CSE413 – Security of Information Systems 2020

PhD Furkan Gözükara, Toros University

https://github.com/FurkanGozukara/Security-of-Information-Systems-CSE413-2020

Lecture 4

Cryptography – Part 2

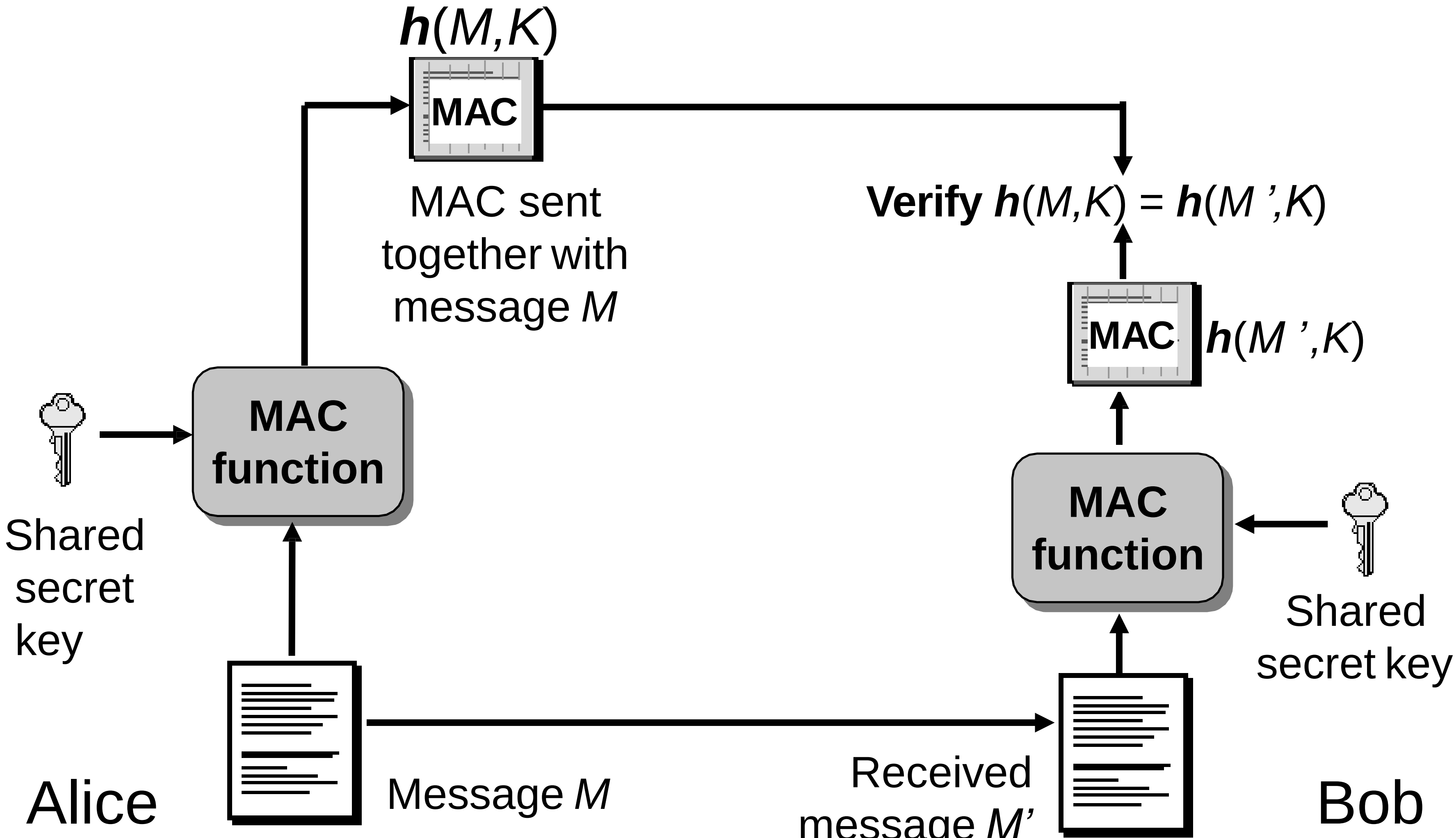
Composed from Prof. Audun Jøsang, University of Oslo, Information Security 2018 Lectures

Source : https://www.uio.no/studier/emner/matnat/ifi/INF3510/v18/lectures/

MAC and MAC algorithms

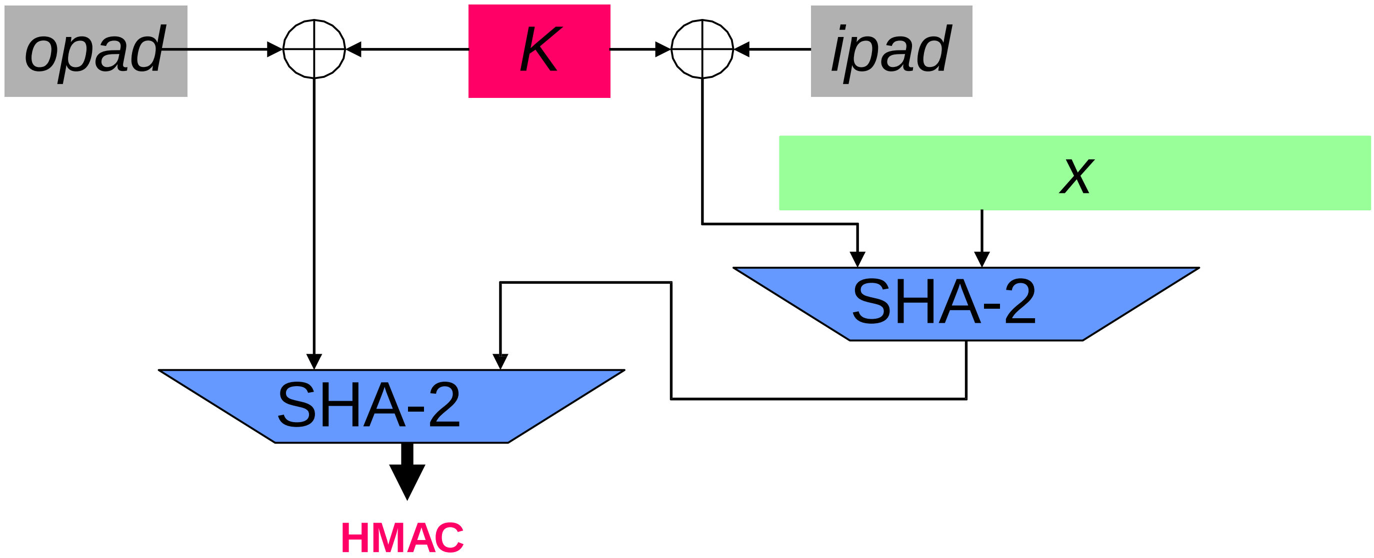
- MAC means two things:
  1. The computed message authentication code $h(M, k)$.
  2. General name for algorithms used to compute a MAC.
- In practice, the MAC algorithm is e.g.
  – HMAC (Hash-based MAC algorithm)).
  – CBC-MAC (CBC based MAC algorithm).
  – CMAC (Cipher-based MAC algorithm).
- MAC algorithms, 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)
- $\oplus$ = XOR
- $\text{HMAC}_K(x) = \text{SHA-1}((K \oplus opad) \,||\, \text{SHA-1}((K \oplus ipad) \,||\, x))$



CBC-MAC

CBC-MAC($\boldsymbol{x}$, $\boldsymbol{K}$)

set $x = x_1 \,||\, x_2 \,||\, \ldots. \,||\, x_n$

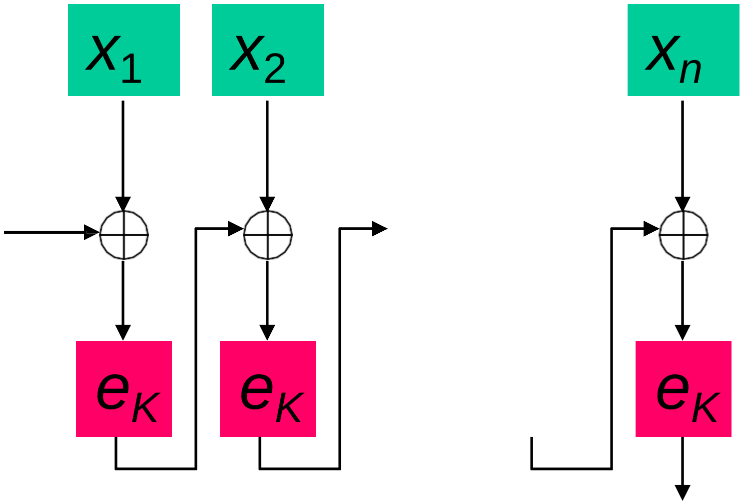
IV $\leftarrow$ 00 … 0

$y_0 \leftarrow$ IV

for $i \leftarrow 1$ **to** n

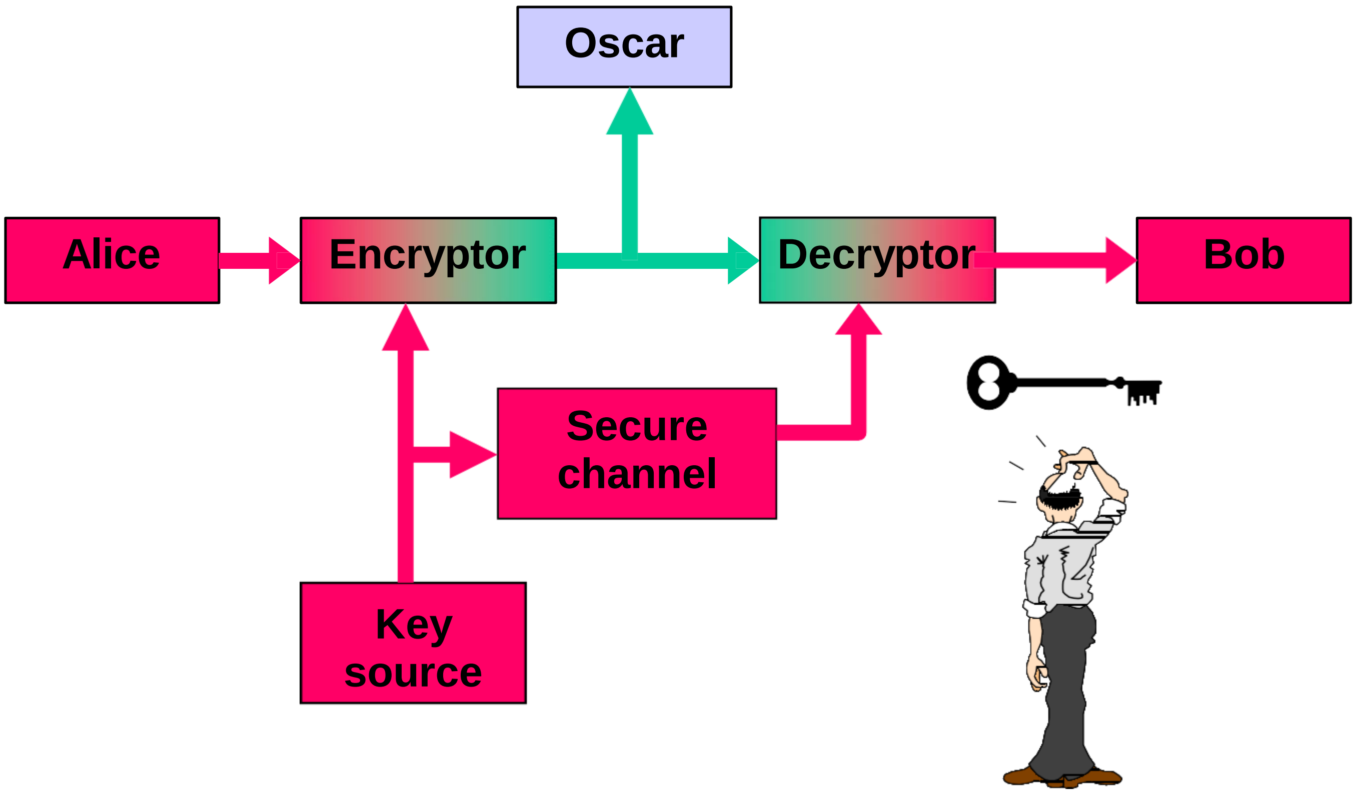
do $y_i \leftarrow e_K(y_{i-1} \oplus x_i)$

return (y_n)

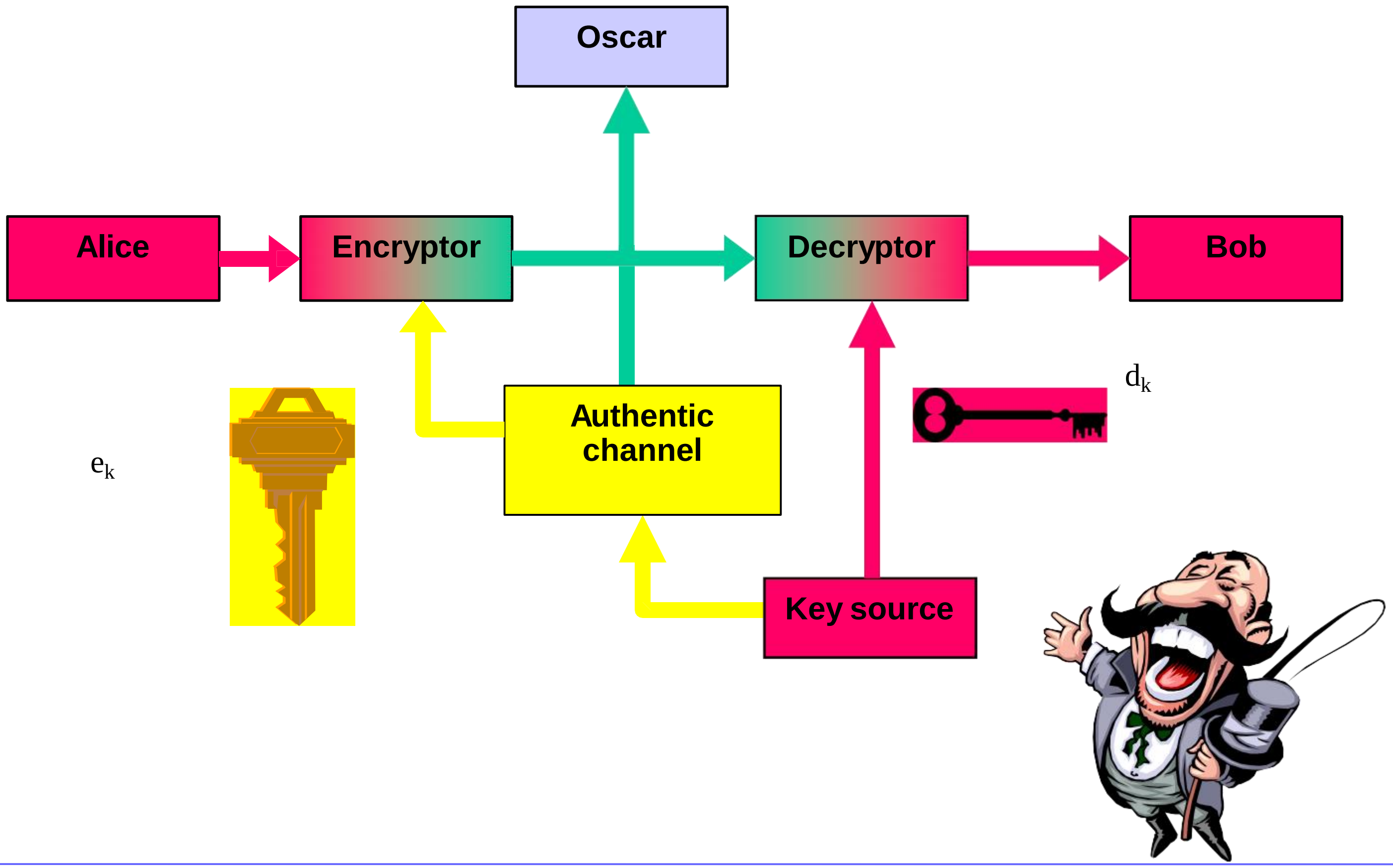


Public-Key Cryptography

Symmetric cryptosystem

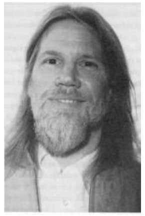


Asymmetric crypto system



Public key inventors?

Marty Hellman and Whit Diffie, Stanford 1976

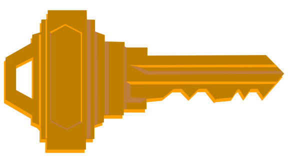
R. Rivest, A. Shamir and L.Adleman, MIT 1978

James Ellis, CESG 1970

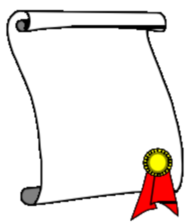
C. Cocks, M. Williamson, CESG 1973-1974


Asymmetric crypto

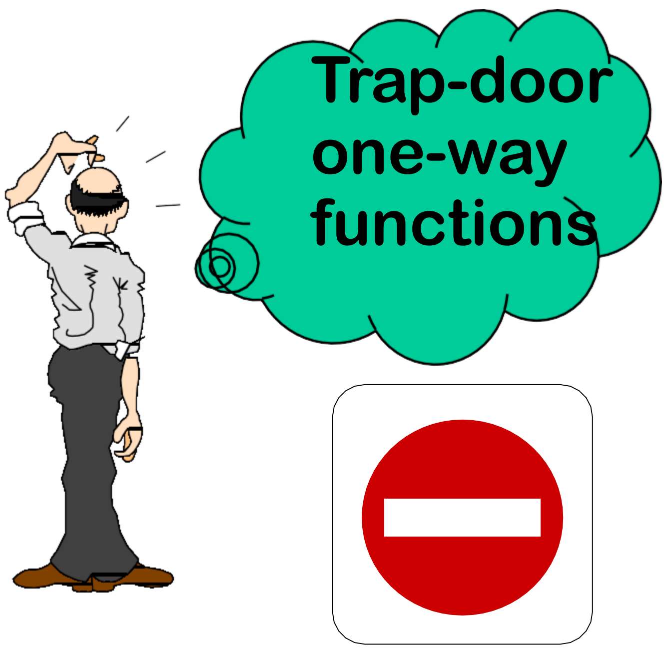
Public key cryptography was born in May 1975, the child of two problems and a misunderstanding!



Key Distribution!



Digital signing!



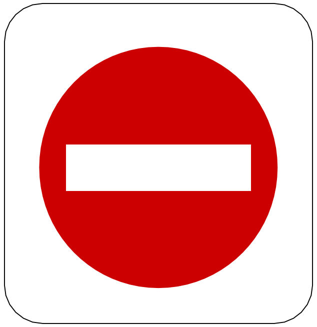
One-way functions

<u>Modular power function</u>

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 $f : \mathbb{Z}_n \to \mathbb{Z}_n$

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



<u>Modular exponentiation</u>

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

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

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 =
3019662633453665226674644411185277127204721722044543980521881984280643980698016315342127777985323
7655786915947633907457862442472144616346714598423225826077976000905549946633556169688641786953396
0040623713995997295449774004045416733136225768251717475634638402409117911722715606961870076297223
4159137526583857970362142317237148068590959528891803802119028293828368386437223302582405986762635
8694772029533769528178666567879514981999272674689885986300092124730492599541021908208672727813714
8522572014844749083522090193190746907275606521624184144352256368927493398678089550310568789287558
755227001418448833563517768339640003

g =
1721484410294542720413651217788953849637988183467987659847411571496616170507302662812929883501017
4348250308006877834103702727269721499966768323290540216992770986728538508742382941595672248624817
9949179397494476750553747868409726540440305778460006450549504248776668609868201521098873552043631
7965394509849072406890541468179263651065250794610243485216627272170663501147422628994581789339082
7991578201408649196984764863302981052471409215846871176739109049866118609117954454512573209668379
5760420560620966283259002319100903253019113331521813948039086102149370446134117406508009893347295
86051242347771056691010439032429058

Finn *a* når

$g^a \pmod{p} =$
4411321635506521515968448863968324914909246042765028824594289876687657182492169027666262097915382
0952830455103982849705054980427000258241321067445164291945709875449674237106754516103276658256727
2413603372376920980338976048557155564281928533840136742732489850550648761094630053148353906425838
5317698361559907392252360968934338558269603389519179121915049733353702083721856421988041492207985
6566434665604898681669845852964624047443239120501341277499692338517113201830210812184500672101247
2700988032756016626566167579963223042395414267579262222147625965023052419869061244027798941410432
6855174387813098860607831088110617

Solution

a =

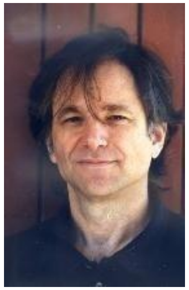
71893136149709653804503478677866573695060790720621260648699193249561437588126371185
81694154929099396752251787268346548051895320171079663652680741564200286881487888963
19895353311170236034836658449187117723820644855184055305945501710227615558093657781
93109639893698220411548578601884177129022057550866690223052160523604836233675971504
25938247630127368253363295292024736143937779912318142315499711747531882501424082252
28164641111954587558230112140813226698098654739025636607106425212812421038155501562
37005192231836155067262308141154795194735834753570104459663325337960304941906119476
181818583000946627658955269636154O6

It is easy to compute g^a (mod p) {0.016 s}, but it is computaionally infeasable to compute the exponent a from the g^a.

Diffie-Hellman Applications

- IPSec (IP Security):
  - IKE (Internet Key Exchange) is part of the IPSec protocol suite.
  - IKE is based on Diffie-Hellman Key Agreement.
- SSL/TLS:
  - Several variations of SSL/TLS protocol including:
    - Fixed Diffie-Hellman.
    - Ephemeral Diffie-Hellman.
    - Anonymous Diffie-Hellman.

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 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:

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

- 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)

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

- <u>Decryption:</u>
- Ciphererertext C = 32986, then M = $32986^{31613} \pmod{35011} = 19726$

Factoring record– December 2009

Find the product of

p = 33478071698956898786044169848212690817704794983713768568
912431388982883793878002287614711652531743087737814467999489

and

q= 36746043666799590428244633799627952632279158164343087642676032283815739666511279233373417143396810270092798736308917?

Answer:

n= 1230186684530117755130494958384962720772853569595334792197322452151726400507263657518745202199786469389956474942774063845925192557326303453731548268507917026122142913461670429214311602221240479274737794080665351419597459856902143413

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?

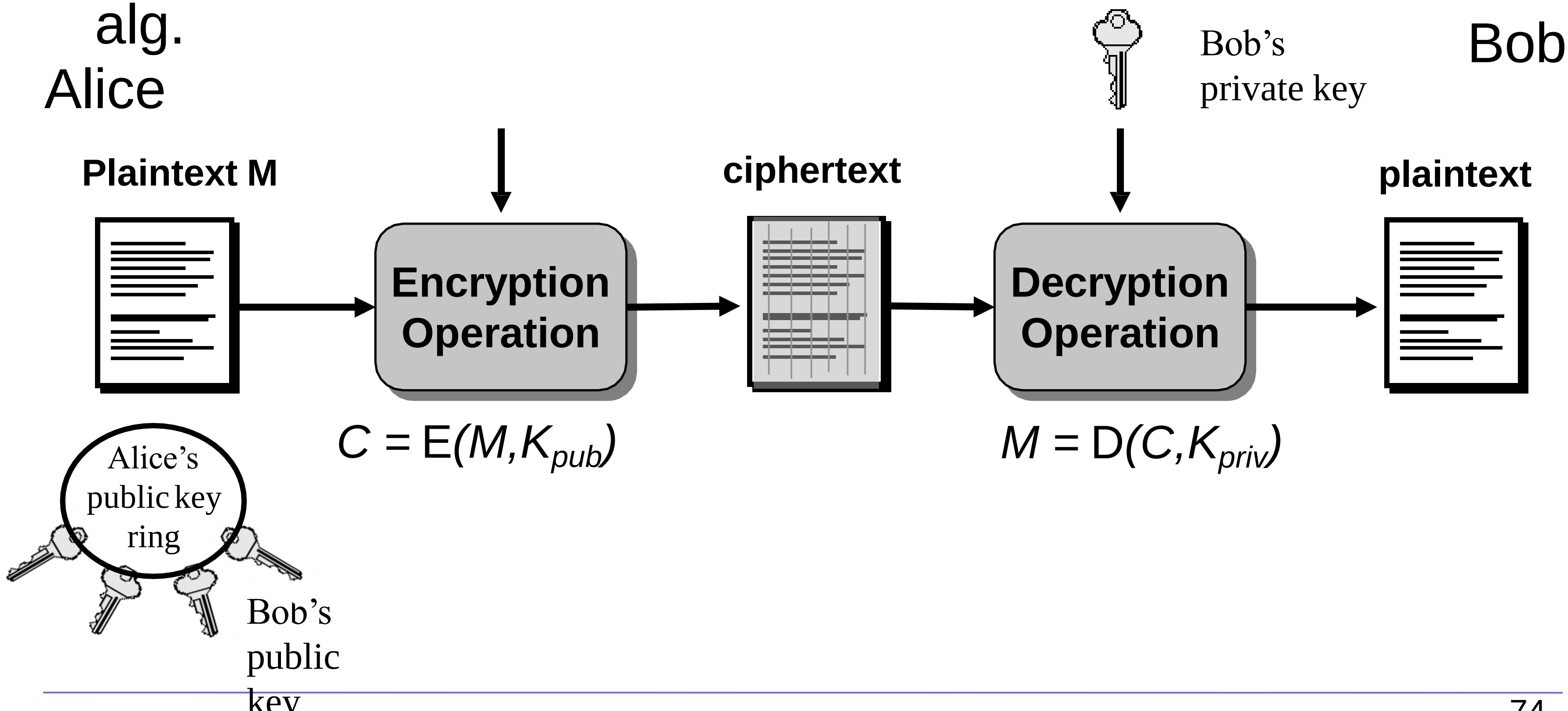
- Factoring using NFS-algorithm (Number Field Sieve).
- 6 mnd using 80 cores to find suitable polynomial.
- Solding from August 2007 to April 2009 (1500 AMD64-år).
- 192 796 550 * 192 795 550 matrise (105 GB).
- 119 days on 8 different clusters.
- Corresponds to 2000 years processing on one single core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

Asymmetric Encryption: Basic encryption operation

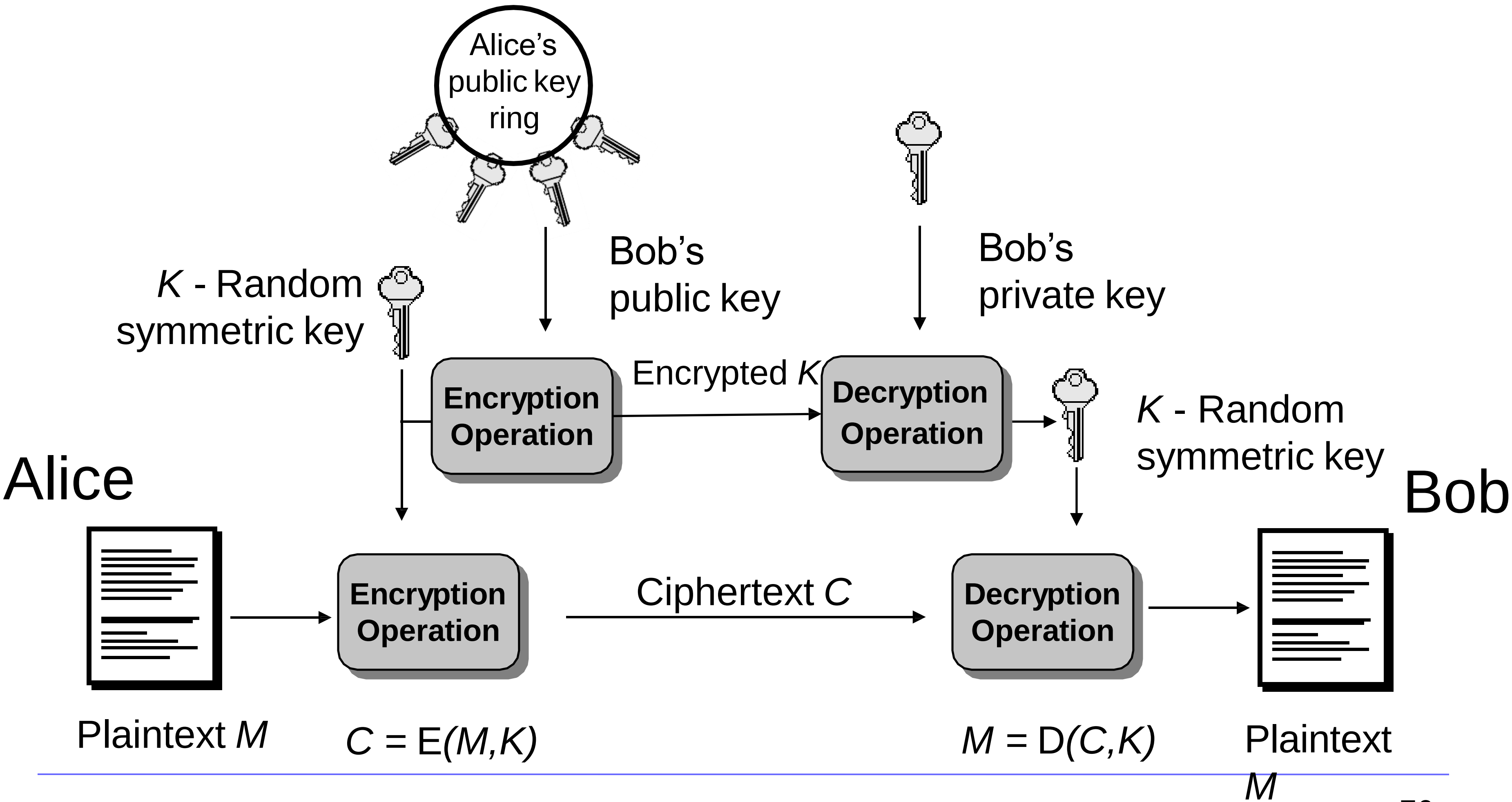
- In practice, large messages are not encrypted directly with asymmetric algorithms. Hybrid systems are used, where only symmetric session key is encrypted with asymmetric alg.



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

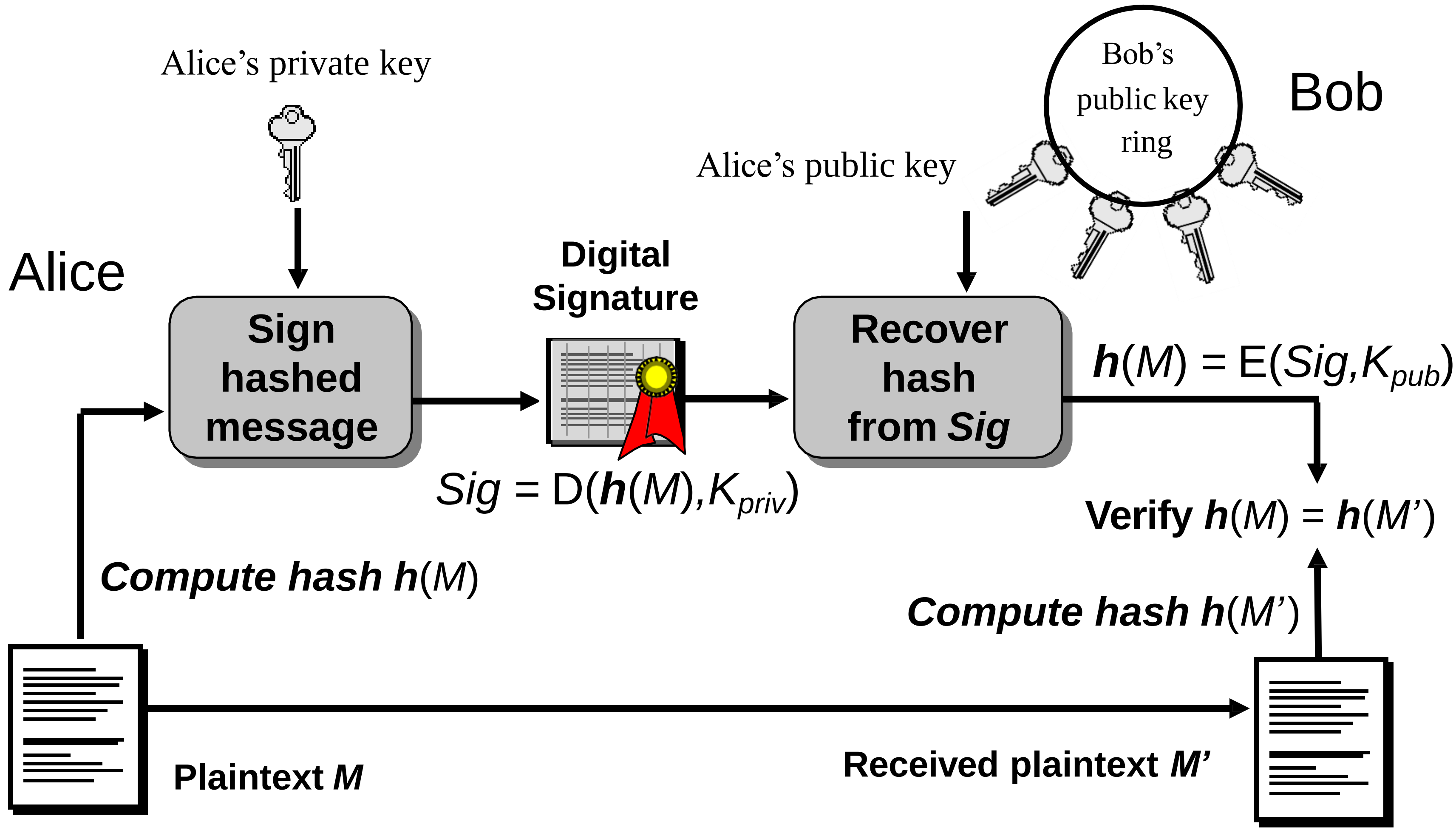


Digital Signatures

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.

security level	volume of water to bring to a boil	bit-lengths: symmetric key	bit-lengths: cryptographic hash	bit-lengths: 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!

MIT Technology Review

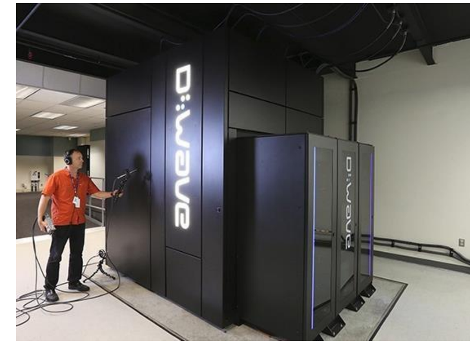
Topics+ Top Stories

Computing

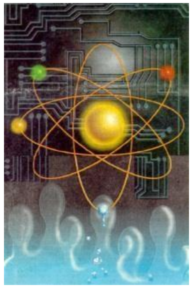
NSA Says It "Must Act Now" Against the Quantum Computing Threat

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



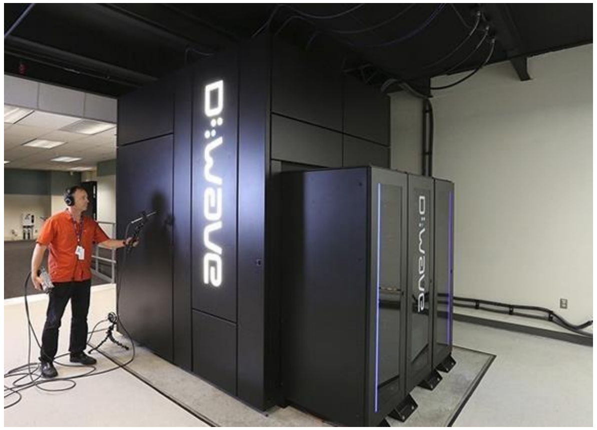
Quantum Computers



- 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.

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!

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

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	✓

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

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.

Brave new crypto world…………….

End of lecture