# INF3510 Information Security
# University of Oslo
# Autumn 2018

## Lecture 2
## Cryptography

University of Oslo, Autumn 2018

Nils Gruschka

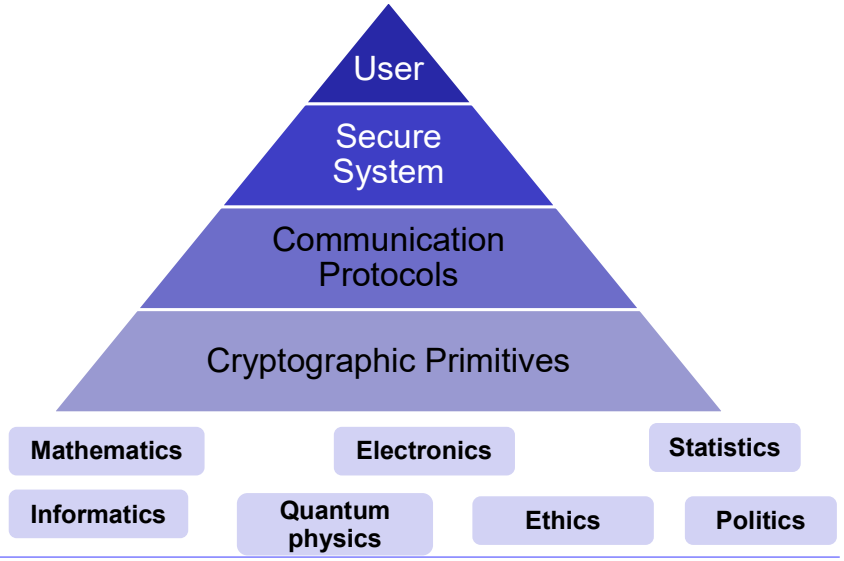
## Outline

- What is cryptography?
- Brief crypto history
- Security issues
- Symmetric cryptography
  - Stream ciphers
  - Block ciphers
  - Hash functions
- Asymmetric cryptography
  - Factoring based mechanisms
  - Discrete Logarithms
  - Digital signatures
  - Quantum Resistant Crypto

Want to learn more?
Look up UNIK 4220

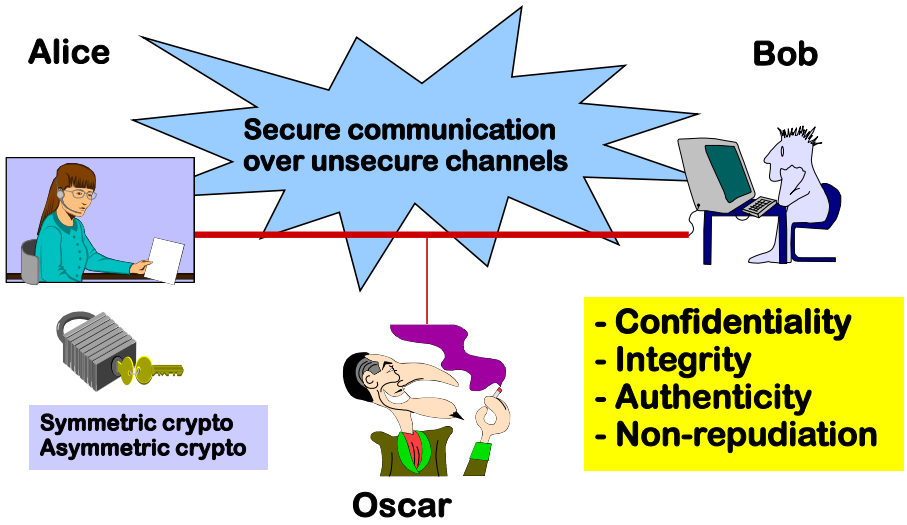


## The security pyramid

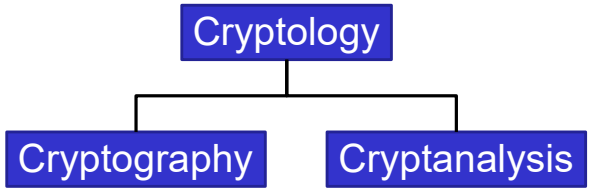





## What is cryptology?



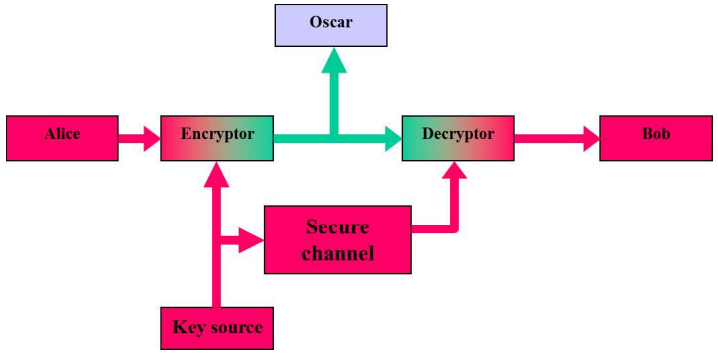
# Terminology




- **Cryptography** is the science of secret writing with the goal of hiding the meaning of a message.
- **Cryptanalysis** is the science and sometimes art of *breaking* cryptosystems.



# Model of symmetric cryptosystem






# Confidential Communication


# Caesar cipher

**Encryption:** Move all characters by *K* positions “to the right”

**Decrytion:** Move all characters by *K* positions “to the left”

**Example:** K = 5

**Plaintext:** `kryptologi er et spennende fag`

**Chiphertext:** `pwduytqtln jw jy xujssjsij kfl`

(Note: In the original Caesar cipher *K* was set fix to 3.)

# Numerical encoding of the alphabet

| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

| p | q | r | s | t | u | v | w | x | y | z | æ | ø | å |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 14 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |

Using this encoding many classical crypto systems can be expressed as algebraic functions over $\mathbb{Z}_{26}$ (English alphabet) or $\mathbb{Z}_{29}$ (Norwegian alphabet)



# Shift cipher

Let $\mathbf{P} = \mathbf{C} = \mathbb{Z}_{26}$. For $0 \le K \le 25$, we define

$E(x, K) = x + K \pmod{26}$

and

$D(y, K) = y - K \pmod{26}$

$(x, y \in \mathbb{Z}_{26})$

Question: What is the size of the key space?

Puzzle: ct =

LAHYCXPAJYQHRBWNNMNMOXABNLDANLXVVDWRLJCRXWB

Find the plaintext!



# Exhaustive search

For[i=0, i<26, i++, Print["Key = ", i, " Plain = ", decrypt[ct,1,i]]]

Key = 0 Plain = LAHYCXPAJYQHRBWNNMNMOXABNLDANLXVVDWRLJCRXWB
Key = 1 Plain = KZGXBWOZIXPGQAVMMLMLNWZAMKCZMKWUUCVQKIBQWVA
Key = 2 Plain = JYFWAVNYHWOFPZULLKLKMVYZLJBYLJVTTBUPJHAPVUZ
Key = 3 Plain = IXEVZUMXGVNEOYTKKJKJLUXYKIAXKIUSSATOIGZOUTY
Key = 4 Plain = HWDUYTLWFUMDNXSJJIJIKTWXJHZWJHTRRZSNHFYNTSX
Key = 5 Plain = GVCTXSKVETLCMWRIIHIHJSVWIGYVIGSQQYRMGEXMSRW
Key = 6 Plain = FUBSWRJUDSKBLVQHHGHGIRUVHFXUHFRPPXQLFDWLRQV
Key = 7 Plain = ETARVQITCRJAKUPGGFGFHQTUGEWTGEQOOWPKECVKQPU
Key = 8 Plain = DSZQUPHSBQIZJTOFFEFEGPSTFDVSFDPNNVOJDBUJPOT
Key = 9 Plain = CRYPTOGRAPHYISNEEDEDFORSECURECOMMUNICATIONS
Key = 10 Plain = BQXOSNFQZOGXHRMDDCDCENQRDBTQDBNLLTMHBZSHNMR
Key = 11 Plain = APWNRMEPYNFWGQLCCBCBDMPQCASPCAMKKSLGAYRGMLQ
Key = 12 Plain = ZOVMQLDOXMEVFPKBBABACLOPBZROBZLJJRKFZXQFLKP

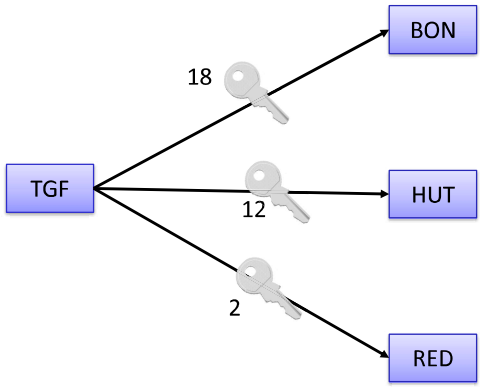


# Exhaustive search

For[i=0, i<26, i++, Print["Key = ", i, " Plain = ", decrypt[ct,1,i]]]

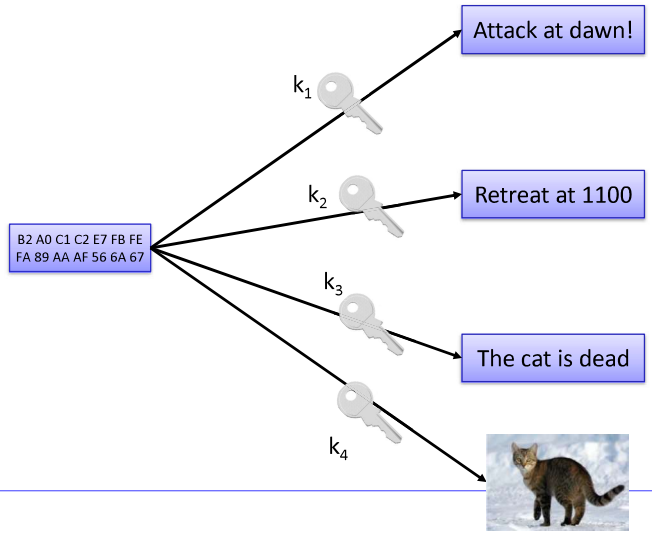
Key = 0 Plain = LAHYCXPAJYQHRBWNNMNMOXABNLDANLXVVDWRLJCRXWB
Key = 1 Plain = KZGXBWOZIXPGQAVMMLMLNWZAMKCZMKWUUCVQKIBQWVA
Key = 2 Plain = JYFWAVNYHWOFPZULLKLKMVYZLJBYLJVTTBUPJHAPVUZ
Key = 3 Plain = IXEVZUMXGVNEOYTKKJKJLUXYKIAXKIUSSATOIGZOUTY
Key = 4 Plain = HWDUYTLWFUMDNXSJJIJIKTWXJHZWJHTRRZSNHFYNTSX
Key = 5 Plain = GVCTXSKVETLCMWRIIHIHJSVWIGYVIGSQQYRMGEXMSRW
Key = 6 Plain = FUBSWRJUDSKBLVQHHGHGIRUVHFXUHFRPPXQLFDWLRQV
Key = 7 Plain = ETARVQITCRJAKUPGGFGFHQTUGEWTGEQOOWPKECVKQPU
Key = 8 Plain = DSZQUPHSBQIZJTOFFEFEGPSTFDVSFDPNNVOJDBUJPOT
Key = 9 Plain = CRYPTOGRAPHYISNEEDEDFORSECURECOMMUNICATIONS
Key = 10 Plain = BQXOSNFQZOGXHRMDDCDCENQRDBTQDBNLLTMHBZSHNMR
Key = 11 Plain = APWNRMEPYNFWGQLCCBCBDMPQCASPCAMKKSLGAYRGMLQ
Key = 12 Plain = ZOVMQLDOXMEVFPKBBABACLOPBZROBZLJJRKFZXQFLKP

# Exhaustive search




Finding the correct key is hard, without knowledge of the plaintext.

# One-Time Pad Encryption
# Exhaustive search




# Brute-Force Attack
# (or Exhaustive Key Search)

- Treats the cipher as a black box
- Requires (at least) one plaintext-ciphertext pair ($x_0$, $y_0$)
- Check all possible keys $K$ until condition is fulfilled:

$$d_K(y_0) = x_0$$

- How many keys do we need to test?

# Substitution cipher - example

| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| U | D | M | I | P | Y | Æ | K | O | X | S | N | Å | F | A |

| p | q | r | s | t | u | v | w | x | y | z | æ | ø | å |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E | R | T | Z | B | Ø | C | Q | G | W | H | L | V | J |

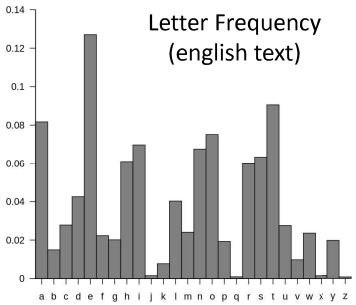
Plaintext: fermatssisteteorem
Ciphertext: YPTÅUBZZOZBPBPATPÅ

What is the size of the key space?

$29! \approx 2^{103}$

# Breaking (monoalphabetic) Substitution ciphers

- How to break a message like this?
  - Tvctujuvujpo djqixs jt b nxuipe pg xodszqujoh cz fijdi vojut pg qmbjouxyu bsx sxqmbdxe fjui djqixsuxyu; bddpsejoh up b gjyxe tztuxn- uix vojut nbz cx tjohmx mxuuxst uix nptu dpnnpo; qbjst pg mxuuxst; usjqmxut pg mxuuxst; njyuvsxt pg uix bcpwx; boe tp gpsui, Uix sxdxjwxs exdjqixst uix uxyu cz qxsgpsnjoh uix jowxstx tvctujuvujpo,



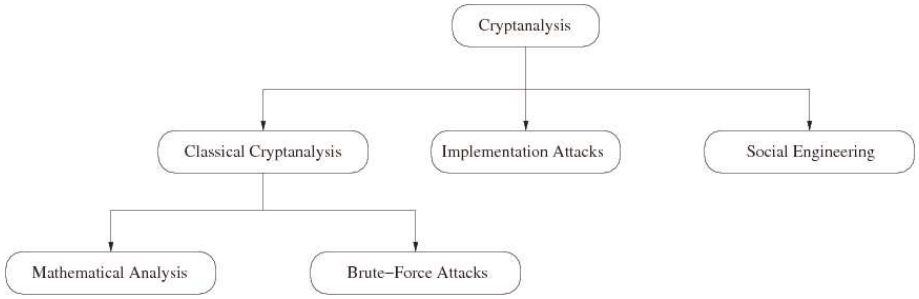



# Lessons learned

- A cipher with a small keyspace can easily be attacked by *exhaustive search*
- A *large keyspace* is necessary for a secure cipher, but it is by itself not suffcient
- Monoalphabetical substitution ciphers can easily be broken



# Cryptanalysis: Attacking Cryptosystems



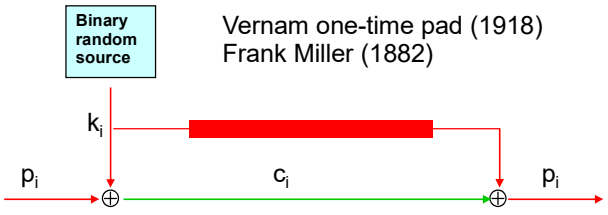

- **Classical Attacks**
  - Mathematical Analysis
  - Brute-Force Attack
- **Implementation Attack**: Try to extract the key through reverse engineering or power measurement, e.g., for a banking smart card.
- **Social Engineering**: E.g., trick a user into giving up her password



# Does secure ciphers exist?

- What is a secure cipher?
  - Perfect security
  - Computational security
  - Provable security

# A perfect secure crypto system




$c_i = p_i \oplus k_i$
$p_i = c_i \oplus k_i = p_i \oplus k_i \oplus k_i = p_i$
Note: $a \oplus b = a + b \pmod 2$

Offers perfect security assuming the key is perfectly random, of same length as The Message; and only used once. Proved by Claude E. Shannon in 1949.



# ETCRRM

- Electronic Teleprinter Cryptographic Regenerative Repeater Mixer (ETCRRM)
- Invented by the Norwegian Army Signal Corps in 1950
- Bjørn Rørholt, Kåre Mesingseth
- Produced by STK
- Used for ”Hot-line” between Moskva and Washington
- About 2000 devices produced



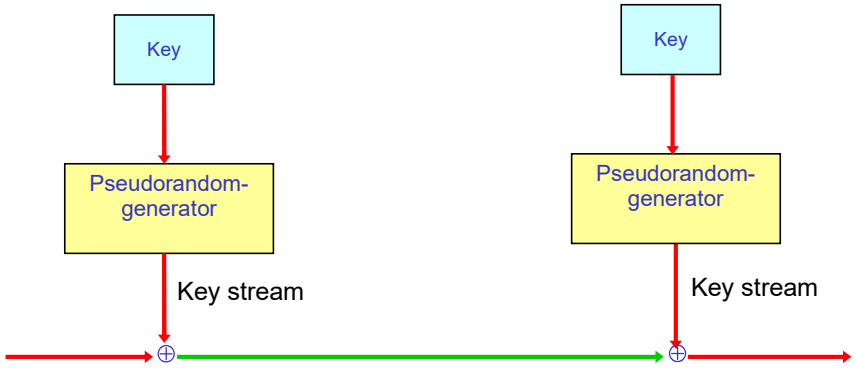
# White House Crypto Room 1960s



# Symmetric encryption

- Is it possible to design secure and practical crypto?

# Symmetric stream cipher



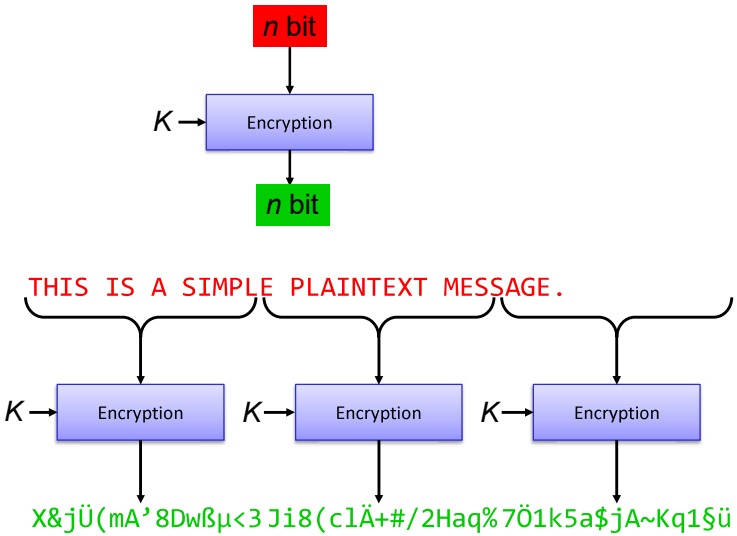



# Symmetric stream cipher

- Examples:
  - A5/1 and A5/2 (1989; used in GSM) → broken
  - RC4 (1987) → broken
  - Salsa20 (2005)
  - ChaCha20 (2008)

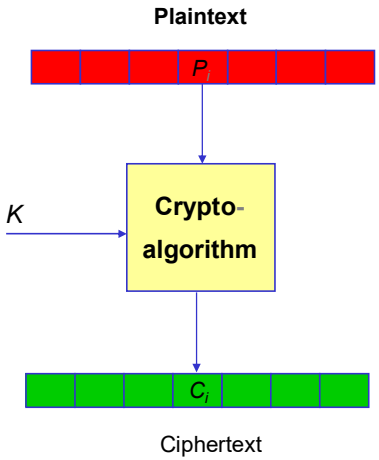


# Symmetric Block Cipher






# Symmetric block cipher




- The algorithm represents a family of permutations of the message space
- Normally designed by iterating a less secure round function
- May be applied in different operational modes
- Must be impossible to derive *K* based on knowledge of *P* and *C*

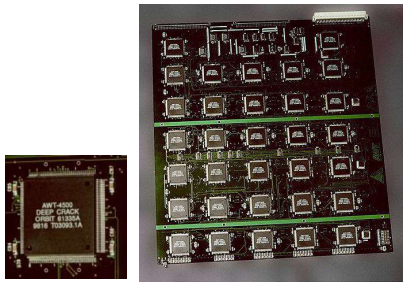
# Data Encryption Standard

- Published in 1977 by the US National Bureau of Standards for use in unclassified government applications with a 15 year life time.
- 16 round Feistel cipher with 64-bit data blocks, 56-bit keys.
- 56-bit keys were controversial in 1977; today, exhaustive search on 56-bit keys is very feasible.
- Controversial because of classified design criteria, however no loop hole was ever found.



# Breaking DES

- Brute force attack on a 56 bit DES key:
  - 1998: EFF DES Cracker (ASICs), 4.5 days, 250.000$
  - 2006: COPACOBANA (FPGA), 6.4 days, 10.000$
  - 2012: Pico Computing (FPGA), 0.5 days



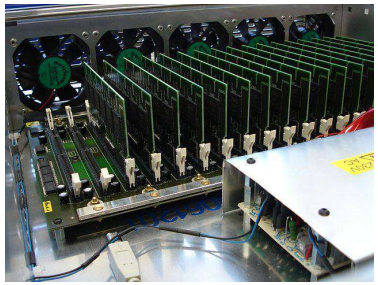


Image Source: Wikipedia, Gerd Pfeiffer, EFF

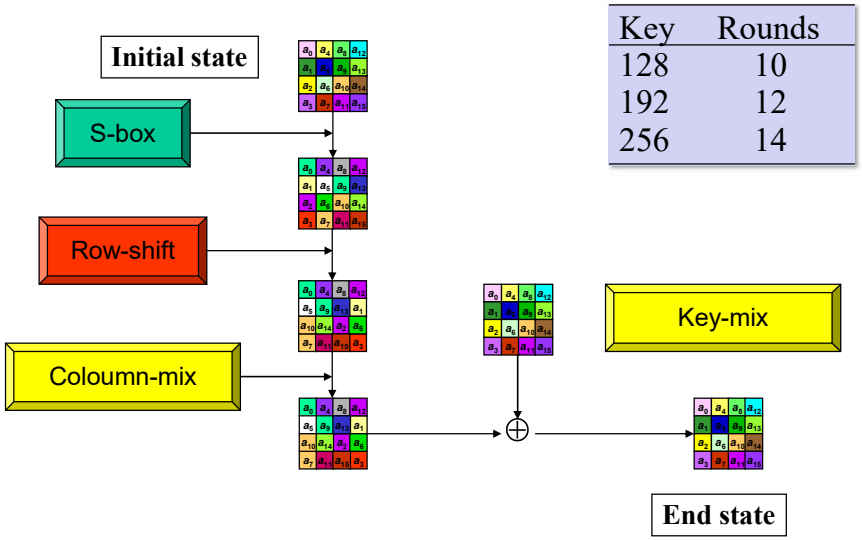


# Advanced Encryption Standard

- Public competition to replace DES: because 56-bit keys and 64-bit data blocks no longer adequate.
- Rijndael nominated as the new Advanced Encryption Standard (AES) in 2001 [FIPS-197].
- Rijndael (pronounce as “Rhine-doll”) designed by Vincent Rijmen and Joan Daemen.
- 128-bit block size (Note error in Harris p. 809)
- 128-bit, 196-bit, and 256-bit key sizes.



# Rijndael round function




| Key | Rounds |
|---|---|
| 128 | 10 |
| 192 | 12 |
| 256 | 14 |

# Breaking AES

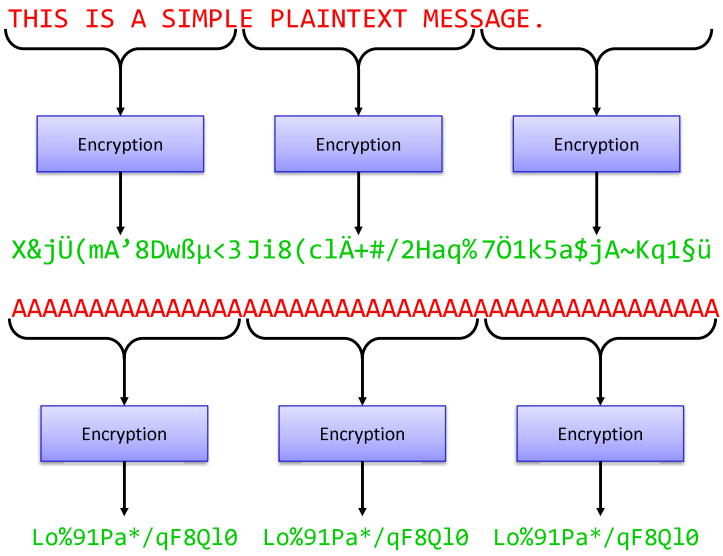
- How long does a brute force attack on a 128 or 256 bit AES key take?
- Assumption: breaking 56 bit in 1 second

| Key length | Duration |
|---|---|
| 56 bit | 1 s |
| 64 bit | 4 m |
| 80 bit | 194 d |
| 112 bit | $10^9$ a |
| 128 bit | $10^{14}$ a |
| 192 bit | $10^{33}$ a |
| 256 bit | $10^{52}$ a |

| Key length in bit | Key space | Security life time (assuming brute-force as best possible attack) |
|---|---|---|
| 64 | $2^{64}$ | **Short term** (few days or less) |
| 128 | $2^{128}$ | **Long-term** (several decades in the absence of quantum computers) |
| 256 | $2^{256}$ | **Long-term** (also resistant against quantum computers – note that QC do not exist at the moment and might never exist) |

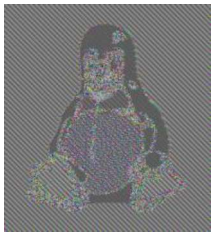


# Block Cipher: Electronic Code Book

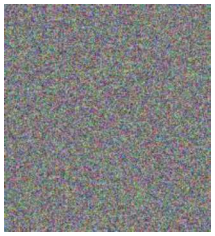



# Modes in Block Ciphers

Plaintext



Ciphertext using ECB mode



Ciphertext using CBC mode



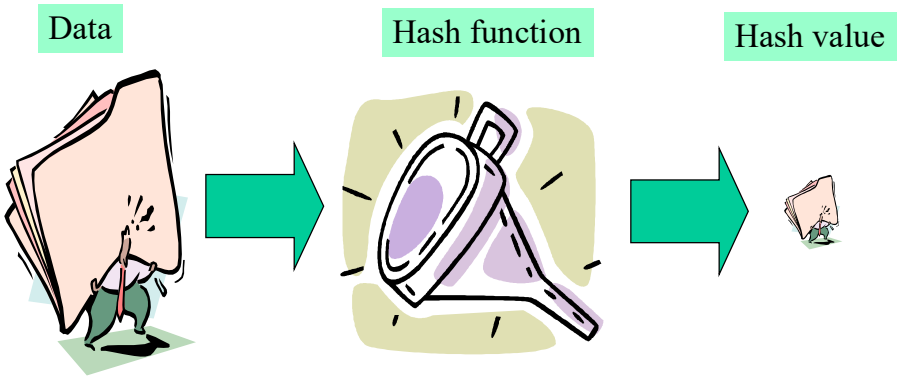
# Block Ciphers: Modes of Operation

- Block ciphers can be used in different modes in order to provide different security services.
- Common modes include:
  - **E**lectronic **C**ode **B**ook (ECB)
  - **C**ipher **B**lock **C**haining (CBC)
  - **G**alois **C**ounter **M**ode (GCM) {Authenticated encryption}

# Integrity Check Functions



# Hash functions






# ISBN-13 Number


- 978-0-306-40615-r

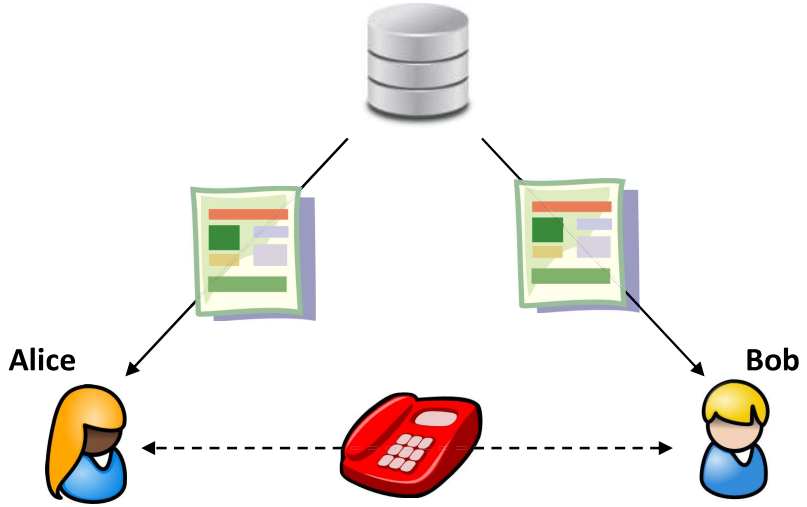
$$r = \left(10 - \left(x_1 + 3x_2 + x_3 + 3x_4 + \cdots + x_{11} + 3x_{12}\right) \bmod 10\right)$$

- r = 7 → 978-0-306-40615-7

- (Weak) integrity protection:
  - Transmit ISBN number (just the first 12 digits)
  - Transmit check digit on a secure channel
  - Receiver compares calculated $r_1$ and received check digit $r_2$
  - If any one of the 12 digits is changed during transmission → $r_1 \neq r_2$



# Applications of hash functions



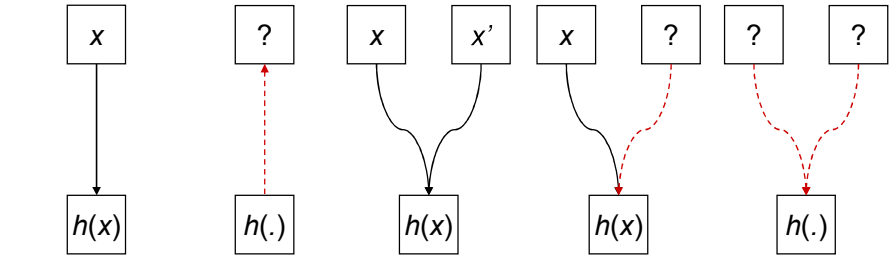
# Hash functions (message digest functions)

Requirements for a one-way hash function *h*:

1. Ease of computation: given *x*, it is easy to compute *h*(*x*).
2. Compression: *h* maps inputs *x* of arbitrary bitlength to outputs *h*(*x*) of a fixed bitlength *n*.
3. One-way: given a value *y*, it is computationally infeasible to find an input *x* so that *h*(*x*)=*y*.
4. Collision resistance: it is computationally infeasible to find *x* and *x'*, where $x \neq x'$, with *h*(*x*)=*h*(*x'*) (note: two variants of this property).



# Properties of hash functions




Ease of computation | Pre-image resistance | Collision | Weak collision resistance ($2^{nd}$ pre-image resistance) | Strong collision resistance



# Further applications of hash functions

- Comparing files
- Protection of password
- Authentication of SW distributions
- Bitcoin
- Generation of Message Authentication Codes (MAC)
- Digital signatures
- Pseudo number generation/Mask generation functions
- Key derivation



# Frequently used hash functions

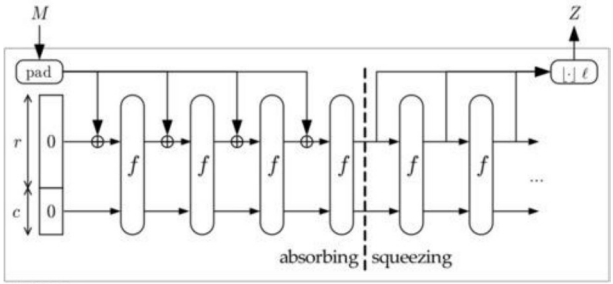
- MD5: 128 bit digest. Broken. Often used in Internet protocols but no longer recommended.
- SHA-1 (Secure Hash Algorithm):160 bit digest. Potential attacks exist.
- SHA-2: Replacement for SHA-1. 256, 384 or 512 bit digest (called SHA-256, SHA-384, SHA-512). Still secure.
- RIPEMD-160: 160 bit digest. Still secure. Hash function frequently used by European cryptographic service providers.
- NIST competition for new secure hash algorithm (SHA-3), closed in 2012

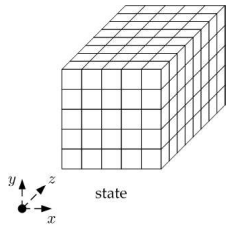
# And the winner is?

- NIST announced Keccak as the winner of the SHA-3 Cryptographic Hash Algorithm Competition on October 2, 2012, and ended the five-year competition.
- Keccak was designed by a team of cryptographers from Belgium and Italy, they are:
  - Guido Bertoni (Italy) of STMicroelectronics,
  - Joan Daemen (Belgium) of STMicroelectronics,
  - Michaël Peeters (Belgium) of NXP Semiconductors, and
  - Gilles Van Assche (Belgium) of STMicroelectronics.



# Keccak and sponge functions



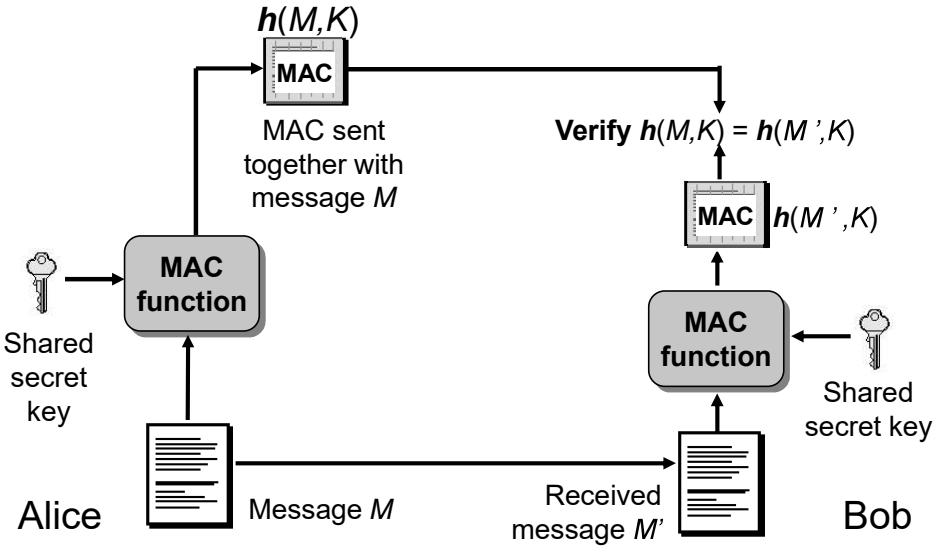



# MAC and MAC algorithms

- MAC means two things:
  1. The computed message authentication code *h(M, k)*
  2. General name for algorithms used to compute a MAC
- In practice, the MAC algorithm is e.g.
  - HMAC (Hash-based MAC algorithm))
  - CBC-MAC (CBC based MAC algorithm)
  - CMAC (Cipher-based MAC algorithm)
- MAC algorithms, a.k.a. keyed hash functions, support data origin authentication services.



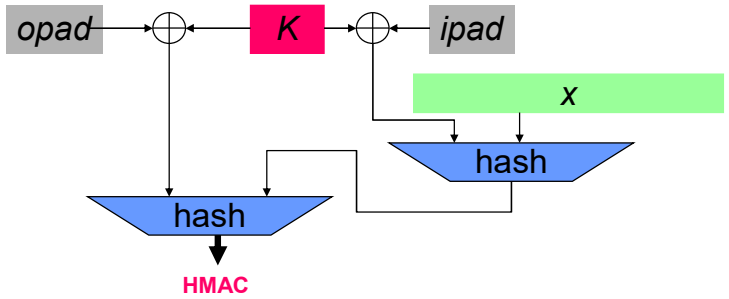
# Practical message integrity with MAC



# HMAC

- Define: *ipad* = 3636….36 (512 bit)
- *opad* = 5C5C…5C (512 bit)

- $HMAC_K(x)$ = hash(($K \oplus opad$) || hash(($K \oplus ipad$) || $x$ ))

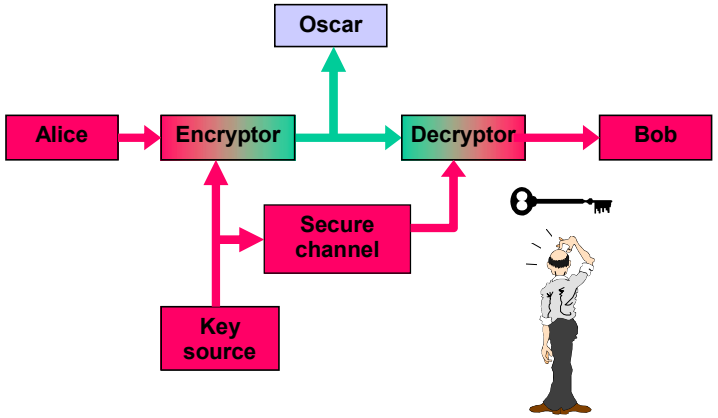





# Public-Key Cryptography

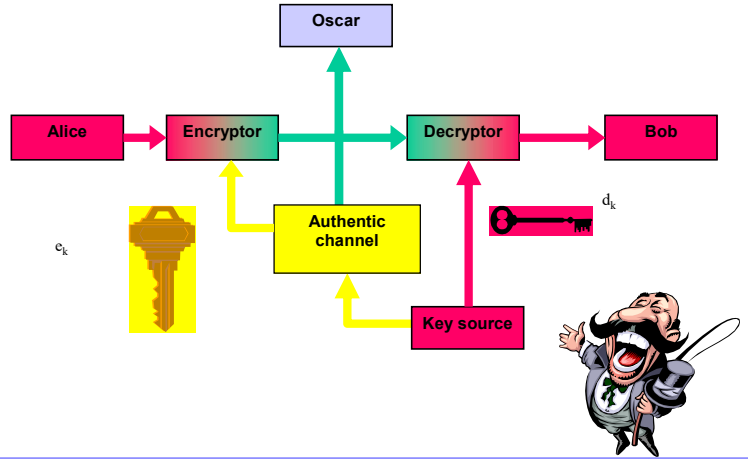


# Symmetric cryptosystem






# Asymmetric crypto system



# Public key inventors?

Marty Hellman and Whitfield Diffie, Stanford 1976


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

James Ellis, CESG 1970

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

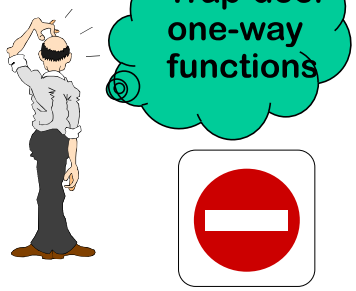



# Asymmetric crypto

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 $f : Z_n \rightarrow Z_n$

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

Modular exponentiation

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

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



# Diffie-Hellman Key Exchange

- https://www.youtube.com/watch?v=YEBfamv-_do&t=2m6s

# Diffie-Hellman key agreement (key exchange)
(provides no authentication)

Alice picks random integer $a$

Bob picks random integer $b$

$g^a \bmod p$

$g^b \bmod p$

Computationally impossible to compute discrete logarithm

Alice computes the shared secret
$(g^b)^a = g^{ab} \bmod p$

Bob computes the same secret
$(g^a)^b = g^{ab} \bmod p$.



# Example

- $Z_{11}$ using $g = 2$:
  - $2^1 = 2 \pmod{11}$ $2^6 = 9 \pmod{11}$
  - $2^2 = 4 \pmod{11}$ $2^7 = 7 \pmod{11}$
  - $2^3 = 8 \pmod{11}$ $2^8 = 3 \pmod{11}$
  - $2^4 = 5 \pmod{11}$ $2^9 = 6 \pmod{11}$
  - $2^5 = 10 \pmod{11}$ $2^{10} = 1 \pmod{11}$

- $\log_2 5 = 4$
- $\log_2 7 = 7$
- $\log_2 1 = 10$ ($\equiv 0 \bmod 10$)



# Example (2)

p =
30196626334536652266746444111852771272047217220445439805218819842806439806980163153421277779853237655786915947633907457862442472144616346714598423225826077976000905549946633556169688641786953396004062371399599729544977400404541673313622576825171747563463840240911791172271560696187007629722341591375265838579703621423172371480685909595288918038021902829382836838643722330258240598676263586947720295337695281786665678795149819992726746898859863000921247304925995410219082086727278137148522572014844749083522090193190746907275606521624184144352256368927493398678089550310568789287558755227001418448833563517776833964003

g =
17214844102945427204136512177889538496379881834679876598474115714966161705073026628129298835010174348250308006877834103702727269721499966768323290540216992770986728538508742382941595672248624817994917939749447675055374786840972654044030577846000645054950424877666860986820152109887355204363179653945098490724068905414681792636510652507946102434852166272721706635011474226289945817893390827991578201408649196984764863302981052471409215846871176739109049866118609117954454512573209668379576042056062096628325900231910090325301911333152181394803908610214937044613411740650800989334729586051242347771056691010439032429058

Finn $a$ når

$g^a \pmod p$ =
44113216355065215159684488639683249149092460427650288245942898766876571824921690276662620979153820952830455103982849705054980427000258241321067445164291945709875449674237106754516103276658256727241360337237692098033897604855715556428192853384013674273248985055064876109463005314835390642583853176983615599073922523609689343385582696033895191791219150497333537020837218564219880414922079856566434665604898681669845852964624047443239120501341277499692338517113201830210812184500672101247270098803275601662656616757996322304239541426757926222214762596502305241986906124402779894141043268551743878130988606078310881106617



# Solution

a =
71893136149709653804503478677866573695060790720621260648699193249561437588126371185816941549290993967522517872683465480518953201710796636526807415642002868814878889631989535331117023603483665844918711772382064485518405530594550171022761555809365778193109639893698220411548578601884177129022057550866690223052160523604836233675971504259382476301273682533632952920247361439377799123181423154997117475318825014240822522816464111195458755823011214081322669809865473902563660710642521281242103815550156237005192231836155067262308141154795194735834753570104459663325337960304941906119476181818583000946627658955269636154 06

It is easy to compute $g^a \pmod p$ {0.016 s}, but it is computaionally infeasable to compute the exponent $a$ from the $g^a$.

## Diffie-Hellman Applications

- SSL/TLS
  - (Ephemeral) Diffie-Hellman is nowadays the standard for key exchange in SSL/TLS protocol
- IPSec (IP Security)
  - IKE (Internet Key Exchange) is part of the IPSec protocol suite
  - IKE is based on Diffie-Hellman Key Agreement



## Ron Rivest, Adi Shamir and Len Adleman



- Read about public-key cryptography in 1976 article by Diffie & Hellman: *"New directions in cryptography"*
- Intrigued, they worked on finding a practical algorithm
- Spent several months in 1976 to re-invent the method for non-secret/public-key encryption discovered by Clifford Cocks 3 years earlier
- Named RSA algorithm



## RSA parameters (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
- $\gcd(e, (p-1)(q-1)) = 1$ and computes the corresponding decryption exponent $d$, by solving:

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

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

Encryption: $C = M^e \pmod{n}$
Decryption: $M = C^d \pmod{n}$



## RSA toy example

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

- Encryption:
- Plaintext M = 19726, then $C = 19726^{14213} \pmod{35011} = 32986$

- Decryption:
- Cipherertext C = 32986, then $M = 32986^{31613} \pmod{35011} = 19726$

# Factoring record– December 2009

Find the product of

p = 33478071698956898786044169848212690817704794983713768568
912431388982883793878002287614711652531743087737814467999489

and

q= 367460436667995904282446337996279526322791581643430876426
76032283815739666511279233373417143396810270092798736308917?

Answer:

n= 123018668453011775513049495838496272077285356959533479219732
245215172640050726365751874520219978646938995647494277406384592
519255732630345373154826850791702612214291346167042921431160222
124047927473779408066535141959745985690214341 3

Computation time ca. 0.0000003 s on a fast laptop!
RSA768 - Largest RSA-modulus 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)



# Asymmetric Ciphers: Examples of Cryptosystems

- RSA: best known asymmetric algorithm.
  - RSA = Rivest, Shamir, and Adleman (published 1977)
  - Historical Note: U.K. cryptographer Clifford Cocks invented the same algorithm in 1973, but didn't publish.
- ElGamal Cryptosystem
  - Based on the difficulty of solving the discrete log problem.
- Elliptic Curve Cryptography
  - Based on the difficulty of solving the EC discrete log problem.
  - Provides same level of security with smaller key sizes.



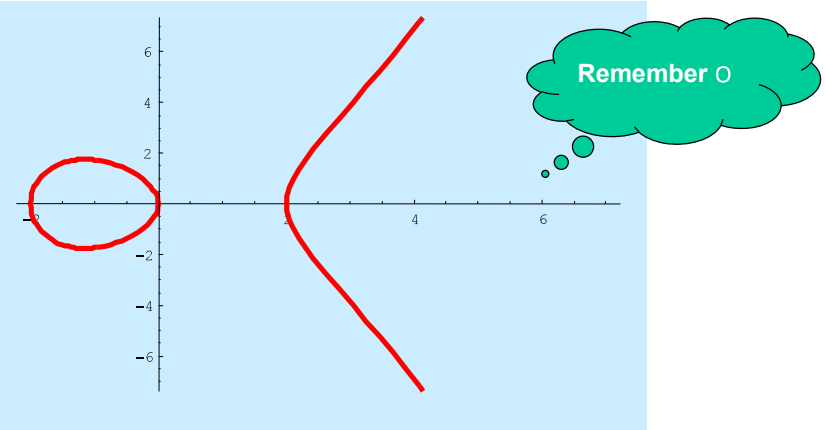
# Elliptic curves

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

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

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

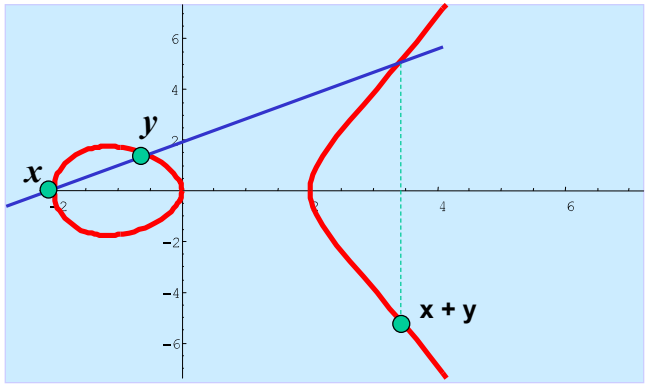
# Elliptic curve over R




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



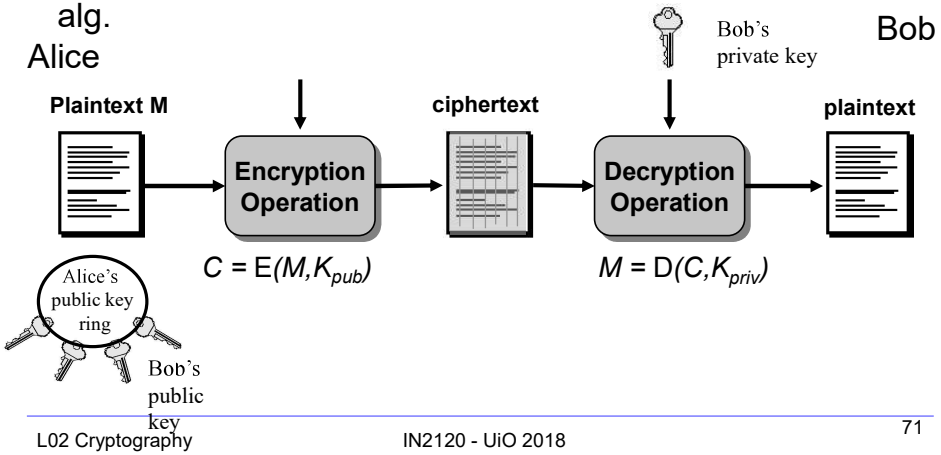
# Point addition






# Asymmetric Encryption: Basic encryption operation

- In practice, large messages are not encrypted directly with asymmetric algorithms. Hybrid systems are used, where only symmetric session key is encrypted with asymmetric alg.

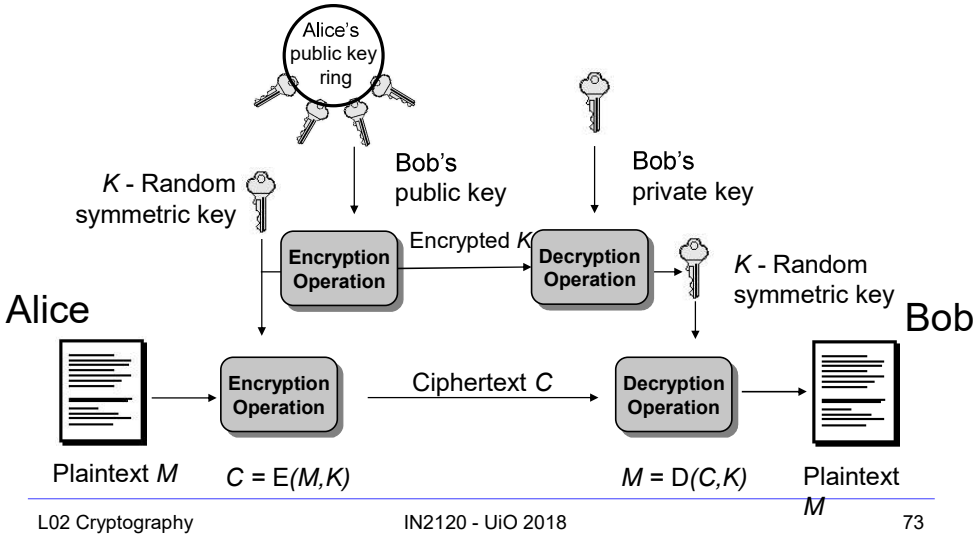





# Hybrid Cryptosystems

- Symmetric ciphers are faster than asymmetric ciphers (because they are less computationally expensive ), but ...
- Asymmetric ciphers simplify key distribution, therefore ...
- a combination of both symmetric and asymmetric ciphers can be used – a hybrid system:
  - The asymmetric cipher is used to distribute a randomly chosen symmetric key.
  - The symmetric cipher is used for encrypting bulk data.

# Confidentiality Services: Hybrid Cryptosystems






# Digital Signatures

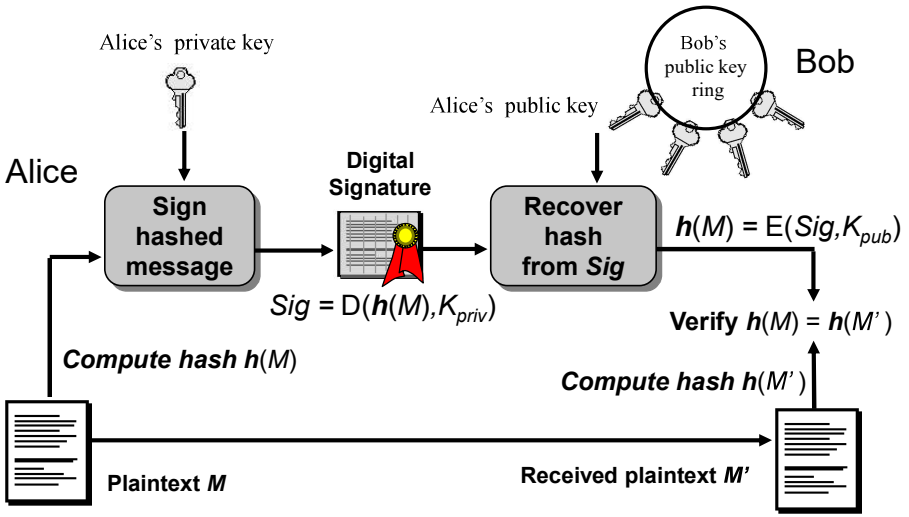


# Digital Signature Mechanisms

- A MAC cannot be used as evidence that should be verified by a third party.
- Digital signatures used for non-repudiation, data origin authentication and data integrity services, and in some authentication exchange mechanisms.
- Digital signature mechanisms have three components:
  - key generation
  - signing procedure (private)
  - verification procedure (public)
- Algorithms
  - RSA
  - DSA and ECDSA



# Practical digital signature based on hash value



# Digital Signatures

- To get an authentication service that links a document to *A*'s name (identity) and not just a verification key, we require a procedure for *B* to get an authentic copy of *A*'s public key.
- Only then do we have a service that proves the authenticity of documents 'signed by *A*'.
- This can be provided by a PKI (Public Key Infrastructure)
- Yet even such a service does not provide non-repudiation at the level of persons.



# Difference between MACs & Dig. Sig.


- MACs and digital signatures are both authentication mechanisms.
- MAC: the verifier needs the secret that was used to compute the MAC; thus a MAC is unsuitable as evidence with a third party.
  - The third party does not have the secret.
  - The third party cannot distinguish between the parties knowing the secret.

- Digital signatures can be validated by third parties, and can in theory thereby support both non-repudiation and authentication.



# Key length comparison:

Symmetric and Asymmetric ciphers offering comparable security

| AES Key Size | RSA Key Size | Elliptic curve Key Size |
|---|---|---|
| - | 1024 | 163 |
| 128 | 3072 | 256 |
| 192 | 7680 | 384 |
| 256 | 15360 | 512 |

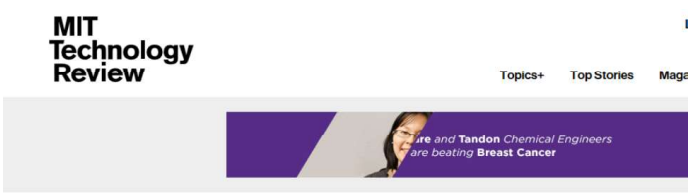


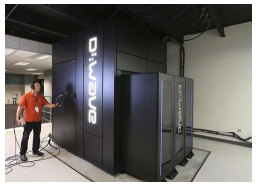
# Another look at key lengths

Table 1. Intuitive security levels.

| | | bit-lengths | | |
|---|---|---|---|---|
| security level | volume of water to bring to a boil | symmetric key | cryptographic hash | RSA modulus |
| teaspoon security | 0.0025 liter | 35 | 70 | 242 |
| shower security | 80 liter | 50 | 100 | 453 |
| pool security | 2 500 000 liter | 65 | 130 | 745 |
| rain security | 0.082 $km^3$ | 80 | 160 | 1130 |
| lake security | 89 $km^3$ | 90 | 180 | 1440 |
| sea security | 3 750 000 $km^3$ | 105 | 210 | 1990 |
| global security | 1 400 000 000 $km^3$ | 114 | 228 | 2380 |
| solar security | - | 140 | 280 | 3730 |

# The eavesdropper strikes back!



Computing

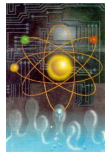
**NSA Says It "Must Act Now" Against the Quantum Computing Threat**

The National Security Agency is worried that quantum computers will neutralize our best encryption – but doesn't yet know what to do about that problem.

by Tom Simonite February 3, 2016



# Quantum Computers

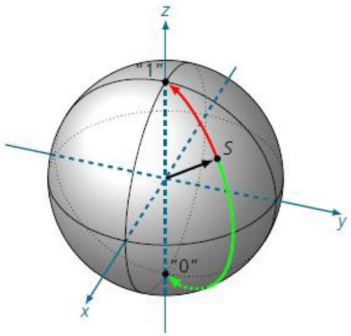


- Proposed by Richard Feynman 1982
- Boosted by P. Schor's algorithm for integer factorization and discrete logarithm in quantum polynomial time
- Operates on qubit – superposition of 0 and 1
- IBM built a 7-bit quantum computer and could find the factors of the integer 15 using NMR techniques in 2001
- NMR does not scale
- Progress continues, but nobody knows if or when a large scale quantum computer ever can be constructed
- QC will kill current public key techniques, but does not mean an end to symmetric crypto



# Classical bit vs. qubits

- 1
- 0

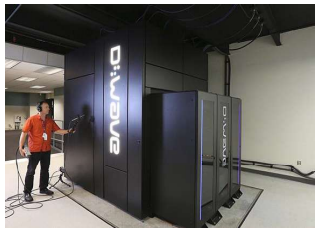



$\Psi = \alpha | 0 \rangle + \beta | 1 \rangle$, where $| \alpha |^2 + | \beta |^2 = 1$



# QC impact to cryptography

- When will a quantum computer be built?
  - 15 years, $1 billion USD, nuclear power plant (PQCrypto 2014, Matteo Mariantoni)
- Impact:
  - Public key crypto:
    - ~~RSA~~
    - ~~Elliptic Curve Cryptography (ECDSA)~~
    - ~~Finite Field Cryptography (DSA)~~
    - ~~Diffie-Hellman key exchange~~
  - Symmetric key crypto:
    - AES Need larger keys
    - Triple DES Need larger keys
  - Hash functions:
    - SHA-1, SHA-2 and SHA-3 Use longer output



# Current world record of QF!

**Table 5:** *Quantum factorization records*

| Number | # of factors | # of qubits needed | Algorithm | Year implemented | Implemented without prior knowledge of solution |
|---|---|---|---|---|---|
| 15 | 2 | 8 | Shor | 2001 [2] | × |
| | 2 | 8 | Shor | 2007 [3] | × |
| | 2 | 8 | Shor | 2007 [3] | × |
| | 2 | 8 | Shor | 2009 [5] | × |
| | 2 | 8 | Shor | 2012 [6] | × |
| 21 | 2 | 10 | Shor | 2012 [7] | × |
| 143 | 2 | 4 | minimization | 2012 [1] | ✓ |
| 56153 | 2 | 4 | minimization | 2012 [1] | ✓ |
| 291311 | 2 | 6 | minimization | not yet | ✓ |
| 175 | 3 | 3 | minimization | not yet | ✓ |

2017

Source: http://phys.org/news/2014-11-largest-factored-quantum-device.html



# Two variants of quantum safe crypto

**Quantum cryptography:**

•The use of quantum mechanics to guarantee secure communication.

•It enables two parties to produce a shared random secret key known only to them, which can then be used to encrypt and decrypt messages.

**Quantum resistant cryptography:**

•The use of cryptographic mechanisms based on computationally difficult problems for which no efficient quantum computing algorithm is known



# QKD via satellite




# Quantum Resistant Cryptography

- Code Based Asymmetric Algorithms
- Lattice Based Asymmetric Algorithms
- Asymmetric Crypto based on Multivariate Polynomials
- Asymmetric Crypto based on Cryptographic Hash Functions
- Asymmetric Crypto based on Isogenies of (supersingular) elliptic curves

# Recent news

News room > News releases >

## IBM Announces Advances to IBM Quantum Systems & Ecosystem

-- Client systems with 20 qubits ready for use; next-generation IBM Q system in development with first working 50 qubit processor
-- IBM expands its open-source quantum software package QISKit; offers the world's most advanced ecosystem for quantum computing

**Select a topic or year**

- News release
- Contact(s) information
- Related XML feeds
- Related resources

**Yorktown Heights, N.Y. - 10 Nov 2017:** IBM (NYSE: IBM) announced today two significant quantum processor upgrades for its IBM Q early-access commercial systems. These upgrades represent rapid advances in quantum hardware as IBM continues to drive progress across the entire quantum computing technology stack, with focus on systems, software, applications and enablement.



# Recent news

KVANTEDATAMASKIN

**Svenske forskere får nær en milliard kroner til å utvikle kvantedatamaskin**

AV: HARALD BROMBACH | MASKINVARE | PUBLISERT: 15. NOV. 2017 - 12:00

Facebook 34 Twitter

Med et budsjett på nærmere en milliard svenske kroner (omtrent 970 millioner NOK) skal prosessor Per Delsing ved Chalmers tekniska högskola i Göteborg lede et initiativ som har som mål å utvikle en virkelig kraftig kvantedatamaskin.

– Målet vårt er å ha en fungerende kvantedatamaskin med minst hundre qubit. En slik datamaskin har mye større regnekraft enn de beste superdatamaskinene i dag og kan bli brukt til for eksempel å løse optimaliseringsproblemer, avansert maskinlæring og tunge beregninger av egenskapene til molekyler, sier Delsing i en pressemelding.

Andre områder hvor kvanteteknologien er aktuell, er innen kryptering og design av medisiner.

**Les også:** Intels nye brikke skal være et gjennombrudd innenfor kvantedatamaskiner

Se alle jobber innen IT

**Minst 100 qubit**

En qubit er i denne sammenheng en elektrisk mikrobrikke som tar vare på kvantetilstanden til ett enkelt foton. Mens 100 vanlig bit vanligvis er en ganske ubetydelig datamengde i dagens tradisjonelle datamaskiner, kan man i kvantedatamaskiner prosessere enorme datamengder selv med relativt få qubit.

IBM kunngjorde i forrige uke at selskapet har fått en prosessor med 50 qubit til å fungere, så 100 qubit er ikke til å kimse av, selv om det svenske prosjektet skal pågå i et tiår.

I likhet med blant annet IBM, skal den svenske forskningsgruppen benytte superledende kretser. Superledende qubit-er er noe forskere ved Chalmers har jobbet med i nærmere 20 år.

Ledige IT-stillinger JOBB

Tekmidler (seniorrådgiver) IKT servicedesk Drammen

Rådgiver IKT servicedesk Drammen

IKT integrasjonsrådgiver Askim

LES OGSÅ

**Intels nye brikke skal være et gjennombrudd innenfor kvantedatamaskiner**



# Brave new crypto world……




# End of lecture