAKE	Jean-Philippe Aumasson		
Grøstl	Lars R. Knudsen		
JH	Hongjun Wu	preimage	
Keccak	The Keccak Team		
Skein	Bruce Schneier		

SHA-3

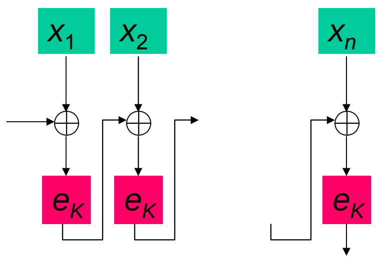
HMAC

- Define: *ipad* = 3636….36 (512 bit)
- *opad* = 5C5C…5C (512 bit)
- $HMAC_K(x) = SHA\text{-}1((K \oplus opad) \,||\, SHA\text{-}1((K \oplus ipad) \,||\, x))$



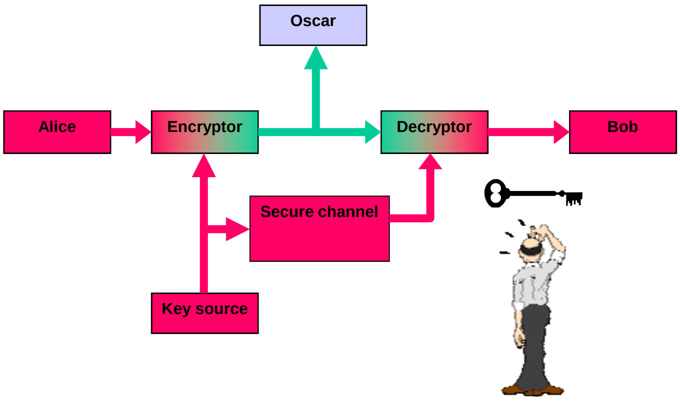
CBC-MAC

- **CBC-MAC(*x*, *K*)**
- sett $x = x_1 \,||\, x_2 \,||\, \ldots \,||\, x_n$
- $IV \leftarrow 00 \ldots 0$
- $y_0 \leftarrow IV$
- **for** $i \leftarrow 1$ **to** n
- **do** $y_i \leftarrow e_K(y_{i-1} \oplus x_i)$
- **return** (y_n)

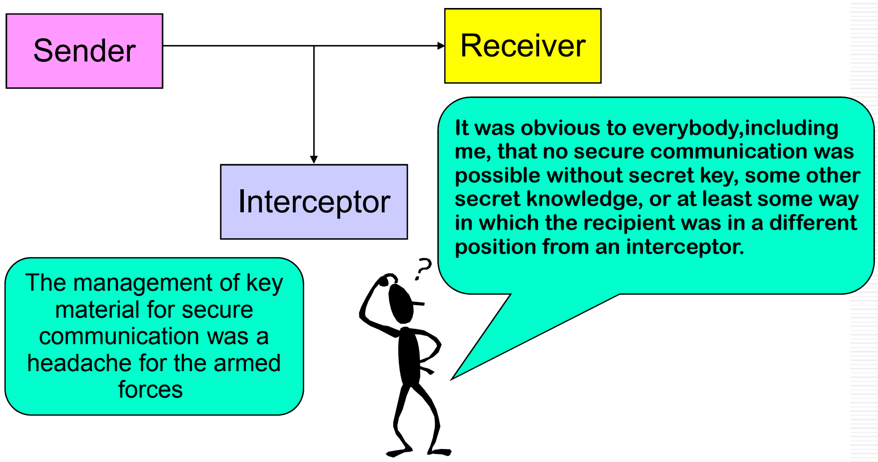


Public Key Crypto

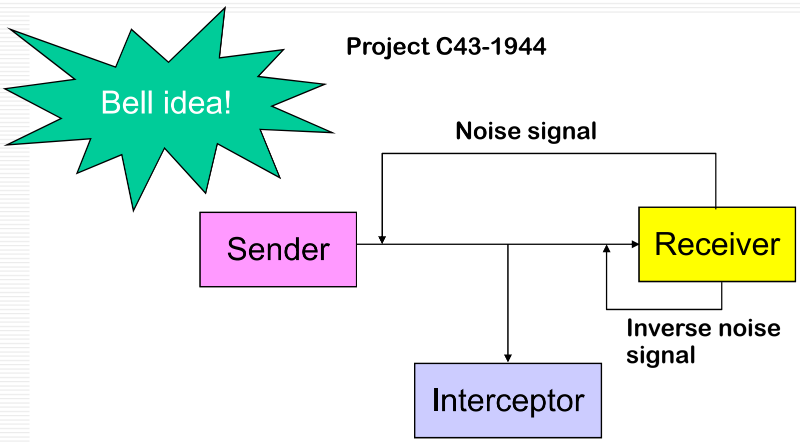
Symmetric cryptosystem



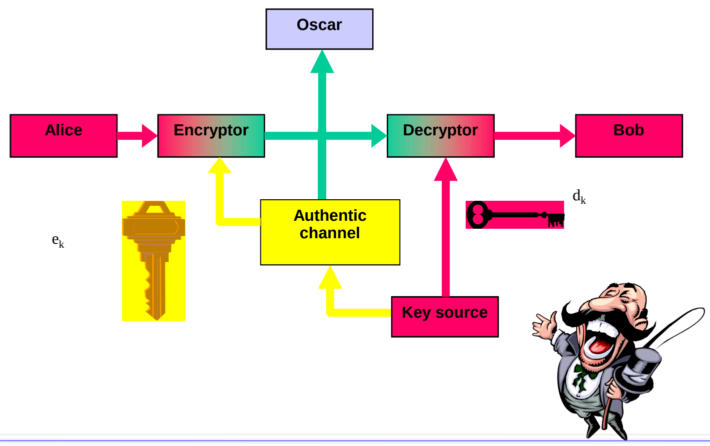
The impossible problem



An idea from the past

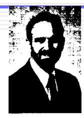


Asymmetrisk kryptosystem



Public key inventors?

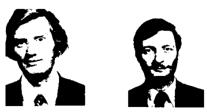
Marty Hellman and Whit Diffie, Stanford 1976



R. Rivest, A. Shamir and L. Adleman, MIT 1978

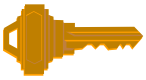
James Ellis, CESG 1970

C. Cocks, M. Williamson, CESG 1973-1974

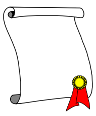


Asymmetric crypto

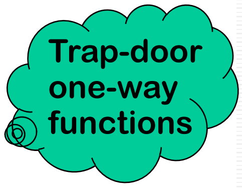
Public key cryptography was born in May 1975, the child of two problems and a misunderstanding!

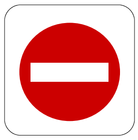


Key Distribution!



Digital signing!





One-way functions

Modular power function

Given $n = pq$, where p and q are prime numbers. No efficient algoritms to find p and q.

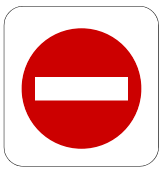
Chose a positive integer b and define $f : \mathbb{Z}_n \rightarrow \mathbb{Z}_n$

$f(x) = x^b \bmod n$

Modular exponentiation

Given prime p, generator g and a modular power $a = g^x \pmod{p}$. No efficient algoritms to find x. $f : \mathbb{Z}_p \rightarrow \mathbb{Z}_p$

$f(x) = g^x \bmod p$



RSA

- An equivalent system was described by Clifford Cocks (CESG) in 1973.
- RSA means Rivest, Shamir, Adleman (MIT) and was indepentently invented by these three in 1977
- RSA is a true asymmetric cryptosystem and can be used both for encryption and digital signatures
- RSA make use of the efficiency modular exponentiation, but to extract the correponding root is computational infeasable, i.e.:
  - Given message *M*, it is easy to compute $C = M^e (\text{mod } n)$, but hard to find *e*-th root of *C*.
- Note: RSA uses modular arithmetic over a composite moduli.
- The security of RSA is based on the difficulty of the factoring problem.

RSA parametre (textbook version)

- Bob generates two large prime numbers *p* and *q* and computes $n = p \cdot q$.
- He then computes a public encryption exponent *e*, such that
- $(e, (p-1)(q-1))) = 1$ and computes the corresponding decryption exsponent *d*, by solving:

$$d \cdot e \equiv 1 \ (\text{mod } (p-1)(q-1))$$

- Bob's public key is the pair $P_B = (e, n)$ and the corresponding private and secret key is $S_B = (d, n)$.

Encryption: $C = M^e \ (\text{mod } n)$
Decryption: $M = C^d \ (\text{mod } n)$

RSA mini example

- Set p = 157, q = 223. Then $n = p{\cdot}q$ =157 · 223 = 35011 and (p-1)(q-1) = 156 ·222 = 34632
- Set encryption exponent: e = 14213 {gcd(34632,14213) = 1}
- Public key: (14213, 35011)
- Compute: $d = e^{-1} = 14213^{-1}$ (mod 34632) = 31613
- Private key: (31613, 35011)

- Encryption:
- Plaintext M = 19726, then C = 19726^{14213} (mod 35011) = 32986

- Decryption:
- Ciphertext C = 32986, then M = 32986^{31613}(mod 35011) = 19726

Factoring record– December 2009

- Find the product of
- p = 33478071698956898786044169848212690817704794983713768568
- 912431388982883793878002287614711652531743087737814467999489
- and
- q= 36746043666799590428244633799627952632279158164343087642 6
- 76032283815739666511279233373417143396810270092798736308917?

Answer:

n= 12301866845301177551304949583849627207728535695953347921973 2
24521517264005072636575187452021997864693899564749427740638459 2
51925573263034537315482685079170261221429134616704292143116022 2
1240479274737794080665351419597459856902143413

Computation time ca. 0.0000003 s on a fast laptop!
RSA768 - Largest RSA-moduli that have been factored (12/12-2009)
Up to 2007 there was 50 000$ prize money for this factorisation!

Computational effort?

- Factoring using NFS-algorithm (Number Field Sieve)
- 6 mnd using 80 cores to find suitable polynomial
- Solding from August 2007 to April 2009 (1500 AMD64-år)
- 192 796 550 * 192 795 550 matrise (105 GB)
- 119 days on 8 different clusters
- Corresponds to 2000 years processing on one single core 2.2GHz AMD Opteron (ca. 2^{67} instructions)

Trends

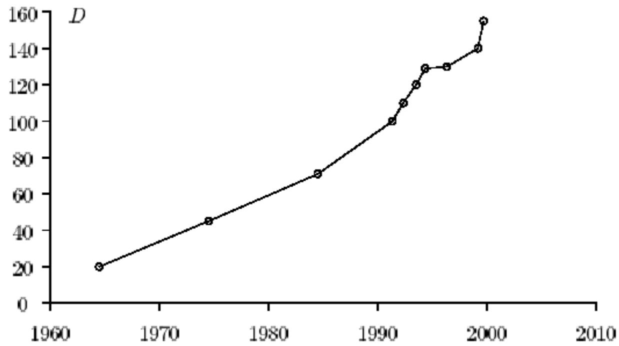


Figure 3: Size of "general" number factored versus year

Trends

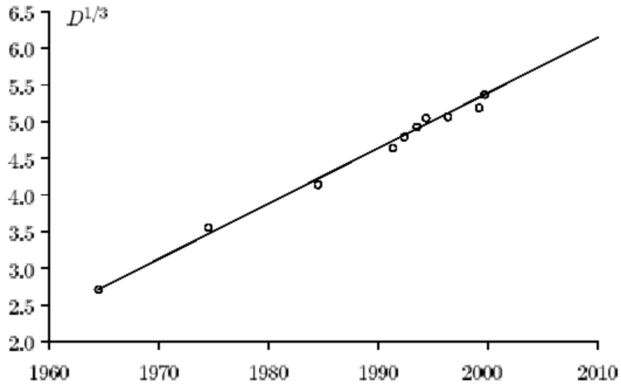


Figure 4: $D^{1/3}$ versus year Y

Quantum computation

- P. W. Shor showed in1994 that factoring can be done in expected polynomial time on a quantum computer!??

Warning

- Describing and understanding the RSA textbook version is easy!
- This version to not meet modern levels of security for a public key cryptosystem
- To use RSA encryption securely, we need randomization. E.g. use RSA-OAEP

The discrete logarithm problem

- Problem instance: $I = (p, g, b)$, where p is prime, $g \in \mathbb{Z}_p$ is a primitive element and $b \in \mathbb{Z}_p^*$.
-
- Question: Find unique a, $0 \leq a \leq p - 2$, such that

$$g^a \equiv b \pmod{p}?$$

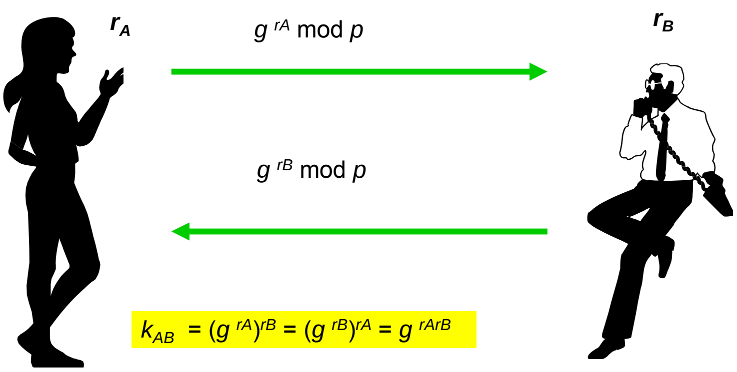
- We will denote a as $\log_g b$.
- Merk at problemet lett kan generaliseres til enhver syklisk gruppe!

Example

- $\mathbb{Z}_{11}$ med $\alpha = 2$:
  - $2^1 = 2 \pmod{11}$ $2^6 = 9 \pmod{11}$
  - $2^2 = 4 \pmod{11}$ $2^7 = 7 \pmod{11}$
  - $2^3 = 8 \pmod{11}$ $2^8 = 3 \pmod{11}$
  - $2^4 = 5 \pmod{11}$ $2^9 = 6 \pmod{11}$
  - $2^5 = 10 \pmod{11}$ $2^{10} = 1 \pmod{11}$
- $\log_2 5 = 4$
- $\log_2 7 = 7$
- $\log_2 1 = 10$ ($\equiv 0 \bmod 10$)

Diffie-Hellman key exchange

System parameters: p, g



$k_{AB} = (g^{\,r_A})^{r_B} = (g^{\,r_B})^{r_A} = g^{\,r_A r_B}$

NB! No authentication!!!

ElGamal public key crypto system

- Public key cryptosystem described by Taher El Gamal in 1984
- Makes use of the discrete logarithm problem
- Randomized encryption
- Used in Norwegian electronic voting system 2011

$$e_K(x, k) = (y_1, y_2)$$

$$y_1 = g^k \pmod{p} \text{ og } y_2 = xb^{\,k} \pmod{p}$$

Discrete Log based crypto

- Cryptographic primitives like Diffie-Hellman key excahnge and El-Gamal encryption can be implemented in any group where discrete log is infeasable.
- Common groups are $\mathbb{Z}_p^*$ and GF(2^n)*. Index-calculus algorithm can be used in these groups and large keys are required. (1000-2000 bit).
- Generic algorihtms has complexity $O(p^{1/2})$.
- *Elliptic curves* offer a rich source for abelian groups where no sub-exponential algorithms for DL are known.
- Use of cryptographice applications was proposed bu V. Miller and N. Koblitz in 1985.
- Same security level for shorter keys!

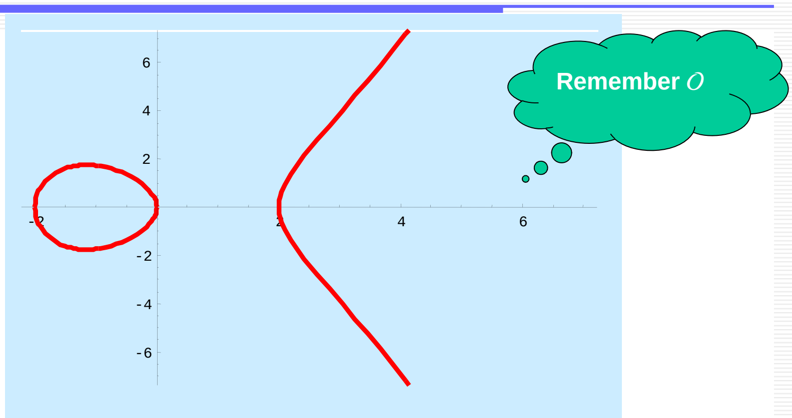
Elliptic curves

- Let $p > 3$ be a prime. An elliptic curve $y^2 = x^3 + ax + b$ over GF(p) = $\mathbb{Z}_p$ consist of all solutions $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$ to the equation

$$y^2 \equiv x^3 + ax + b \pmod{p}$$

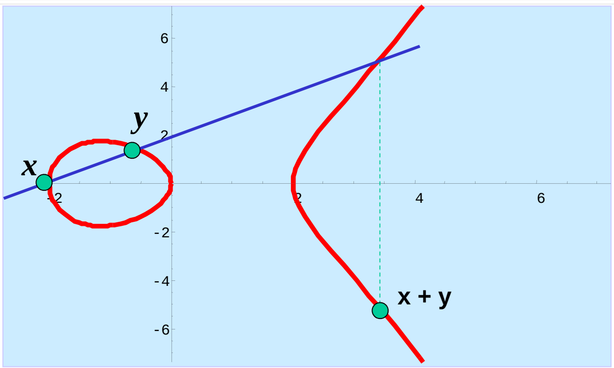
- where $a, b \in \mathbb{Z}_p$ are constants such that $4a^3 + 27b^2 \neq 0 \pmod{p}$, together with a special point $\mathcal{O}$ which is denoted as *the point at infinity*.

Elliptic curve over ℝ



$y^2 = x^3 - 4x$

Point addition



Security levels

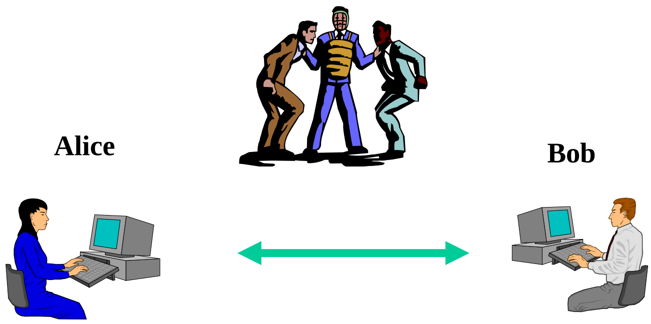
Symmetric	56	80	112	128	192	256
RSA n	512*	1024	2048	3072	7680	15360
DSA p	512*	1024	2048	3072	7680	15360
DSA q	112*	160	224	256	384	512
ECC n	112*	161	224	256	384	512

Asterisk (*) means below minimum keysize specified by ANSI X9 standard.

Table 2: Approximate equivalence of keys in bits to known best general attacks

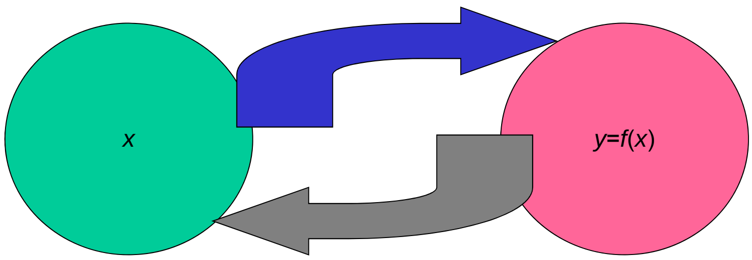
Motivation

How to solve electronic disputes?



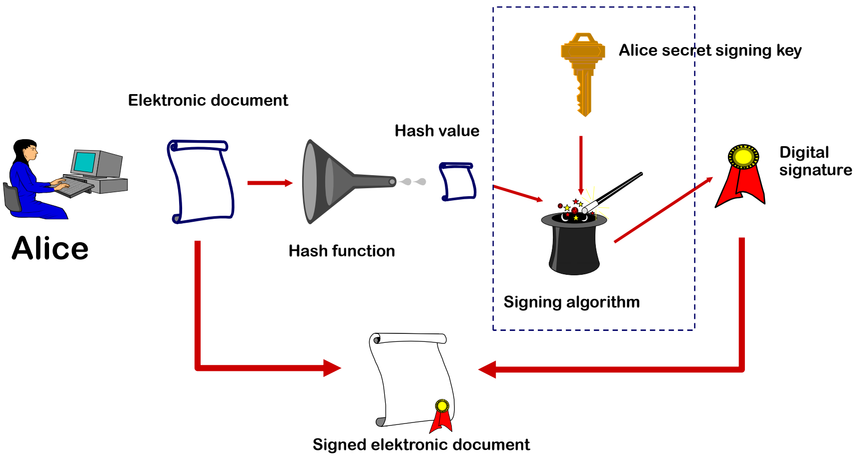
Diffie-Hellman approach

Trapdoor One Way Function

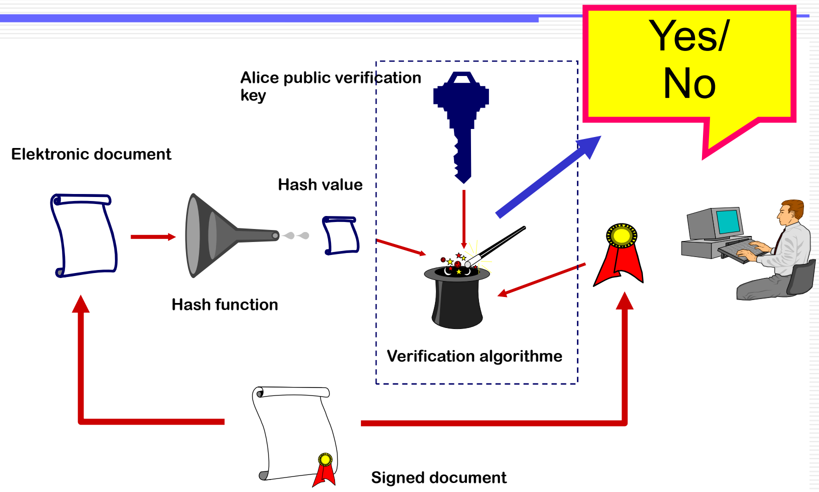


Signature is possible if *f* is a permutation

Signing using hash function



Verification using hash function



Digital Signature Algorithms

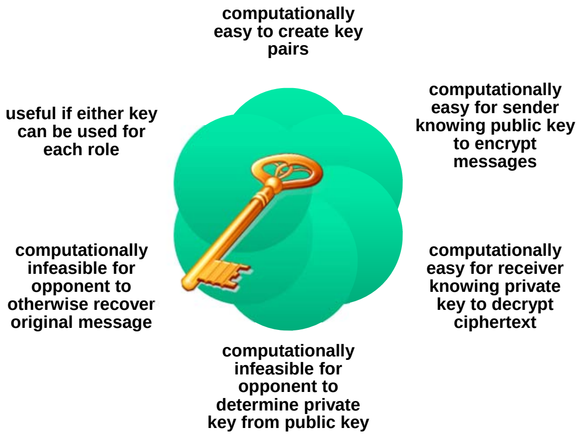
- RSA
- DSA
- El Gamal
- ECDSA

Table 2.3

Algorithm	Digital Signature	Symmetric Key Distribution	Encryption of Secret Keys
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

Applications for Public-Key Cryptosystems

Requirements for Public-Key Cryptosystems



Asymmetric Encryption Algorithms

- **RSA (Rivest, Shamir, Adleman)**
  - developed in 1977
  - most widely accepted and implemented approach to public-key encryption
  - block cipher in which the plaintext and ciphertext are integers between 0 and n-1 for some n.
- **Diffie-Hellman key exchange algorithm**
  - enables two users to securely reach agreement about a shared secret that can be used as a secret key for subsequent symmetric encryption of messages
  - limited to the exchange of the keys
- **Digital Signature Standard (DSS)**
  - provides only a digital signature function with SHA-1
  - cannot be used for encryption or key exchange
- **Elliptic curve cryptography (ECC)**
  - security like RSA, but with much smaller keys

Summary

- have considered:
  - Message authentication
  - Cryptographic hash functions
  - Public key cryptosystems
  - RSA
  - Discrete logarithms
  - Elliptic curve cryptography
  - Digital signatures