Cryptography

Symmetric and Asymmetric Key encryption (chapter 3.1, 4.4, 7.2-6, 9.1)

Cryptographic Hash functions (chapter 11.1)

Message Authentication Codes, MAC (chapter 12.1)

Key distribution and Certificates (chapter 15)

Diffie-Hellman key agreement (chapter 10.1)

Don’t invent your own ciphers!

- 1998, Bruce Schneier wrote:
  ”Anyone, from the most clueless amateur to the best cryptographer, can create an algorithm that he himself can't break. It's not even hard.”

- 1864, Charles Babbage wrote:
  "One of the most singular characteristics of the art of deciphering is the strong conviction possessed by every person, even moderately acquainted with it, that he is able to construct a cipher which nobody else can decipher.“

- Bruce Schneier, again:
  What is hard is creating an algorithm that no one else can break,
  even after years of analysis. And the only way to prove that is to subject the algorithm to years of analysis by the best cryptographers around.”

Cryptography and security

We can never say that an algorithm is secure. We can either show how it's insecure, or say something like:

- All of these smart people have spent lots of hours trying to break it, and they can't, but we don't know what a smarter person who spends even more hours analyzing it will come up with.

Cryptography and Engineering

- Cryptography is a science: Applied mathematics
  - Design of crypto-systems, understanding weaknesses, proving properties
- Applied cryptography is Engineering (security engineering)
  - Use of cryptographic building blocks to create secure systems
  - Almost all exploitable weaknesses come from the engineering
- For cryptography to work:
  - it needs to be written in software,
  - embedded in a larger software system,
  - managed by an operating system,
  - run on hardware,
  - be connected to a network,
  - and configured and operated by users

Semantics for description

- BAN logic is a semantics with a clear meaning when describing protocols
  - Burrows-Abadi-Needham logic
  - The book almost follows this semantic
- Components:
  - Individuals are named Alice (**A**), Bob (**B**), Charlie (**C**), etc.
  - Servers **S**
  - Private keys: $\mathbf{K_{AB}}$
  - Public key $\mathbf{PU_A}$ and private key $\mathbf{PR_A}$
  - Timestamps **T**
  - Nonces **N**, one-time arbitrary random numbers (numbers used once)
- Operations
  - **||** concatenation
  - ⊕ XOR
  - **E** Encrypt: E (K_{AB}, X) means encrypt X with key K_{AB} ← Alternative: $E_{K_{ab}}(X)$

Symmetric Key Encryption

- Same key is used to encrypt and decrypt (symmetric)
- Pseudorandom permutations – no mathematical algorithms
- **Block ciphers** and **stream ciphers**

Symmetric Key Encryption

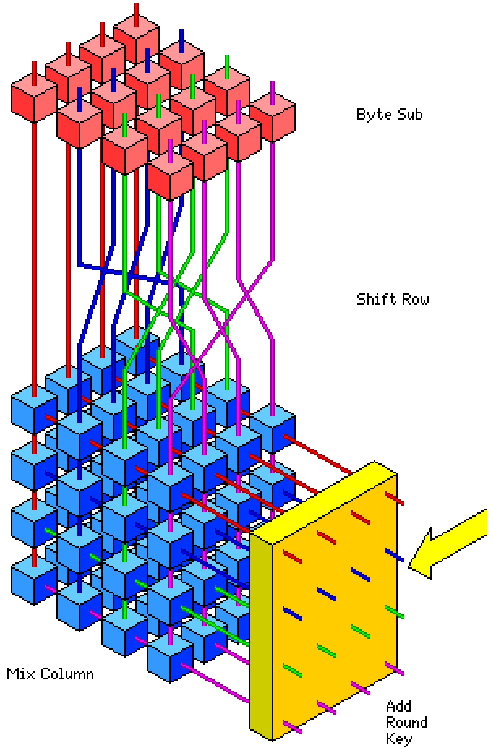
- **DES** (Data Encryption Standard)
  - Designed by IBM 1975, Adopted by NIST* 1977
  - Criticized for key length (64→ 56) and mysterious “S-boxes”
  - Turned out to have protection against differential cryptanalysis (found 1990)
  - Probably more effort is spent on cracking DES than on all other ciphers together
  - Key length is a major problem: 56-bit keys can be cracked in less than a day
- **3-DES** (repeating DES three times with two or three different keys)
  - 3-DES with three keys assumed secure until 2030 but way too slow!
  - Disallowed by NIST after 2023

- **AES** (Advanced Encryption Standard)
  - Replaced DES 2001
  - Result of an official competition – Belgian winner
  - Key lengths: 128, 192 or 256 bits
  - Brute force decryption: if DES takes 1 second, AES-128 takes 149 trillion years, AES-256 would take 10^{52} years
- **RC4**
  - Ron’s code 4; Rivest cipher 4; …; ?
  - Stream cipher
  - RC4 is weak but fast – avoid it!

*NIST = National Institute of Standards and Technology, US, formerly NBS

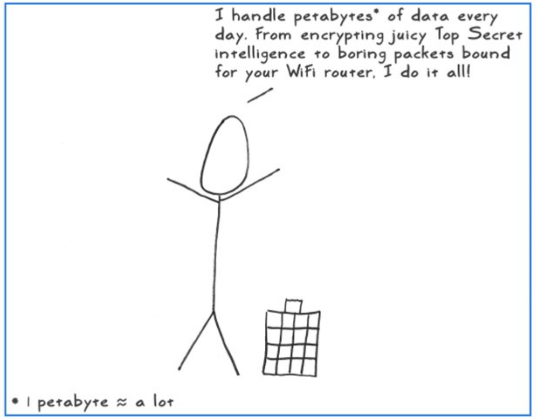
The AES Block Cipher [FIPS-197]

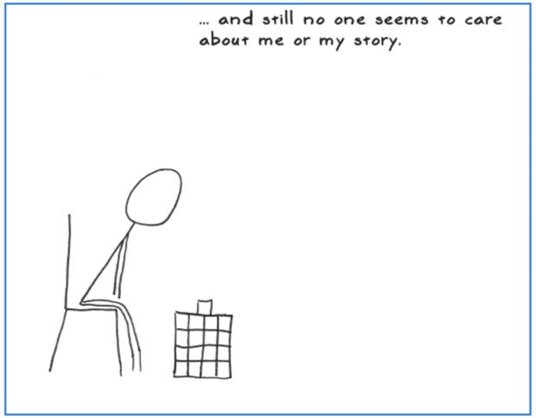
- AES = Advanced Encryption Standard
  - International competition (NIST)
  - Call for candidates 1997
  - Winners from Belgium
  - Official standard 2001
- 4x4 = 16 byte data blocks
- Permutations and substitutions
  - Not completely symmetric (encr != decr.)
- Supports 128, 192 and 256 bit **key lengths**
- Supported in most applications:
  - TLS, IPsec, SSH, WPA2, ZigBee, WiMAX, …

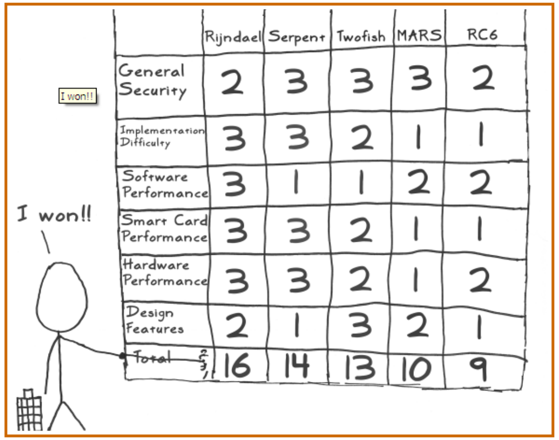


AES explained – Good!

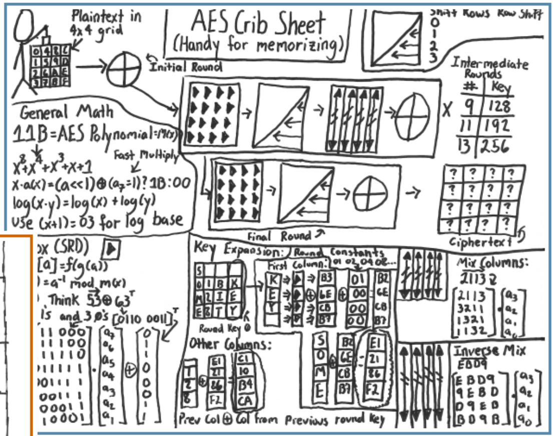
https://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html


I handle petabytes* of data every day. From encrypting juicy Top Secret intelligence to boring packets bound for your WiFi router, I do it all!
* 1 petabyte ≈ a lot



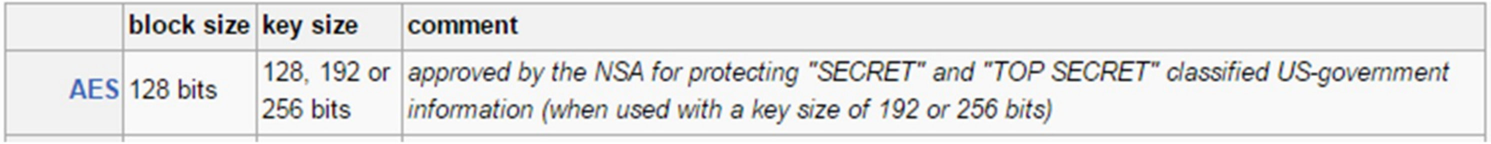


	Rijndael	Serpent	Twofish	MARS	RC6
General Security	2	3	3	3	2
Implementation Difficulty	3	3	2	1	1
Software Performance	3	1	1	2	2
Smart Card Performance	3	3	2	1	1
Hardware Performance	3	3	2	1	2
Design Features	2	1	3	2	1
Total	16	14	13	10	9

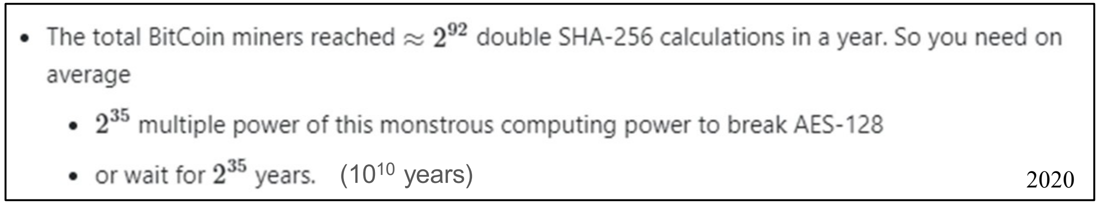


AES attacks so far…

- The first key recovery attack took 10 years to find (Aug 2011)
  - "Biclique Cryptanalysis of the Full AES," by Bogdanov, Khovratovich and Rechberger
  - AES-128 with computational complexity $2^{126.0}$
  - AES-192 with computational complexity $2^{189.9}$
  - AES-256 with computational complexity $2^{254.3}$
- Requires 10^{13} TB of storage – so theoretical attack only
- AES is trusted:



	block size	key size	comment
AES	128 bits	128, 192 or 256 bits	*approved by the NSA for protecting "SECRET" and "TOP SECRET" classified US-government information (when used with a key size of 192 or 256 bits)*



- The total BitCoin miners reached $\approx 2^{92}$ double SHA-256 calculations in a year. So you need on average
  - 2^{35} multiple power of this monstrous computing power to break AES-128
  - or wait for 2^{35} years.   (10^{10} years)

2020

Exhaustive key search for ciphers

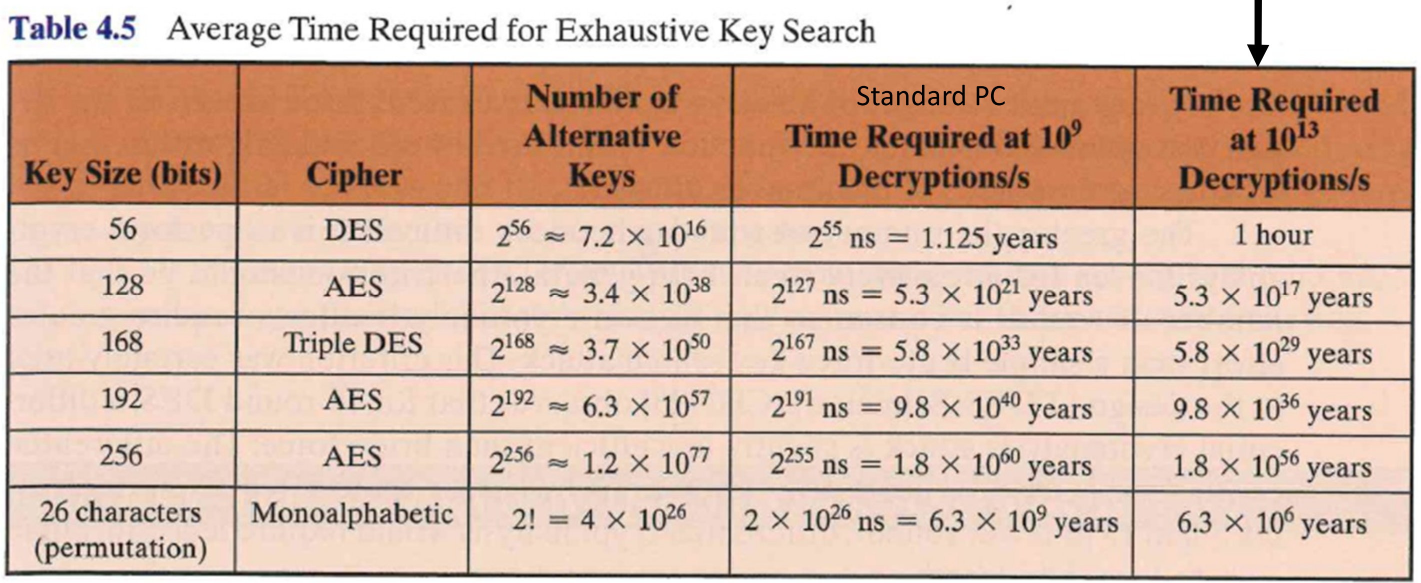


Table 4.5 Average Time Required for Exhaustive Key Search

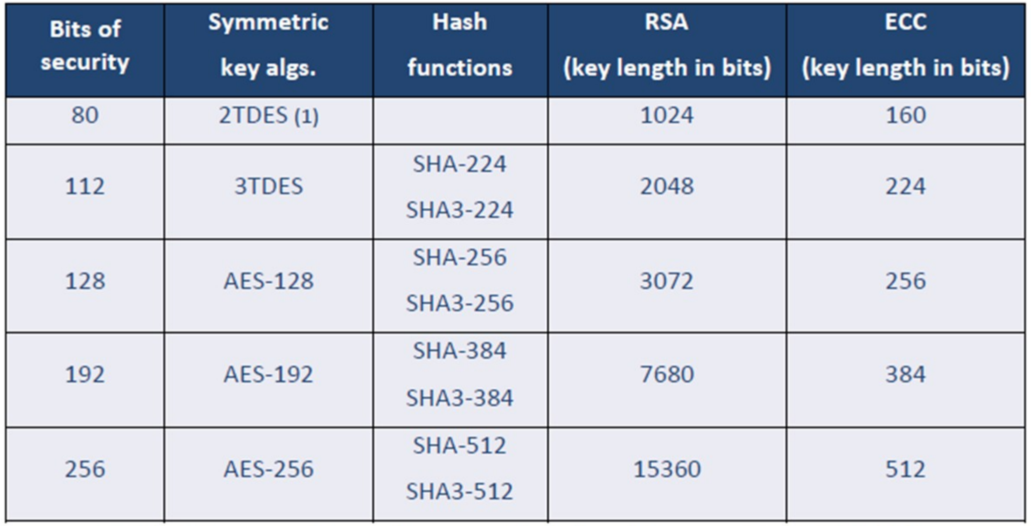
Key Size (bits)	Cipher	Number of Alternative Keys	Standard PC Time Required at 10^9 Decryptions/s	Time Required at 10^{13} Decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns $= 5.3 \times 10^{21}$ years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns $= 5.8 \times 10^{33}$ years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns $= 9.8 \times 10^{40}$ years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns $= 1.8 \times 10^{60}$ years	1.8×10^{56} years
26 characters (permutation)	Monoalphabetic	$2! = 4 \times 10^{26}$	2×10^{26} ns $= 6.3 \times 10^{9}$ years	6.3×10^{6} years

https://securityintelligence.com/the-cracken-the-evolution-of-password-cracking/

European Payments Council

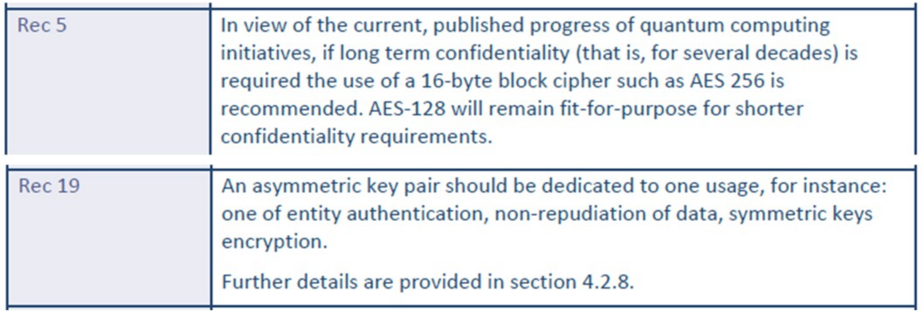
GUIDELINES ON CRYPTOGRAPHIC ALGORITHMS USAGE AND KEY MANAGEMENT

Version 13.0
13 March 2024



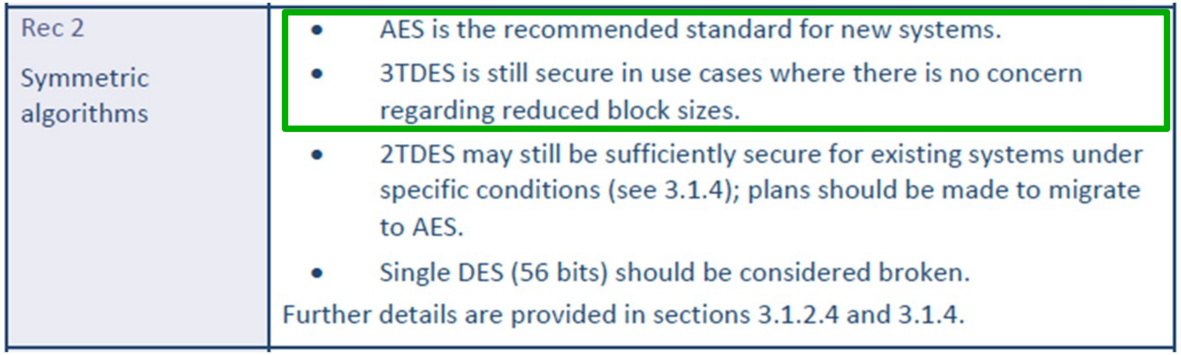
Bits of security	Symmetric key algs.	Hash functions	RSA (key length in bits)	ECC (key length in bits)
80	2TDES (1)		1024	160
112	3TDES	SHA-224 SHA3-224	2048	224
128	AES-128	SHA-256 SHA3-256	3072	256
192	AES-192	SHA-384 SHA3-384	7680	384
256	AES-256	SHA-512 SHA3-512	15360	512

Table 5: Comparable security strengths (adapted from section 5.6 of [130])



Rec 5	In view of the current, published progress of quantum computing initiatives, if long term confidentiality (that is, for several decades) is required the use of a 16-byte block cipher such as AES 256 is recommended. AES-128 will remain fit-for-purpose for shorter confidentiality requirements.
Rec 19	An asymmetric key pair should be dedicated to one usage, for instance: one of entity authentication, non-repudiation of data, symmetric keys encryption. Further details are provided in section 4.2.8.

NIST disallows keys shorter than 112 bits [2019]



Rec 2 Symmetric algorithms	• AES is the recommended standard for new systems. • 3TDES is still secure in use cases where there is no concern regarding reduced block sizes. • 2TDES may still be sufficiently secure for existing systems under specific conditions (see 3.1.4); plans should be made to migrate to AES. • Single DES (56 bits) should be considered broken. Further details are provided in sections 3.1.2.4 and 3.1.4.

Many good recommendations can be found in this document!

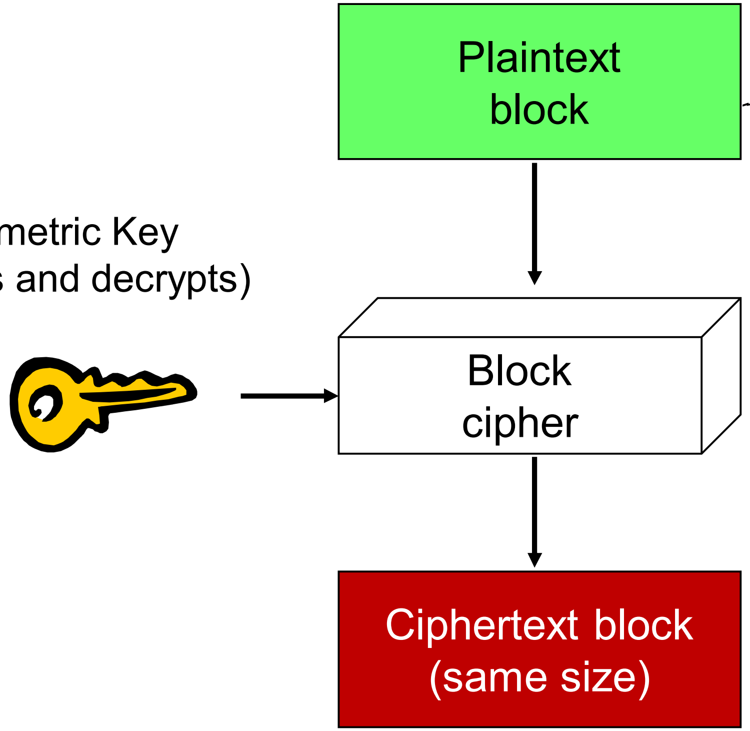
https://www.europeanpaymentscouncil.eu/document-library/guidance-documents/guidelines-cryptographic-algorithms-usage-and-key-management-0

Block ciphers and Modes

- Block ciphers act on one block, typically 8 or 16 bytes
- Modes describe how the block cipher is applied to a large amount of data
  - Initialization vector (IV)
  - Feedback between blocks
  - Padding
- Examples:
  - ECB, CBC, CTR, …
- Cipher modes also have weaknesses and strengths

ECB – Electronic Code-Book mode

Symmetric Key
(encrypts and decrypts)

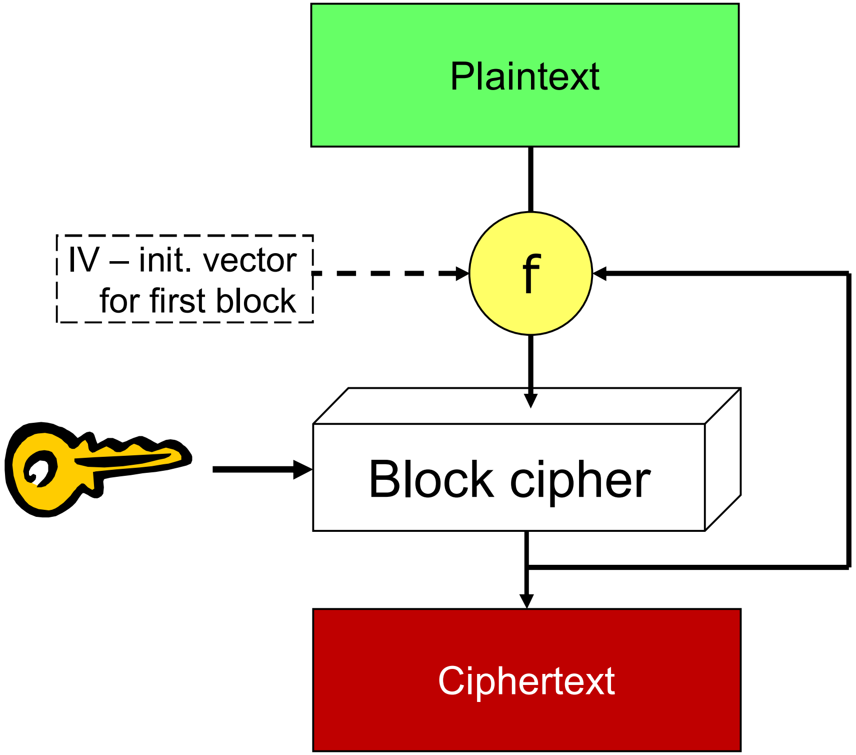


Block size depends on cipher:
DES=64 bits, AES=128 bits, …
Independent of key length.

The algorithm is publicly known!

Potential problem: same plaintext always results in the same ciphertext (“**block effect**”)

CBC – Cipher Block Chaining mode

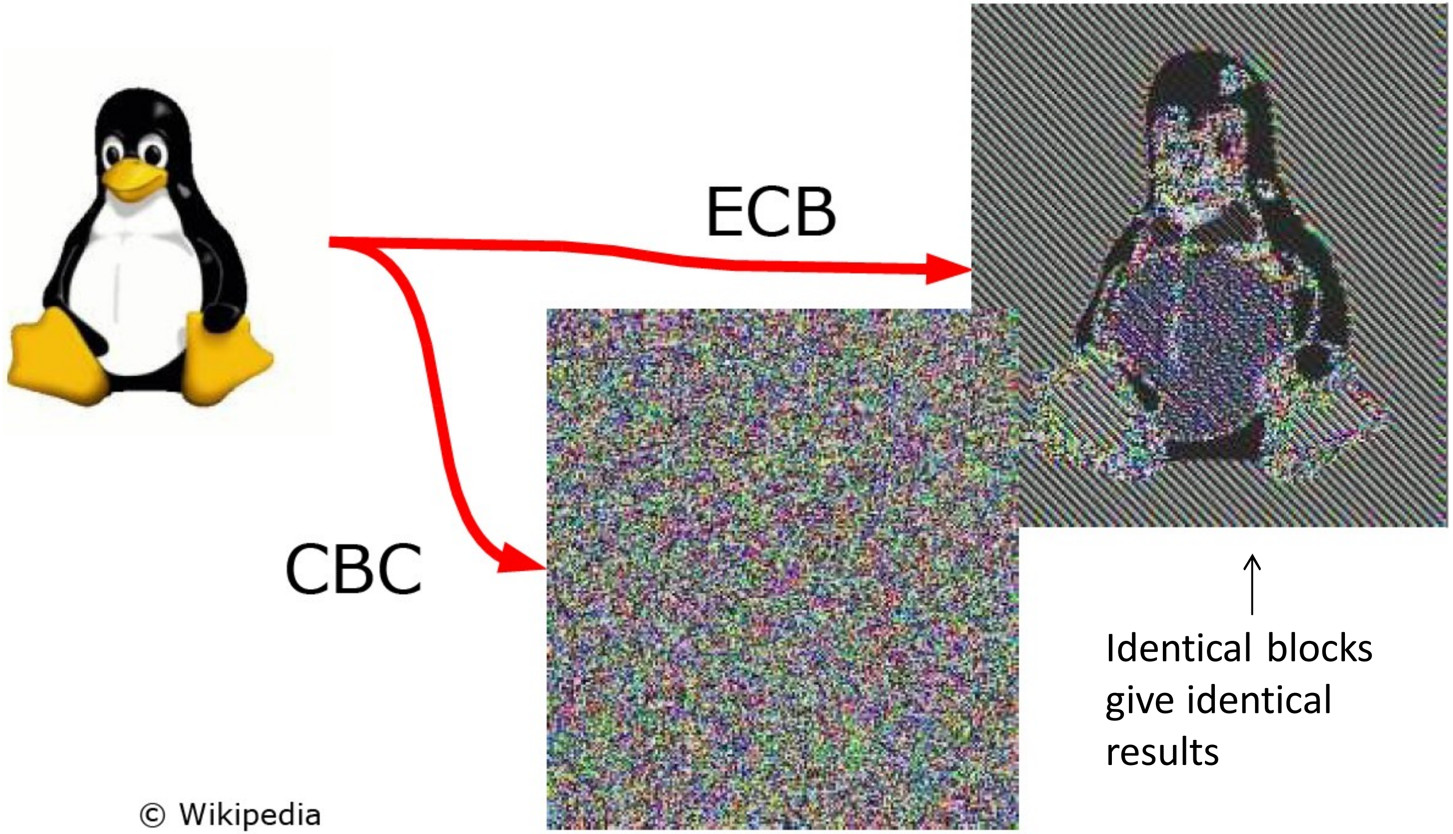


Identical plaintexts have different ciphertexts

Blocks can be decrypted (but not encrypted) in **parallel**

May not always be practical, for example hard disk encryption: *modification of one block affects all the following*

ECB vs. CBC

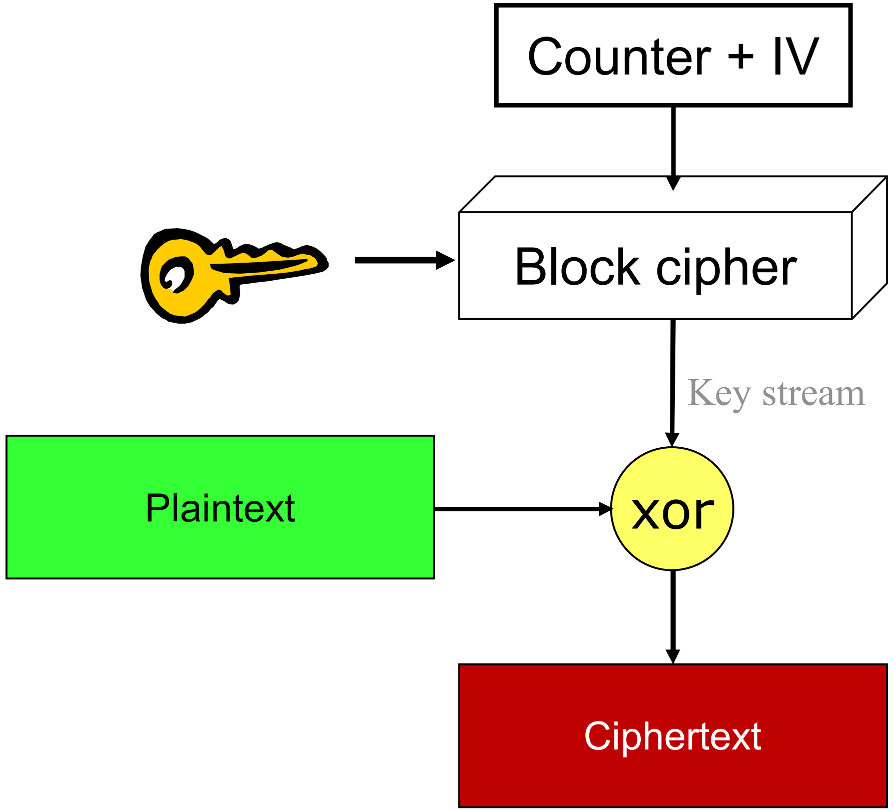


↑

Identical blocks give identical results

© Wikipedia

CTR – Counter mode



Identical plaintexts have different ciphertexts

Blocks do not depend on each other → calculations can be done in parallel (e.g. in multi-CPU system)

IV (a nonce) should be used. Reuse of the same key stream is catastrophic. Why?

Key stream can be pre-computed to reduce packet latency

Chapter 7.6

Stream ciphers

Operates on one digit or bit at a time,
output stream is XOR:ed with data to encrypt:

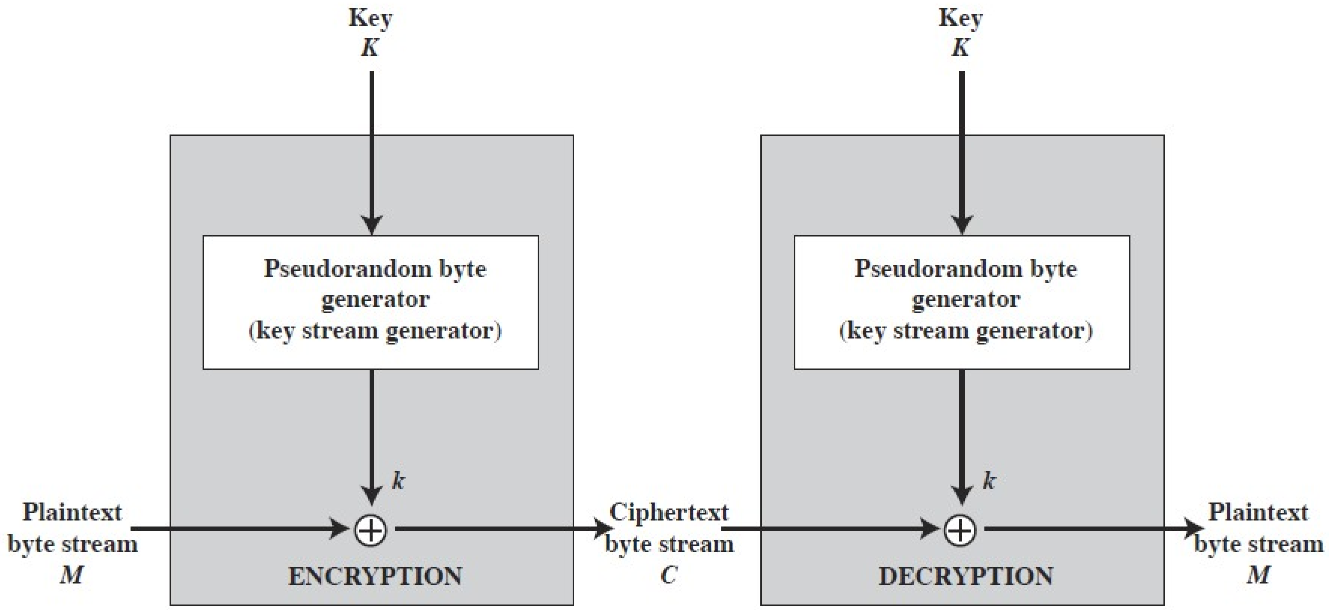
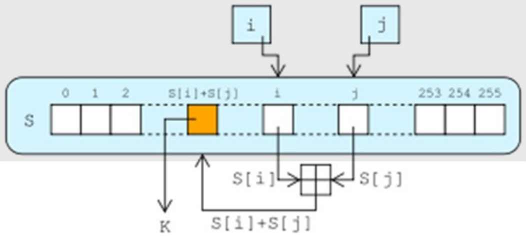

Key
K
Pseudorandom byte generator (key stream generator)
k
Plaintext byte stream M
ENCRYPTION
Ciphertext byte stream C
Key
K
Pseudorandom byte generator (key stream generator)
k
DECRYPTION
Plaintext byte stream M

Figure 7.5 Stream Cipher Diagram

RC4



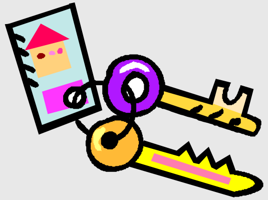
- RC4 is a single self-modifying rotor machine, but modifies itself very slowly
  - The rotor leaks information about its internal state with every output byte, but the self-modifying structure makes it hard to analyze

- Many attacks and problems exist:
  - 2016: Algorithms exist to find multiple keys generating the same output stream, for example:

```
–   1st key: 1C 50 22 84 5B 74 EC B6 62 F5 95 7E 04 5F 5A 08 4A 6B C7 2C (2D) 02 95
–   2nd key: 58 76 1D 95 DC F6 E8 13 00 47 4E D2 41 C1 01 4A 57 7D C7 BD (BE) 02 68
–   3rd key: 8F E2 82 5D 51 74 8A BF 28 1D E1 23 0A 8E 3C FE 98 5C C7 21 (22) 02 6A
```

- Don’t use RC4 in new designs!

https://eprint.iacr.org/2016/1001



Public key (asymmetric) encryption

One key is used to encrypt, another to decrypt

Asymmetric key encryption

- One key is used to encrypt, the other to decrypt
- One key can be public – the other is secret
- Based on mathematically hard problems
- Slower than symmetric systems because of the large numbers involved
- RSA – Rivest, Shamir, Adleman (Patented 1983-2000)
  - Factorization of very large numbers
  - 1024 bit keys and up, $2^{1024} = 10^{308}$ i.e. >300 digit numbers
  - Really discovered by GCHQ 1973 but kept secret until 1997…
  - Use 2048-bit or longer keys in sensitive applications (>600 digits)
  - Similar in strength as 112-bit symmetric ciphers
- ECC – Elliptic Curve Cryptography (1985)
  - Public key cryptography based on elliptic curves
  - A curve is defined and keys created from it
  - Standardized curves exist: NIST, IETF, Microsoft, …
  - Short keys, 384 bits
  - Hard to know what curves are secure!
  - Advantage: Shorter keys, faster in hardware, faster signature generation
  - Disadvantage: some curves patented, signature verification with ECC slower than RSA
- McEliece (1978)
  - Very long keys – 512 kbit, (NP hard problem)
  - Fast
- Lattice based systems

RSA strength

829-bit RSA was cracked April 2020

- Corresponds to the RSA-250 cryptographic challenge. This integer is the product of two prime numbers, each with 125 decimal digits
- 2700 years of running on a powerful computer
- Done on tens of thousands of machines around the world over the course of a few months

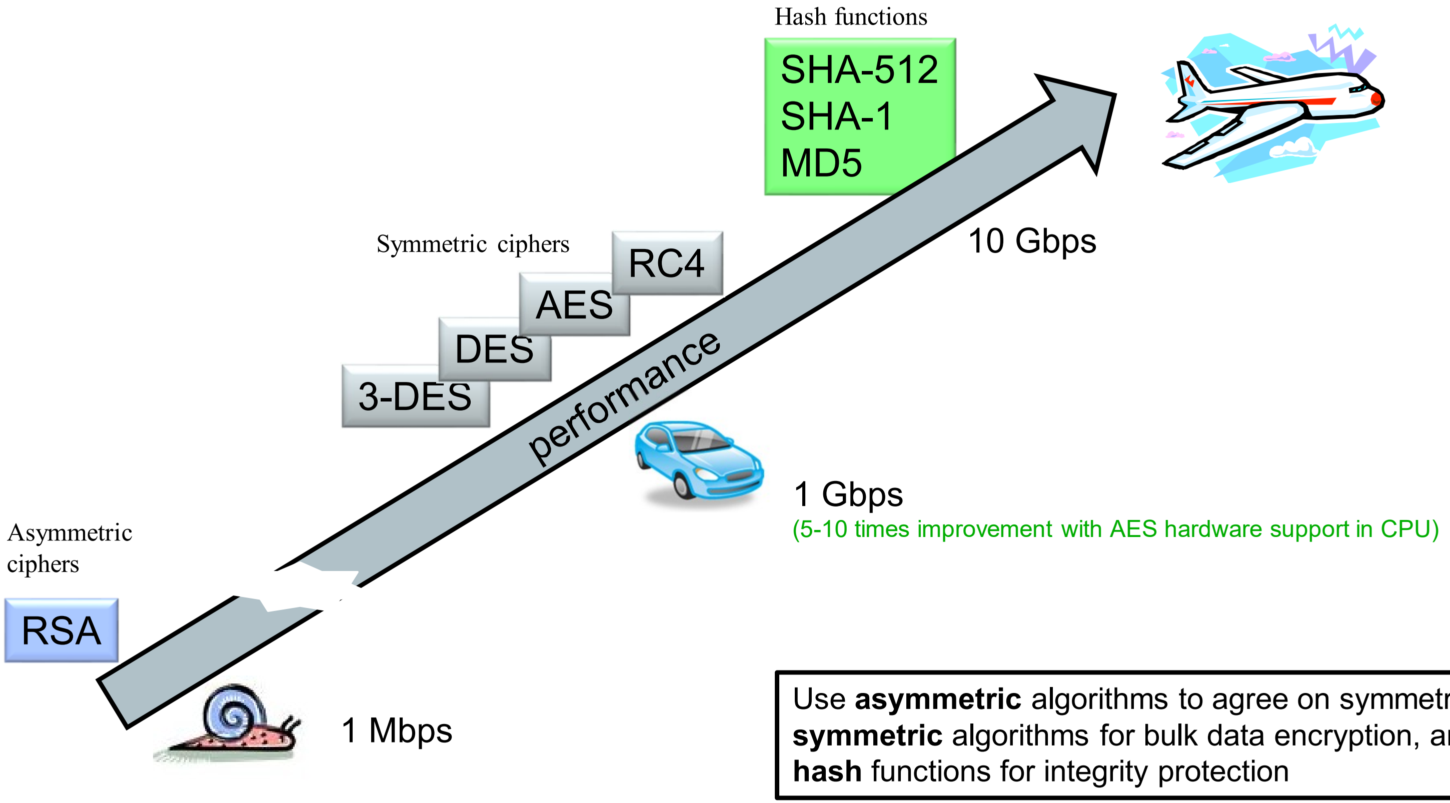
Asymmetric key encryption

- One key is normally made public
(“Public key encryption”)

You decide whether it is the encryption or decryption key that is public!

1. **Encryption key public**: everyone can send encrypted messages to owner of private key
2. **Decryption key public**: only one can encrypt, everyone can verify that the secret key has been used. Can be used to sign documents and data.

Relative performance



Hash functions

SHA-512
SHA-1
MD5

10 Gbps

Symmetric ciphers

RC4

AES

DES

3-DES

performance

1 Gbps

(5-10 times improvement with AES hardware support in CPU)

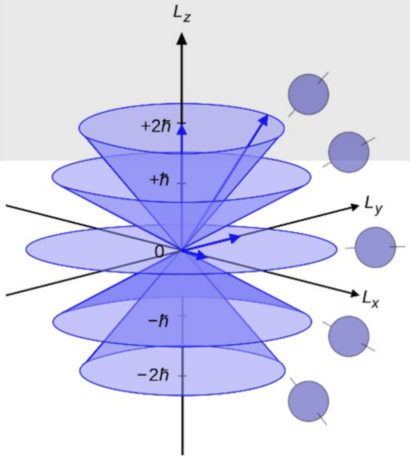
Asymmetric ciphers

RSA

1 Mbps

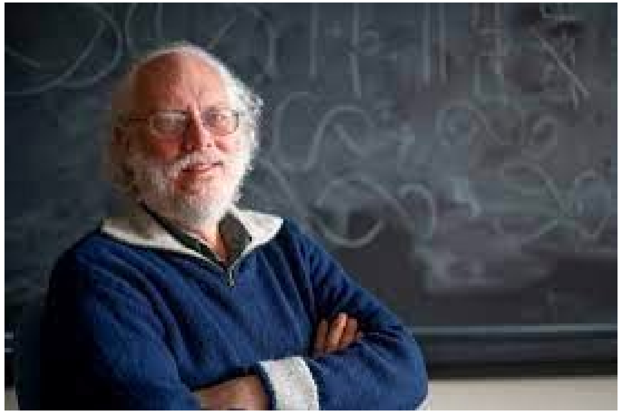
Use **asymmetric** algorithms to agree on symmetric keys, **symmetric** algorithms for bulk data encryption, and **hash** functions for integrity protection

Post-quantum Cryptography



- Complexity of attacking a cipher using brute force: $\mathbf{2^n}$ (n is the key length)
- Cryptanalytic attacks are shortcuts – they reduce effective key length
- Quantum computing is extremely expensive, not very capable, and very few people know how to program them in a meaningful way. This will likely remain the case for at least a decade* [2019]
- Will halve the key size for symmetric ciphers (Grover's algorithm) → use 256 bit keys
- If factorization problem is solved (Shore's algorithm)
  - Diffie-Hellman, RSA and ECC public key cryptography broken
  - McEliese and lattice based algorithms untouched
- Even if Quantum computers are not common today, we may want to consider someone "record now, crack later"

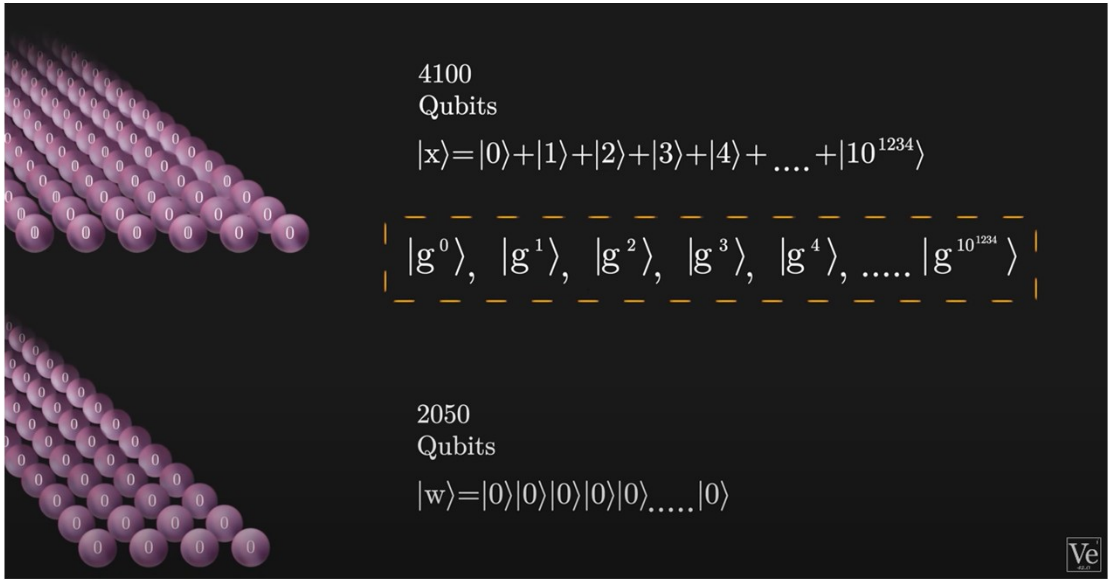
Peter Shor – Professor of applied mathematics, MIT



- He was awarded the **Nevanlinna Prize** (now called IMU Abacus Medal) **at the 23rd International Congress of Mathematicians** in 1998 for giving a Quantum algorithm to factor integers in polynomial time and the Gödel Prize in 1999.

- Shor was awarded **the 2023 Breakthrough Prize in Fundamental Physics** for "foundational work in the field of quantum information.

Post-quantum cryptography explained


4100
Qubits
$|x\rangle = |0\rangle + |1\rangle + |2\rangle + |3\rangle + |4\rangle + \ldots + |10^{1234}\rangle$
$|g^0\rangle, |g^1\rangle, |g^2\rangle, |g^3\rangle, |g^4\rangle, \ldots |g^{10^{1234}}\rangle$
2050
Qubits
$|w\rangle = |0\rangle|0\rangle|0\rangle|0\rangle|0\rangle \ldots |0\rangle$
Ve

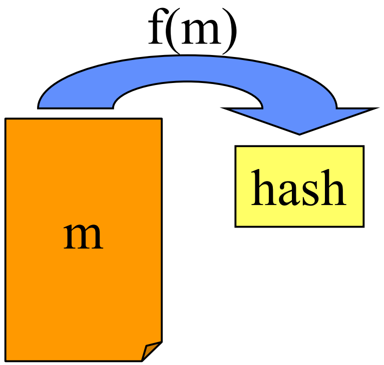
https://www.youtube.com/watch?v=-UrdExQW0cs

0:00—6:00	Explanation of RSA and asymmetric encryption
6:00 – 18:00	Quantum cryptography
18:00 – 23:00	ECC and quantum safe algorithms

Cryptographic Hash Functions

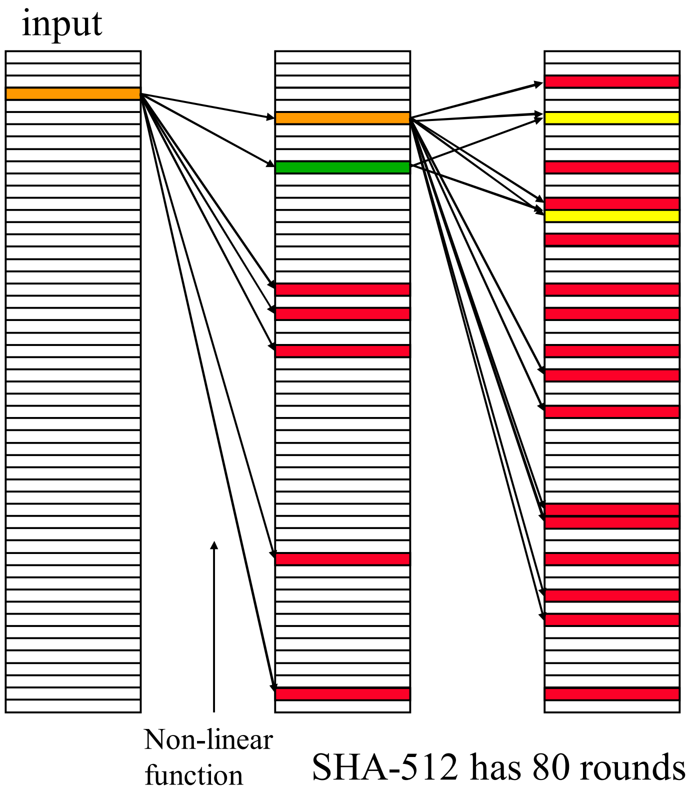
- **Chapter 11**
- Cryptographic operations that cannot be reversed

Cryptographic hash functions



- Input: arbitrary length bit-string
  Output: fixed length bit-string
  - Note: not a one-to-one mapping, output space typically 128 bits
- Requirements:
  - Computationally efficient
  - Must be repeatable (same input → same output)
  - Output space should be efficiently used
  - Given a hash *h* it should be infeasible to find any message *m* such that *h = hash(m)*
  - Preimage resistant: Given an input m_1 it should be difficult to find another input m_2
  - Collision resistance: Infeasible to find two inputs resulting in the same hash
  - Any change in the input should result in a new unpredictable hash: pseudo-randomness
- Hash functions are not based on mathematical foundations – problematic

Hash functions



Even one changed bit gives a completely different result (avalanche effect):

```
md5(“hello”) =
5d41402abc4b2a76b9719d911017c592

md5(“Hello”) =
8b1a9953c4611296a827abf8c47804d7
```

Common Hash functions

- **MD5** – Message Digest 5 (RFC 1321, 1992)
  - 128-bit (16 byte) message digest
  - Don’t use in new designs – broken !!
- **SHA-1** – Secure Hash Algorithm
  - Designed by NSA, became NIST standard 1995: FIPS-180-2
  - 160-bit (20 byte) message digest
  - Don’t use in new designs – broken !!
- **SHA-2** (family name for SHA-224, SHA-256, SHA-384 and SHA-512)
  - Still used and is long enough for now (SHA-256 widely used in Bitcoin blockchain)
- **SHA-3** – next generation hash functions
  - Keccak - winner of open competition (NIST standard 2015)
  - Arbitrary digest size (standard proposes 224, 256, 384 and 512 bit digests)
  - Considered more secure than SHA-2

MD5 and SHA-1 are “broken”

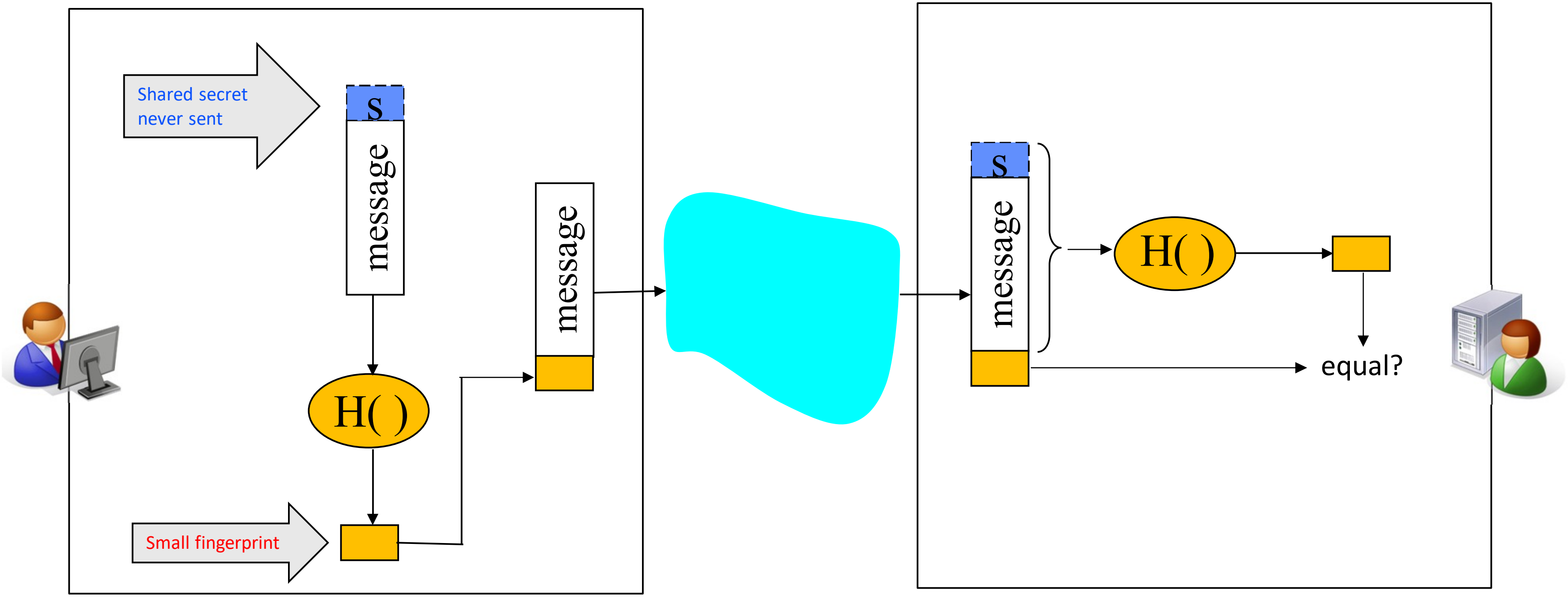
- CWI and Google announced **2017** that they have created two PDF documents with identical hashes (“collision attack”) with SHA-1

- **Chrome** web browser does not accept SSL/TLS certificates signed with SHA-1 as of Jan 2017, such **certificates are no longer issued**

- MD5 only requires 2^{33} searches (< 1 minute) to generate a collision
  - Note: If you sign something with MD5 that someone else has created (executable file, certificate, PDF file, JPEG image, etc.), something else with the same hash may exist!

- Researchers demonstrated Jan 2020 that fake certificates can be created (“**chosen** preimage attack”)
  - Cost: $11,000 for 900 NVIDIA GTX 1060 graphics cards
  - Two months of computation
  - https://eprint.iacr.org/2020/014.pdf [Usenix Security 2020]

Keyed hash for integrity protection



Authenticates sender and verifies message integrity

Faked messages cannot be created. Note that **encryption is not needed**!

HMAC – Keyed hash for message authentication

- HMAC is a standard for message authentication (MAC) – RFC 2104
  - HMAC = Keyed-**H**ashing for **M**essage **A**uthenti**c**ation
  - Used in TLS, SSH and IPSec

$$\boxed{\mathrm{HMAC}_K(\mathrm{text}) = \mathrm{hash}(\ K \oplus \mathrm{opad}\ ||\ \mathrm{hash}(\ K \oplus \mathrm{ipad}\ ||\ \mathrm{text}\ ))}$$

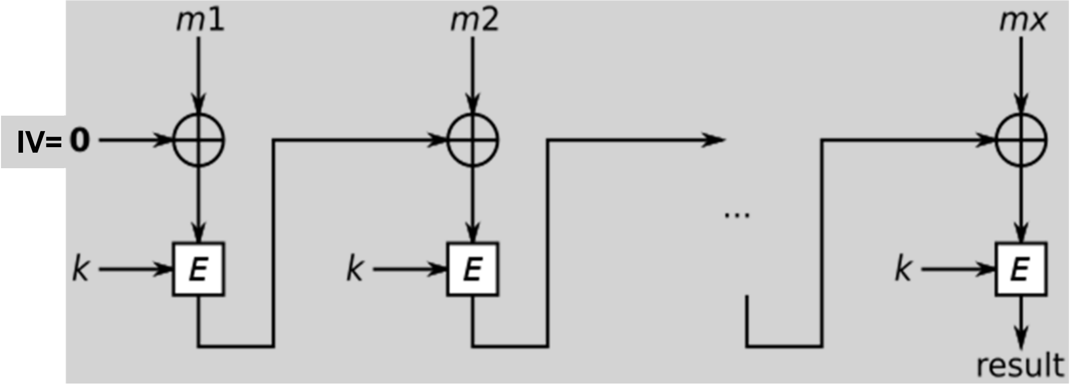
  - "||" means concatenation
  - **K** is a **secret key** padded with zeros to block size of hash function (e.g. 512 bits)
  - Key length same as Hash length: 128 (MD5), 160 (SHA-1), 256 (SHA-256), etc.
  - ipad = 0x363636…
  - opad = 0x5c5c5c... (length same as block size)

- Often called HMAC_MD5, HMAC_SHA1, HMAC-SHA256, etc.

- HMAC is today not affected by weaknesses in MD5 and SHA-1

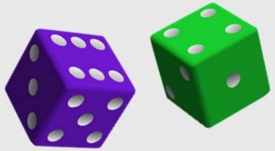
CBC MAC – Cipher Block Chaining MAC [RFC3610]

- Alternative to using a hash function:
  encrypt with a block cipher in CBC mode
- Each block depends on all previous blocks
- Only secure for fixed length messages
  - knowing two messages and their hashes, a third longer message with the same hash can be constructed without knowing the key
  - A solution can be to include the length in the first message



Wikipedia

Random number generation



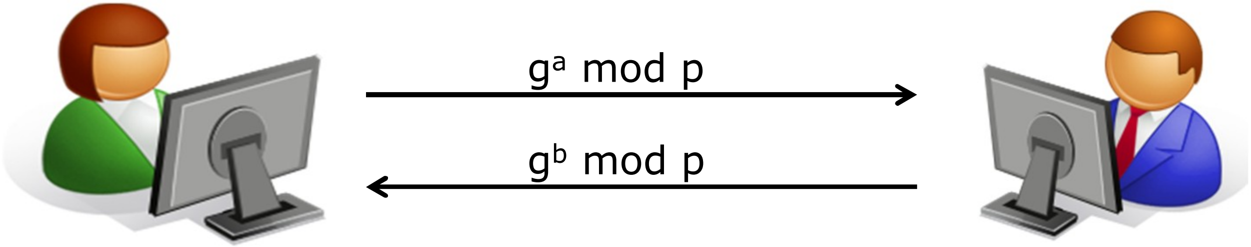
- A normal Pseudo-random number generator (PRNG) is not the same as a *cryptographic PRNG*
  - A normal PRNG may produce good random numbers
  - But the sequence may be predictable given some numbers – for us, this matters!
- Truly random data is needed to seed the PRNG
  - **Any hash function or block cipher can then be used** to generate arbitrary long pseudo-random output
  - If no more random data is added, this seed must be kept secret
- Desirable is to have at least 128 bits of entropy ($2^{128} = 10^{38}$ guesses)
  - Biggest problem is to collect this random data and know when entropy is enough
- 2012: Intel and AMD started to ship CPUs with an *RdRand* (read random) instruction
  - High speed, good RNG capable of multi Gbps performance
  - 256-bit entropy, hardware generated and uses AES-CBC-MAC
  - Political problem? Linux uses RdRand and adds other non-Intel controlled sources for /dev/random

Diffie-Hellman Key Agreement

g^x mod p

Chapter 10.1

Diffie-Hellman Key Agreement (1976)

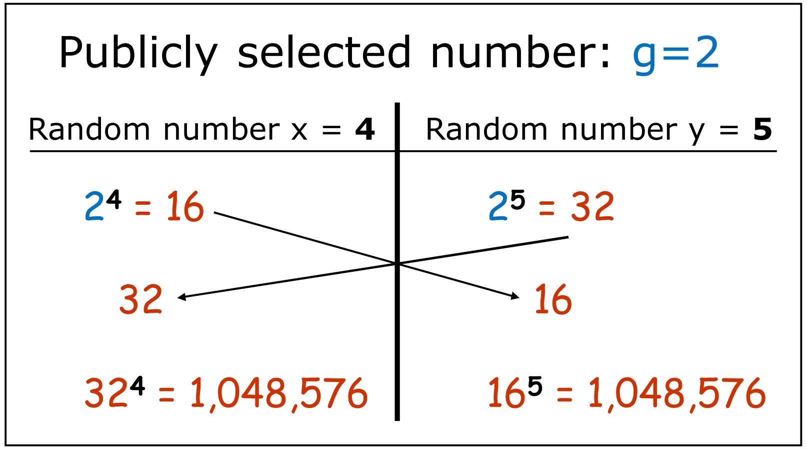


- Allows two parties not knowing each other to agree on a secret key using an insecure channel
- Relies on the fact that $(g^x)^y = (g^y)^x$
- Does not offer protection against MITM attacks
  - We only share a key now – we don’t know with whom!
  - Authentication still needed e.g. with public/private keys such as RSA
- DOS attacks: a server may be fooled to do lots of key calculations
- Eliptic Curve Diffie-Hellman (ECDHE) key agreement is faster and gaining in popularity

Diffie-Hellman Key Agreement

- Allows two parties not knowing each other to agree on a secret key using an insecure channel
- D-H was invented before RSA and public-key cryptography (1978)
- It was discovered by GCHQ in the U.K. earlier but kept secret!
- Relies on the fact that $(g^x)^y = (g^y)^x$

First a simplified example:



Publicly selected number: $g=2$

Random number $x = \mathbf{4}$ | Random number $y = \mathbf{5}$

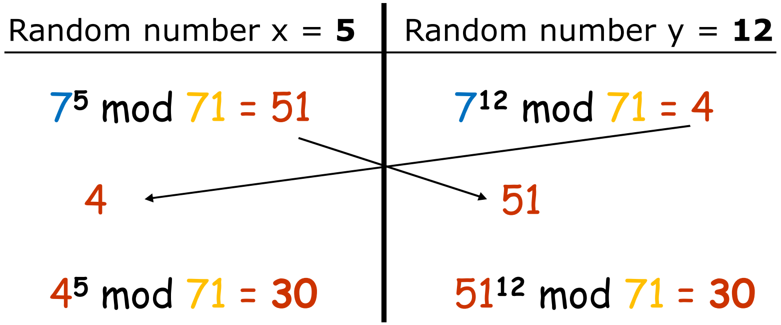
$2^4 = 16$ | $2^5 = 32$

32 | 16

$32^4 = 1,048,576$ | $16^5 = 1,048,576$

Diffie-Hellman Key Agreement

Prime selected: p = 71
A primitive root of 71 is 7 (g)



Random number x = **5**

Random number y = **12**

7^5 mod 71 = 51

7^{12} mod 71 = 4

4

51

4^5 mod 71 = **30**

51^{12} mod 71 = **30**

Note that an attacker knows p, g, 4 and 51.
Doing exponentiation is simple, going backward is hard.

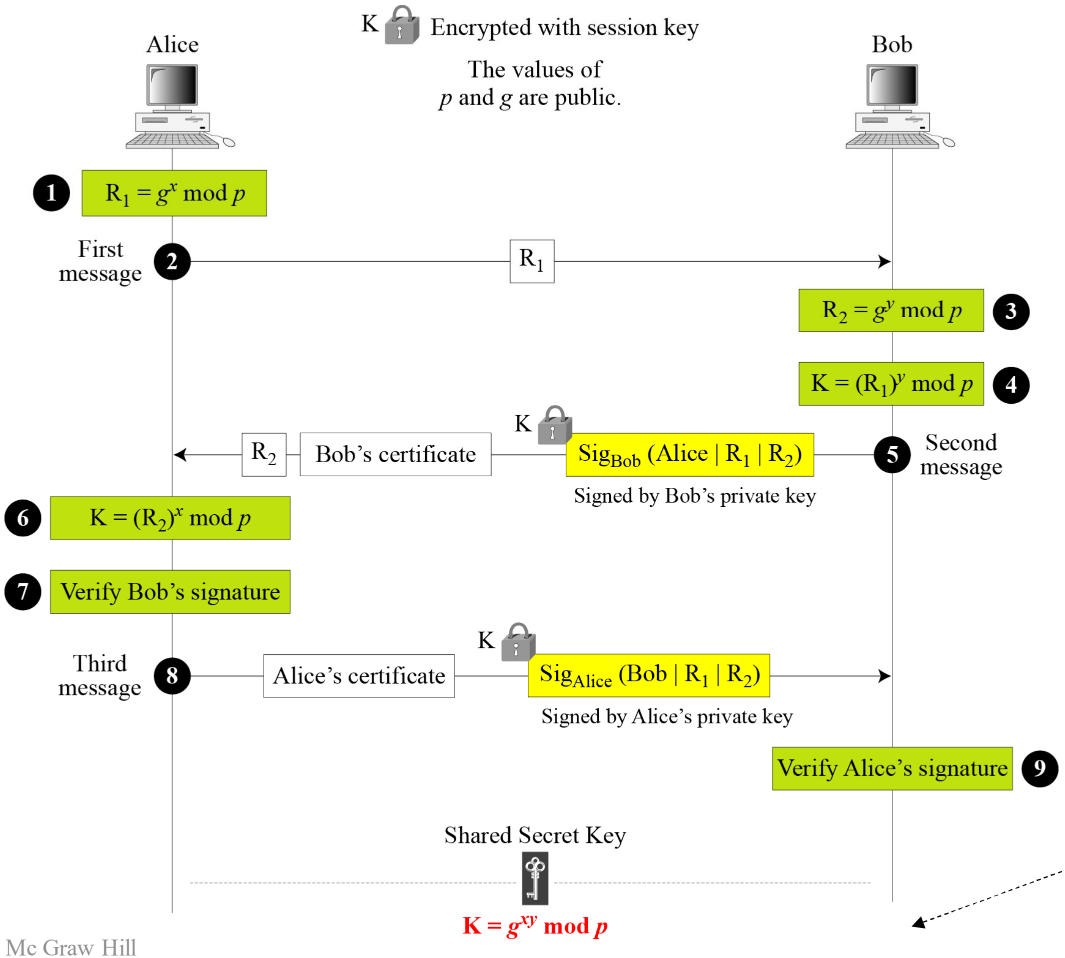
- All calculations are done modulo some large prime number (p)
  - $(g^x)^y$ is also true for modulo arithmetic!
  - Finding a large prime is computationally expensive
- The number (g) (generator) should be a primitive root of (p)
  - Meaning that [*g mod p*, *g^2 mod p*, …, *g^{p-1} mod p*] are distinct and generate all numbers from 1 to p-1 in some permutation
  - OK for g to be a small number (one digit), the number 2 also works!
- (p–1)/2 should also be a prime
- (g) and (p) are publicly shared
- The private numbers (x) and (y) must be less than (p)
- The numbers being used are quite long: p>300 digits, x and y>100 digits

Not part of this course.

Pre-defined Diffie-Hellman Groups

- Calculation of p and g is computationally expensive
  - Calculating them for each occasion is called “Ephemeral (short lived) Diffie-Hellman”
- There exist pre-defined Diffie-Hellman groups:
  - RFC 2539 (D-H keys in DNS), RFC 2409+3526 (IKE/IPsec), RFC 5114, etc.
  - ISO, ANSI and NIST standardize similar/same groups
- Example group 2: uses a 1024-bit prime with generator g=2
  p=79769313486231590770839156793787453197860296048756011706444236
  84197180216158519368947833795864925541502180565485980503646440548199239100050792877003355816639229553136239076508735759914822574862575007425302077447712589550957937778424442426617334727629299387668709205606050270810842907692932019128194467627007
- At least one 1024-bit group is believed to be cracked by NSA
  - 1 year of work, $100,000,000 → 66% of all VPN connections and 25% of SSH connections
  - A second group → 20% of the largest HTTPS sites open
- Shortest recommended length today is 2048 bits

Using D-H with public key authentication



It is still possible to take over the session.

All we have achieved so far is to have a common secret with someone we know.

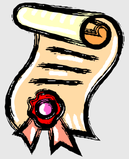
Forouzan: Cryptography and network security, Mc Graw Hill

X.509 certificates

Homework – used in LAB 3

Chapter 15.4-5

Digital Certificates

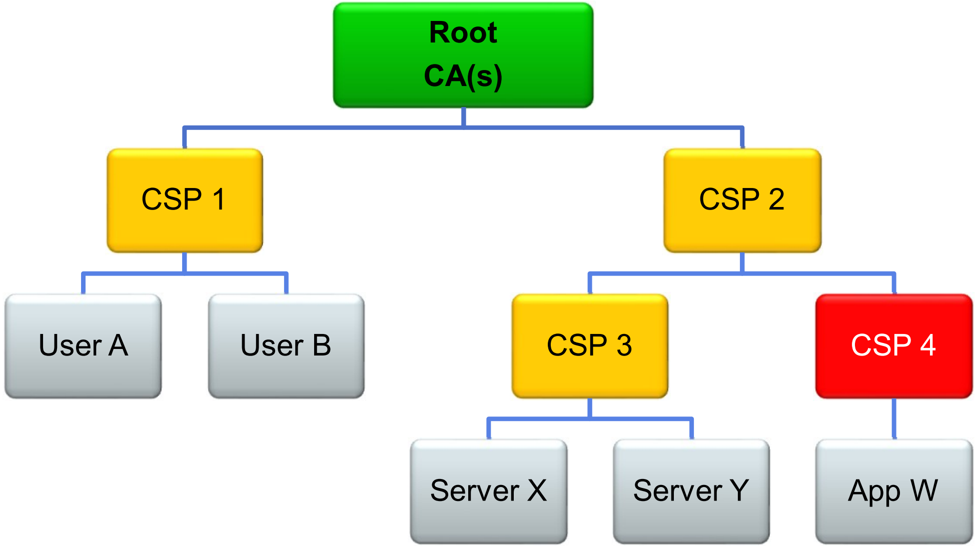


Homework

- **Associates a public key with a name**
- The certificate contains the owner’s **ID**, **public key** and a signature from the CA (the certificate authority – the issuer)
- The certificate does not contain the holder’s private key
- User authentication possible given a subject’s certificate
  - Signed → it does not matter how the certificate is distributed
  - Receiver must also check with Certificate Revocation Lists (CRLs) published by the CA
  - The OCSP protocol (on-line certificate status protocol) does it in real time
- Certificates are used to authenticate web sites
  - Some CAs, such as VeriSign, are widely trusted
  - But web browsers trust too many CA:s (>100) – this is a problem
  - Some have been hacked, other are not careful enough when issuing certificates
- Certificate use is not limited to web sites

Chain of trust

Homework



Someone in the chain (normally the root) must be trusted

Each certificate in the CA chain must be verified.

Note that if CSP4 (certificate service provider 4) is not reliable, it can even sign faked certificates for, for example, user A

Whole chains of issuer certificates can be cached → New certificates from the same issuer can then instantly be verified

Certificates are signed by a CA

Homework

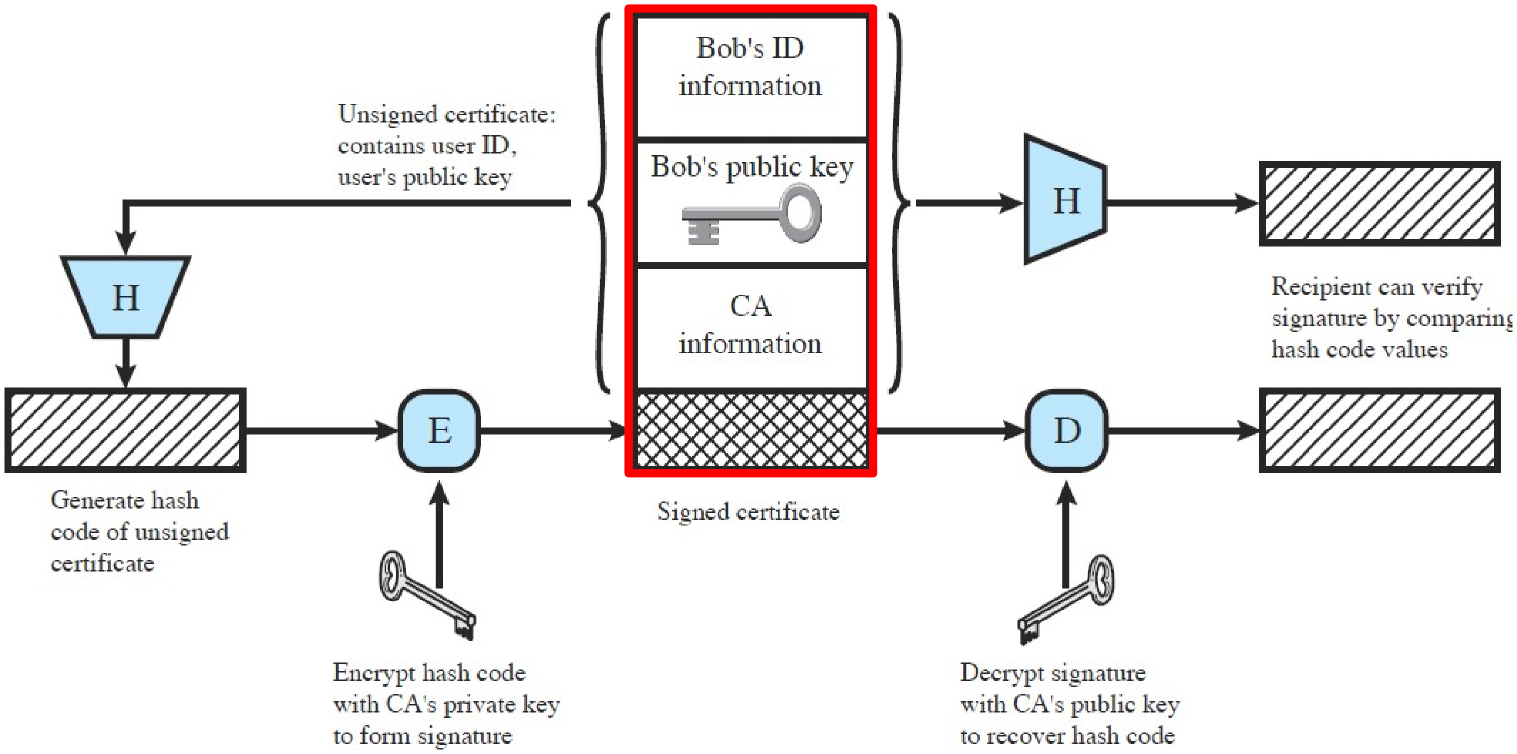
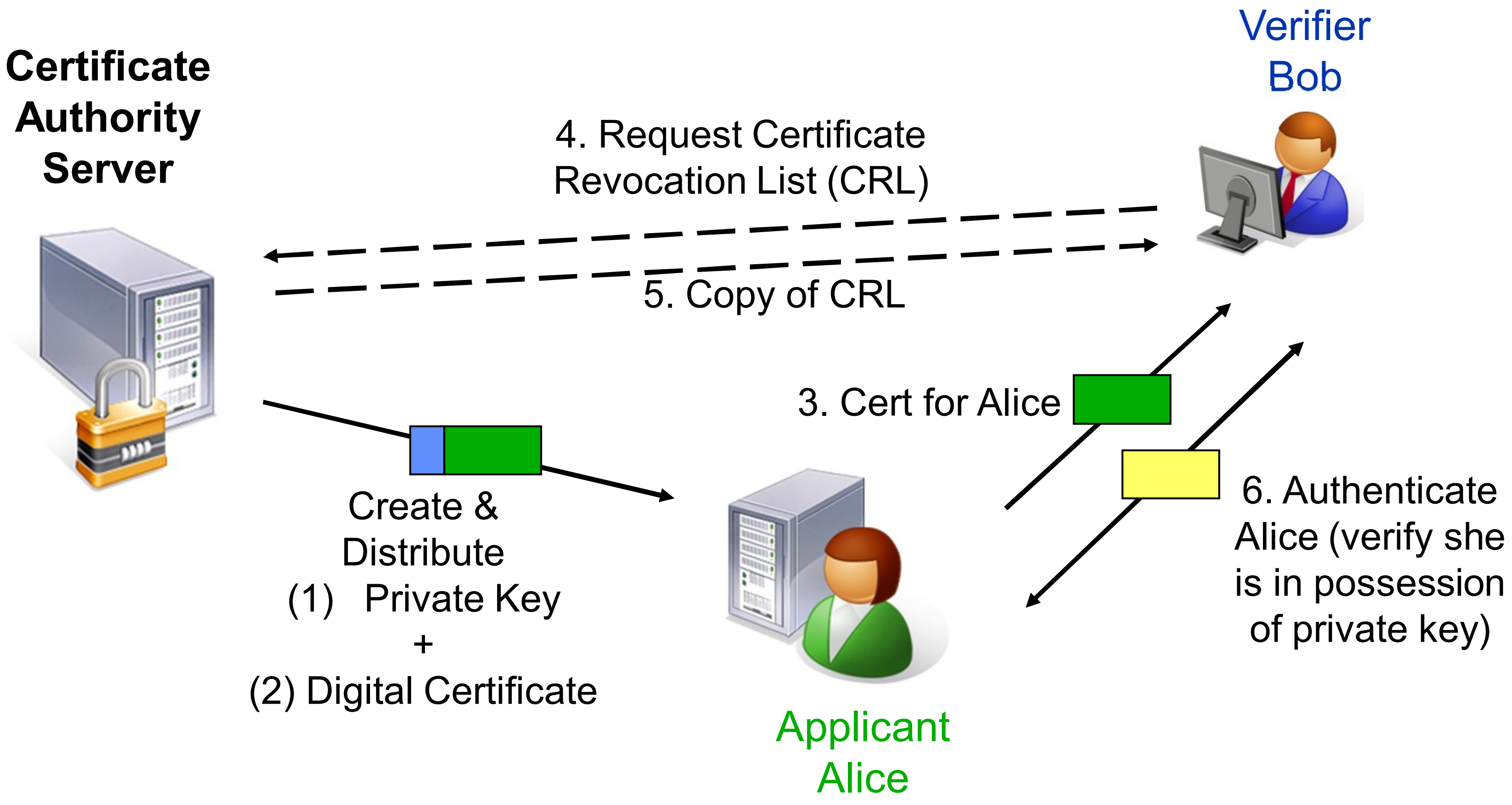


Fig. 15.10

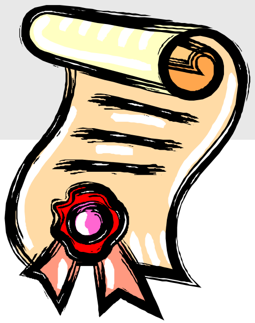
Public Key Infrastructure (PKI) with a CA

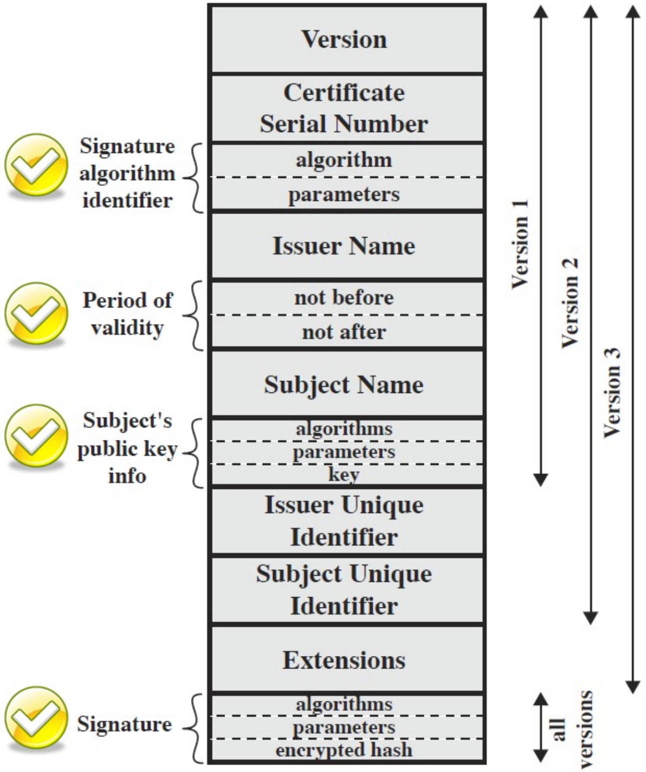
Homework


Certificate Authority Server
Verifier Bob
4. Request Certificate Revocation List (CRL)
5. Copy of CRL
3. Cert for Alice
Create & Distribute
(1) Private Key
+
(2) Digital Certificate
6. Authenticate Alice (verify she is in possession of private key)
Applicant Alice

The X.509 v3 certificate standard

Homework





(a) X.509 Certificate

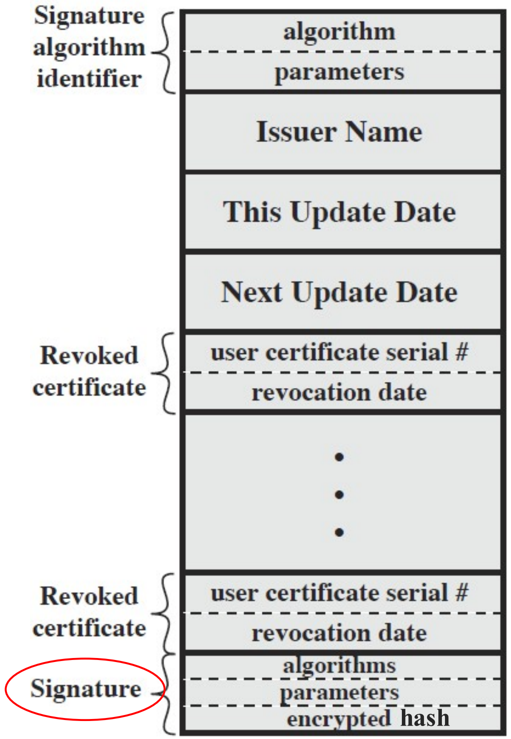
Fig. 15.11a

Extensions allow entities to have more than one key ("key usage" extension):

- signing
- key encryption
- data encryption, etc.

Certificate revocation lists (CRLs)

Homework



(b) Certificate Revocation List

Fig. 15.11b

Summary

Use:

- Asymmetric ciphers for authentication (e.g. certificates)
- Diffie-Hellman for session-key agreement
- Symmetric ciphers and stream ciphers for bulk data encryption
- Keyed hashes to detect data modification (e.g. HMAC)