Security Engineering

Cryptography and Information Security
Master in Informatics

José Bacelar Almeida

Cryptography

Preamble

Basic Concepts

Applied Cryptography

Information Security

CIA triad

- **Confidentiality:** Preserving authorised restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information. A loss of confidentiality is the unauthorised disclosure of information.
- **Integrity:** Guarding against improper information modification or destruction, including ensuring information nonrepudiation and authenticity. A loss of integrity is the unauthorised modification or destruction of information.
- **Availability:** Ensuring timely and reliable access to and use of information. A loss of availability is the disruption of access to or use of information or an information system.

Cryptography & Information Security

- "*Cryptography is an important computer security tool that deals with techniques to store and transmit information in ways that prevent unauthorized access or interference*" [ISO]
- Cryptography serve as a crucial line of defence against various threats, including unauthorised access, data breaches, tampering, and eavesdropping.
- Cryptography is a **security mechanism**, used in combination with others to achieve the goals of information security.

Cryptography

Preamble

Basic Concepts

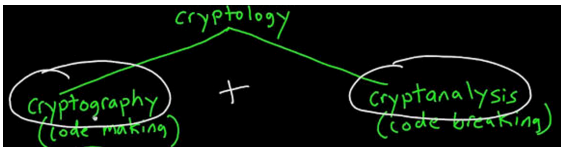
Applied Cryptography

Origins of the concept

- "Kryptós" + "Gráphein" (hidden writing)
- ...art of transmitting "secrets" (in an open channel)
- Used since ancient times, often associated with military and diplomatic activities.
- Today, the goals have expanded to encompass additional aspects of the communication process in hostile environments:
  - **Confidentiality**: secrecy of messages;
  - **Authenticity**: establish the origin of messages;
  - **Integrity**: transmitted message has not been tampered;
  - …
- Obs.: notice that the meanings of properties have been narrowed w.r.t. their use in the context of Information Security.

Cryptoanalysis

- In contrast, **cryptanalysis** aims to undermine the goals of cryptography.
- A cryptographic technique is deemed **broken** if its objectives have been successfully compromised through cryptanalysis.
- Cryptography and Cryptanalysis form an area sometimes called **Cryptology**.
- As a scientific discipline, cryptology touches on several different fields, such as probability, complexity theory, information theory, computer science, etc.



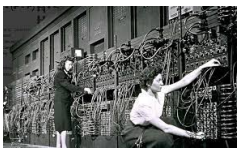
brief history

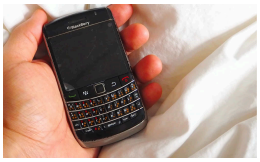
- The use of (proto) ciphers has been known since the 15th century BC.
- Armies of classical empires (e.g. Romans) regularly used cryptography

- The sophistication of the techniques has evolved over the centuries
- E.g. Enigma machine - WW2, Germany.
- Shortly after World War II, Claude Shannon (video) published two papers on Information Theory, which allowed for the formalisation of cipher security — considered the birth of Cryptography as a scientific field.

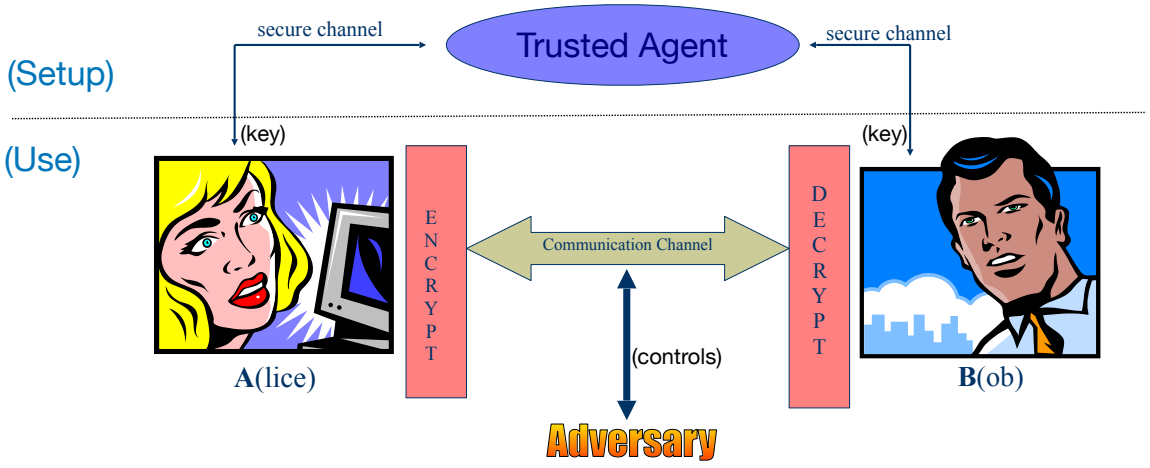
Modern Cryptography

- 1948/9 – Teoria da Informação (Claude Shannon).
- 1970/7 – Data Encryption Standard (DES).
- 1976 – Criptografia de Chave-Pública (Diffie & Hellmann)
- 2001 – Escolha do substituto do DES: Advanced Encryption Standard (AES).
- 2022/… – Criptografia Post-Quantum
- …



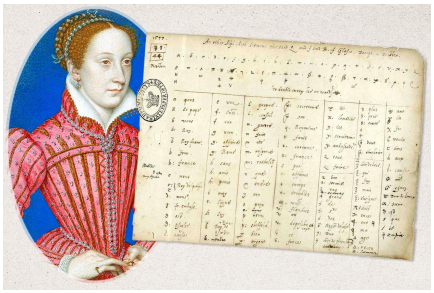

Model
(confidentiality)



Some more terminology

- **plaintext** (or **cleartext**): content of the message;
- **encryption**: operation that encodes the plaintext into an obscured message (**cyphertext**, or **cryptogram**);
- **decryption**: inverse operation of encryption
- **cryptographic system**: specification of the (possibly probabilistic) algorithms that perform some cryptographic technique (e.g. key-generation; encryption; decryption).
- **attack**: compromising the goals of a cryptographic technique (e.g. obtain the transmitted message without the corresponding key).
- **adversary** (intruder; enemy; …): hostile environment.

Security of Encryption



- Historically, security was largely attributed to the secrecy associated with the underlying technique
- ...which had disastrous consequences!
- The trend described was still present in the 20th century. However, as early as the 19th century, *Auguste Kerckhoff* established the following principle.:

  the security of a cipher must be assessed on the assumption that all the details of its construction is public knowledge

- Corollary: security can only be derived from a parameter that is explicitly secret - the **key**

Adversary

- Personalisation of the **hostile environment**
- …also known by the name of **enemy**; **spy**; **(E)ve**; **(M)allory**; …

Adversary success ⇒ ***Attack*** *on the cryptographic technique*

- **Cryptographic security** can defined as the absence of attacks:

A cryptographic technique is said to be **secure** if no adversary succeeds in attacking it.

- In practice, it's important to describe the adversary's capabilities:
  - The control he exercises over the channel (read only, read + write):
    - **Passive** - the adversary can only eavesdrop on the communication channel.;
    - **Active** - in addition, the adversary has the ability to manipulate the information circulating on the communication channel (modify / block / insert or repeat messages).
  - Computational power:
    - **computationally unbounded** - adversary is able to execute any algorithm instantaneously (with no memory limitation);
    - **computationally bounded** - adversary can only execute algorithms in some complexity class (Probabilistic Polynomial Time - PPT).
  - Additional information (e.g. previous communications);
  - etc.
- Depending on the type of adversary being considered, different notions of (cryptographic) security can be achieved
  - **Unconditional** (or **information-theoretically**) s**ecurity**— secure against a computationally unbounded adversary;
  - **Computational security** — secure against a computationally bounded adversary (PPT).

Classic Ciphers

Caesar Cipher

- Known to have been used by Julius Caesar during the Gallic campaign
- The encryption is a shift of the letters of the alphabet. .

Texto limpo:	A	B	C	D	E	F	···	T	U	V	X	W	Y	Z
Criptograma ($K = 6$):	G	H	I	J	K	L	···	Z	A	B	C	D	E	F

- To decipher, the shift is made in the opposite direction.
- The total number of possible keys is 26 (one of which is weak).
- Example: Encrypting the message `CartagoEstaNoPapo` with key K=6 results in `IGXZGMUKYZGTUVGVU`.

…attacking Caesar

- E.g. want to attack the ciphertext: FXLNTQCLOPNPDLC
- Small key-space suggests the following **strategy #1**:
  - decrypt with every possible key…
  - …and spot the one(s) that "make sense"

CRIPTOGRAMA:		FXLNTQCLOPNPDLC
+1	:	GYMOURDMPQOQEMD
+2	:	HZNPVSENQRPRFNE
....	:	.....
[MSG]+15	:	UMACIFRADECESAR (K=26-15=11)
....	:	.....
+24	:	DVJLROAJMNLNBJA
+25	:	EWKMSPBKNOMOCKB

- **Strategy #2**: frequency analysis allows for more efficient attacks…

  high frequency of 'L' suggests that it encrypts 'A' (that is, K = 'L'-'A' = 11)

Brute-force Attack

- Strategy #1 attack is known as **Brute-force Attack**

  the adversary searches the entire key space in the hope of finding the right key

- It assumes that:
  - there is enough redundancy in the original message,
  - or a plaintext/ciphertext pair is known.
- Often seen as an attack that can always be applied to a cryptosystem...
- …but its feasibility depends on the size of the key space.
- **Conclusion**: key sizes are a necessary (although not sufficient) criterion for the security of ciphers.

big numbers!

…what is as reasonable key size?

- If we consider keys to be arbitrary bitstrings, the size of the key space increases exponentially on the key size
- Example:

Key Size	**Time** ($1\mu sec/test$)	**Time** ($1\mu sec/10^6 tests$)
32 bit	35.8 min.	2.15 msec.
40 bit	6.4 days	550 msec
56 bit	1140 years	10 hours
64 bit	500000 years	107 days
128 bit	$5 * 10^{24}$ years	$5 * 10^{18}$ years

- Baseline: 2^{112} provides a reasonable security level!

Monoalphabetic Substitution Cipher

- Instead of the shift used in the Ceaser Cipher, encryption performs an arbitrary permutation of the alphabet (decryption uses the inverse permutation);
- E.g.:

A	B	C	.....	W	Y	Z
R	X	K	.....	B	I	F

- Much larger key-space ($26! \approx 17.5 * 10^{24}$)

…should we trust the security of this cipher?

…an attack

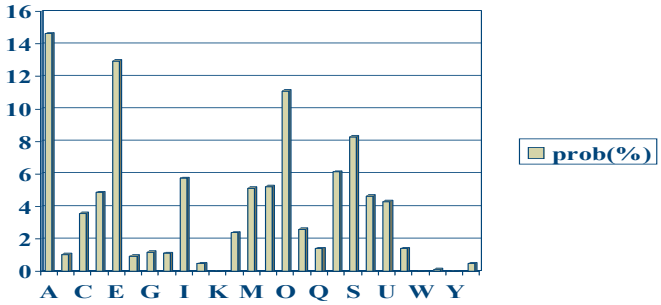
- Consider the following ciphertext:

```
FPGFBNBVPKFBDMSBEMDMGUCDKDGUGDMUSPMMDBEFLEFEQDCPPGIDEXDCBKPMDHKPMPFQBUGPSUGHKEGPF
QBMPXPKSESEBSURBHKBHBMEQBFUFSDSBGHKPPFCECPHDKQPDHDKQPFDBADVEDFDCDDCEZPKLDZEDGMPPM
NDKPMDGDVPEMPDNUPFQDVDMPCPZGEFUQBMCPUGMEOPFSEBHPFBMBFBDUNPCDPXSEQDSDBCBBUQBFBCPMU
KNEKDGUGDMHPKMBFDNPFMCPBKENPGBIMSUKDSBGJUPGPFQKPQEVPSBFSEOEDIUOBMPGKPMQDUKDFQPMPX
SPFQKESBMPMMPMQKDZEDGUGDHPKNUFQDQPKKEVPOBJUPPJUPVDEMUSPCPKKPMHBFCEOAPMJUPFDBMDIED
PPOPMBOADKDGHDKDBHKDQBSBGEFJUEPQDSDB
```

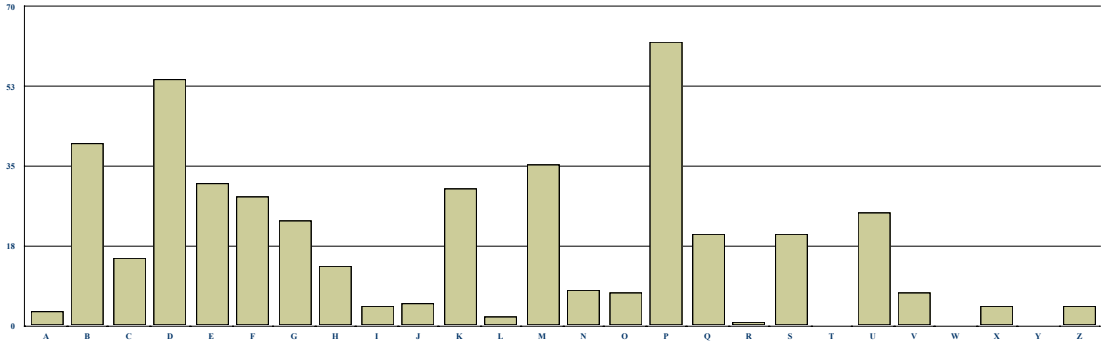
- …known to be a Portuguese text.

how should we proceed?

- Letter frequency of Portuguese:



- Histogram of letters in the ciphertext:



- Suggests matching “A”; “E”; “O” with “P”; “B”; “D”
- Pairs or triples of letters can also be considered, such as “os”; “es”; “que”; “nao”; …
- Several occurrences of the pairs “PM”, “PF”, “MP”, and “JUP” suggest the following partial decryption: {…; F:M; …}

```
ME-MO-O-E-MOAS-O-SAS-U-A-A-U-ASU-ESSAO-M--M--A-EE--A--A-O-ESA--ESEM-OU-E-U--
---EM-OSE-E-----O-U-O--O-OS--OMUM-A-O---EEM---E-A--EA-A--EMAO-A--AMA-AA---E-
-A--A-SEES-A-ESA-A-E-SEA-UEM-A-ASE-E---MU-OS-EU-S--EM--O-EMOSOMOAU-E-AE----
A-AO-OOU-OMO-ESU----A-U-AS-E-SOMA-EMS-EO---E-O-S-U-A-O-QUE-EM--E---E-OM----
A-U-OSE--ES-AU-AM-ESE--EM----OSESSES--A--A-U-A-E--UM-A-E----E-OQUEEQUE-A-SU-
E-E--ES-OM----ESQUEMAOSA--AEE-ESO--A-A--A-AO--A-O-O--MQU-E-A-AO
```

- …which, on closer inspection, does not seem to make much sense.… :-(

- We need to backtrack on some of our hypotheses…
- Reassigning the decryption of F into N…
- …we get something a lot more promising:

```
NE-NO-O-ERNOAS-O-SAS-U-ARA-U-ASU-ESSAO-N--N-TA-EE--A--A-ORESA-RESENTOU-E-U--R--
ENTOSE-ER----O-U-O-RO-OS-TONUN-A-O--REEN---E-ARTEA-ARTENAO-A--ANA-AA---ER-A--A-
SEES-ARESA-A-E-SEA-UENTA-ASE-E---NUTOS-EU-S--EN--O-ENOSONOAU-E-AE---TA-AO-
OOUTONO-ESUR--RA-U-AS-ERSONA-ENS-EOR--E-O-S-URA-O-QUE-ENTRET--E-ON----A-U-OSE-
RESTAURANTESE--ENTR--OSESSESTRA--A-U-A-ER-UNTATERR--E-OQUEEQUE-A-SU-E-ERRES-ON---
-ESQUENAOSA--AEE-ESO--ARA--ARAO-RATO-O--NQU-ETA-AO
```

- which rapidly leads to the whole plaintext

```
NEMNOGOVERNOASCOISASMUDARAMUMASUCESSAOINFINITADEEMBAIXADORESAPRESENTOUMECUMPRIMEN
TOSEXERCICIOCUJOPROPOSITONUNCACOMPREENDIDEPARTEAPARTENAOHAVIANADAADIZERFAZIAMSEES
GARESAMAVEISEAGUENTAVASEDEZMINUTOSDEUMSILENCIOPENOSONOAUGEDAEXCITACAODOOUTONODESU
RGIRAMUMASPERSONAGENSDEORIGEMOBSCURACOMQUEMENTRETIVECONCILIABULOSEMRESTAURANTESEX
CENTRICOSESSESTRAZIAMUMAPERGUNTATERRIVELOQUEEQUEVAISUCEDERRESPONDILHESQUENAOSABIA
EELESOLHARAMPARAOPRATOCOMINQUIETACAO
```

Vigenère Cipher

(polyalphabetic substitution)

- Attributed to *Blaise Vigenère* (16th century). Known as "*le chiffre indéchiffrable*".
- It interleaves multiple Caesar ciphers.
- Broken in the 19th century by *Charles Babbage* and *Friedrich Kasiski*.
- Description:
  - key is a password — each letter corresponds to a single Caesar cipher key (A=0; B=1; …);
  - Encryption: apply the Caesar cipher with each character of the key in sequence, starting over when no more key characters are available.
- Observations:
  - same letter is not always encrypted in the same way (add some hurdles to the frequency analysis)
  - … but if the plaintext is much larger than the key, patterns in the plaintext shall be reflected in the ciphertext.
  - The cryptanalysis techniques that have been developed exhibit already some degree of sophistication.

Transposition Cipher

- Encryption permutes character positions (instead of changing them…)
- E.g.
  - consider a permutation [2; 1; 3] (key)
  - to encrypt “AindaOutraCifra”, write it on a matrix…

2	1	3
A	I	N
D	A	O
U	T	R
A	C	I
F	R	A

  - …and read the columns (with header [1; 2; 3])
  - resulting ciphertext: “IATCRADUAFNORIA”
- Observation: frequency analysis is now useless!

Cipher Composition

- Having seen different simple ciphers…
- Is it sensible to construct more intricate ciphers, which are more secure, by combining multiple simpler ciphers?

- It depends!
  - it may happen that it doesn't add any value (e.g. combining two ciphers by substitution)
  - but there are combination patterns that can be advantageous (e.g. by interleaving substitutions with permutations — aka *SP-networks*).

One-Time Pad

(Vernam Cipher - 1917)

- Generalises Vigenère cipher with:
  1. key size is, at least, the plaintext size;
  2. key is fully random;
  3. key is only used in a single encryption.
- Normally described as operating on a binary alphabet: encryption/decryption is the *exclusive-or* (xor) with the corresponding key bit.

$$C_i = T_i \oplus K_i \qquad M_i = C_i \oplus K_i$$

- Shown to be secure (**information-theoretically secure**) by *Claude Shannon — "knowing the ciphertext does not reduce the uncertainty about the plaintext"*.
- Key generation and distribution make this cipher unfeasible in realistic scenarios.

Conclusion

- In cryptanalysis, all available information is used, including:
  - partial information about the transmitted message;
  - previous cipher's use (e.g. messages encrypted with the same key);
  - and possible weaknesses in the cipher's use (e.g. deficiencies in the choice of keys, etc.).
- Although unconditionally secure techniques exist, they often have such strict requirements that they become impractical.
- Most (all?) cryptographic techniques used today are based on **computational security**, which limits the adversary's computational capabilities.
- The size of a cipher's key should be determined based on the desired level of security (e.g. 2^{112}), while taking into account the amount of key size that is “consumed” by known cryptanalysis techniques.

Cryptography & Security

(Cryptographic) Security Properties

- We have mainly focused on ciphers — a cryptographic primitive whose goal is to ensure *confidentiality*.
- But cryptography is used today to provide guarantees for a wide range of security properties:
  - **confidentiality** — content of the message is only known to the legitimate parties;
  - **integrity** — the recipient would "reject" messages that have been tampered;
  - **autenticity** — ensures the "origin" of the message to the recipient;
  - **non-repudiation** — the “origin” of the message cannot deny it;
  - **anonymity** — no information available regarding the “origin” of the message;
  - **identification** — establish the “identity” of a party;
  - …

Cryptographic Services and Protocols

- Usually, one is interested in a combination of these properties (e.g. a secure channel between two parties aims to guarantee confidentiality, authenticity and integrity).
- On the other hand, some of these properties do not directly follow from a single cryptographic technique, but rather from a combination of techniques.
- Leading to what is known as **cryptographic protocols** — specifications for message exchanges that rely on cryptography to achieve a desired end.
- The security of those protocols relies not only on the security of the underlying cryptographic techniques, but also on subtle interactions between them.

Cryptography & Security

*The security of a system using cryptography **is not just** the security of the underlying cryptographic techniques.*

- We can distinguish (at least) the following levels when establishing the security of a system that uses cryptography:
  - cryptographic scheme;
  - protocol;
  - implementation (coding);
  - usage.

A breach at any of these levels jeopardises the security of the entire system!

Applied Cryptography

Security Engineering
Master in Informatics — Cryptography and Information Security

José Bacelar Almeida

Applied Cryptography

Symmetric Crypto

Asymmetric Crypto

Applications

Roadmap

- Ciphers
  - Stream ciphers
  - Block ciphers
- One-Way Functions
  - Cryptographic Hash Functions
  - Message Authentication Codes (MAC)
  - Key-Derivation Functions (KDF)
- Key-Management

Stream Ciphers

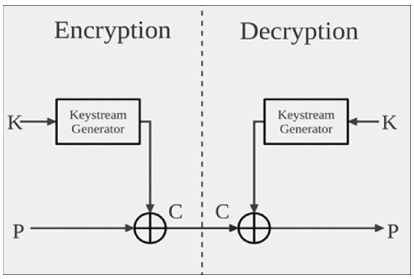
Recall OTP (one-time pad)

- Known to be (information-theoretically) secure

- Inherent issues:
  1. keys cannot be reused (hence the name!)
  2. depend on a truly random key with a size greater than the plaintext
  3. does not promote diffusion — information on the structure of the message can be used to manipulate it (e.g. bit swapping).
- Difficulties associated with key generation and key distribution render the OTP mostly unusable in real-world applications.

Stream-Ciphers

- **Basic idea**: approximate OTP using a key stream generator which produces an arbitrary-length keystream from a short, fixed-length key.

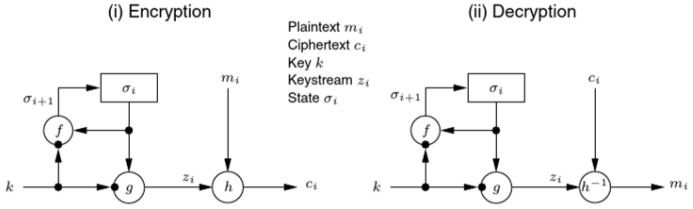


- The generation of the keystream must be reproducible (deterministic) — i.e. a finite state machine.
- Therefore, the sequence must necessarily be **cyclic**. The **period** is the length of the longest sequence before starting to repeat…

Criteria for the design of Stream Ciphers

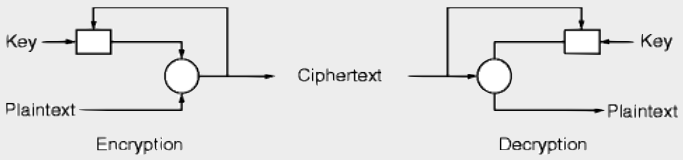
- Period should be as long as possible (always longer than the plaintext).
- Key sequence must be:
  - **pseudo-random**: statistic properties of the sequence are those of a truly-random sequence;
  - **unpredictable**: should be impossible to predict the “next bit”, after observing a given prefix of the sequence
- Other characteristics
  - synchronism:
    - **synchronous** — keystream is independent of the message;
    - **self-synchronising** — able to recover synchronism when bits of the ciphertext are lost .
  - error-propagation: the impact of transmission errors on decrypted messages
  - …

Synchronous Stream Ciphers



- Keystream is independent of the message;
- Loss or insertion of bits in the cryptogram results in loss of synchronisation.
- Transmission errors (bit swapping) only affect the corresponding position of the original message.
- The key might affect:
  - The next-state function *f* — **Output-Feedbak Mode**.
  - The output function *g* — **Counter Mode**.
  - Both...

Self-synchronising Stream Ciphers



- Next-bit is computed from the last n bits of the ciphertext (and key);
- An IV (initialization vector) is used to initiate the process;
- In case of a transmission error, synchronization is restored once the flipped bit is no longer used for the next bit computation.
- Possible problem: vulnerable to replay attacks.

Cryptographic-Secure Pseudo-Random Number Generation

- *Golomb*'s Randomness Postulates:
    1. The difference in the number of 1s and 0s must tend towards zero;
    2. The expected number of sub-sequences of repeated symbols (runs) with length *l* is given by $r(l)=r/2^l$;
    3. The auto-correlation must be a constant value for any deviation other than *0 (mod p)*.
- Cryptographic Security:

    The generated sequence must be indistinguishable from a random sequence for any Probabilistic Polynomial Time (PPT) adversary.

Key reuse and NONCEs

- The above description of sequential ciphers inherits a problem from OTP: **key reuse**
    - encrypting different messages with the same key leads to the same generated keystream.
- This problem is generally overcome through the use of **NONCE**s — an abbreviation of **Number used only ONCE**.
    - A number that should never be repeated;
    - but is not required to be kept secret (i.e. it can be sent along with the cryptogram).
- In practice, the Nonce is typically a sequence of randomly generated bytes used during encryption and sent along with the cryptogram.

Some Examples

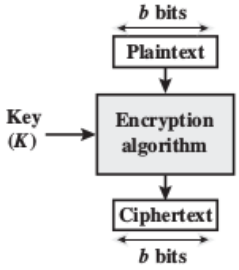
- Stream ciphers are used in several well-known applications:
  - **A5** (**A5/2**), used in the GSM standard;
  - **E0**, used in the Bluetooth protocol;
  - **CSS** (Content Scramble System), used to protect DVD discs.

  (Note: All of the above examples are known to offer weak security guarantees.)
- **RC4** (ArcFour) — a cipher designed for efficient software implementation, which was once widely used but **is now considered broken**.
- **ChaCha20** — a modern (and secure) stream cipher.

Block Ciphers

Block Ciphers

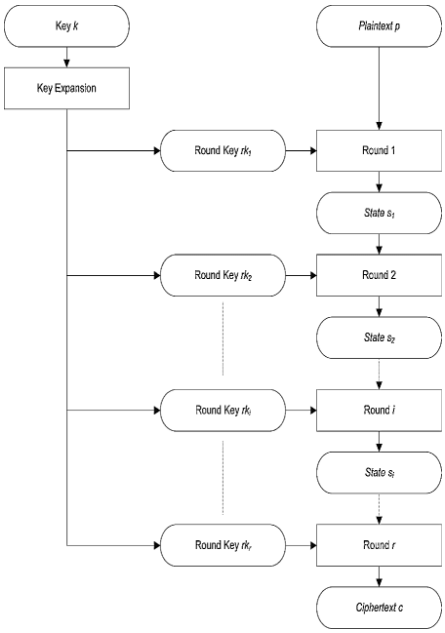
- A block cipher focuses on a simpler primitive that processes fixed-length messages (a **block**)



- The generalisation to a cipher proper (which deals with messages of arbitrary length) is handled by a standardised construction (**mode of operation**).
- Security-wise, the primitive is expected to be indistinguishable from a random permutation (when k is sampled randomly) — what is named a (keyed indexed) **pseudo-random permutation** (**PRP**).

Design of block ciphers

- Two distinguished desirable properties [*C. Shannon* 1949]:
  - **Diffusion** — the influence of changes in the plaintext should be spread throughout the ciphertext;
  - **Confusion** — the relationship between the ciphertext and the encryption key should be as complex as possible.
- Most modern designs are iterated ciphers, where a bijective **round** transformation is applied repeatedly with distinct *round keys* (derived from the main cipher key).



Block vs. Stream Ciphers

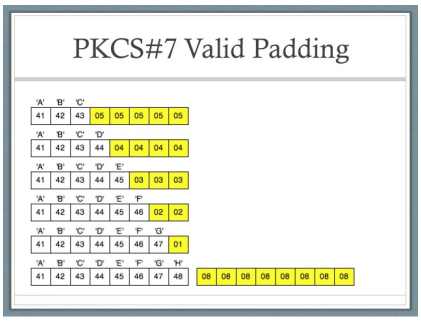
- Different “processing unit” (block vs. bit/byte);
- Use of block ciphers depend on some *mode of operation*;
- Stream ciphers do not promote diffusion;
- Block cipher inverse operation (decryption) is harder to obtain;
- Stream ciphers are typically faster than block ciphers;

Basic Modes of Operation

- A block cipher primitive, by itself, can only process fixed-length bitstrings (blocks).
- A block cipher **mode of operation** describes how the primitive is used to encrypt/decrypt arbitrary-length messages.
- Depending on the applications, we can choose among several standard modes:
  - *Electronic Code Book* (**ECB**)
  - *Cipher Block Chaining* (**CBC**)
  - *Counter Mode* (**CTR**)
  - …
- Each mode of operation comes equipped with its own security analysis: assumptions, security proofs, etc.

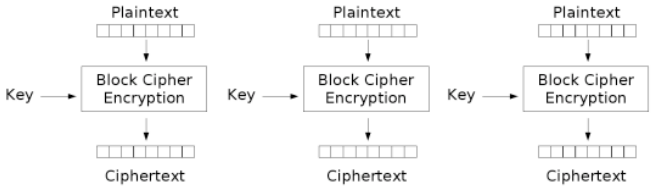
Padding

- Certain uses of block ciphers (modes of operation) require the size of the plaintext to be a multiple of the block size.
- This requirement is usually solved by adopting a **padding** method: a reversible transformation that fills the last block of the plaintext.
  - Several standard constructions available
    - bit vs. byte oriented;
    - deterministic vs. radomised;
    - etc.
  - E.g. PKCS7 padding

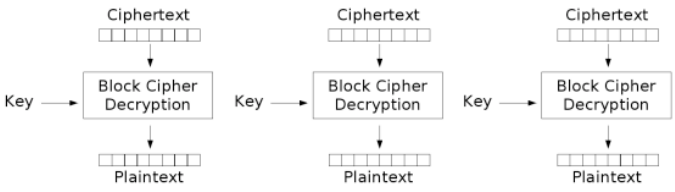


- [remark: *Ciphertext Stealing* is an alternative strategy for handling messages of arbitrary size].

Electronic Code Book (ECB)

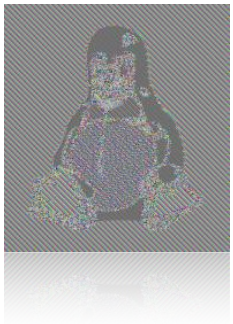


Electronic Codebook (ECB) mode encryption

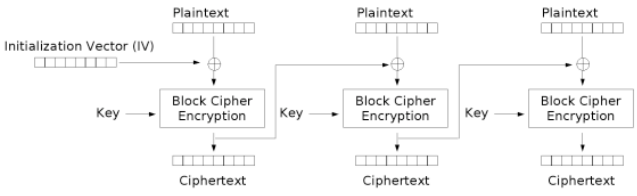


Electronic Codebook (ECB) mode decryption

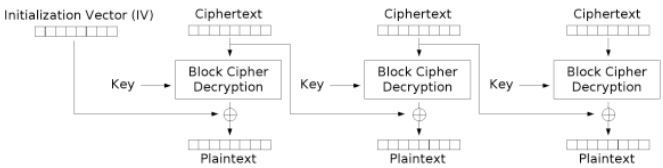
- Requires Padding.
- Repetition of blocks is detectable - **code book attack**;
- …leading to the leakage of patterns of the plaintext.
- **It should only be used to encrypt single-block messages.**
- Vulnerable to repetition/replacement attacks.


Cipher Block Chaining (CBC)



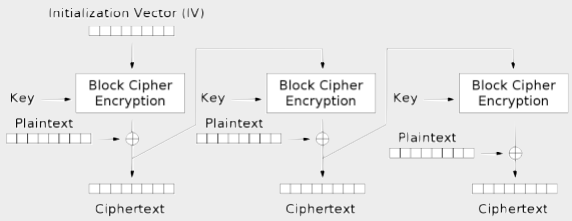
Cipher Block Chaining (CBC) mode encryption



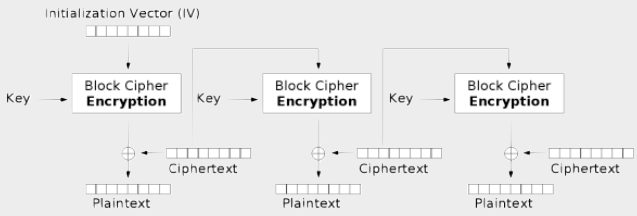
Cipher Block Chaining (CBC) mode decryption

- Each plaintext block is masked by the previous ciphertext
- …which induces a dependency on all previous plaintext blocks.
- A random **Initialisation Vector** (**IV**) shall be used to initiate the process.
- As with Nonces, the IV doesn't need to be kept secret.
- The swapping of one bit of the ciphertext would impact the corresponding block plus a single bit on the next-block.
- The last block of the ciphertext can be used as a Message Authentication Code (CBC-MAC).
    - In such use, a fixed IV shall adopted;
    - Secure for fixed-length messages (but simple fixes can be adopted to make it suitable for arbitrary length messages).

Cipher FeedBack mode (CFB)



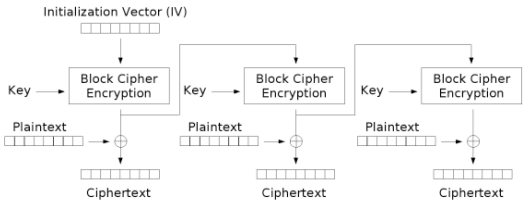
Cipher Feedback (CFB) mode encryption



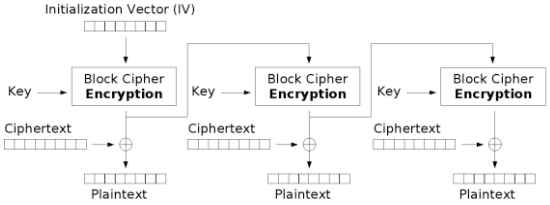
Cipher Feedback (CFB) mode decryption

- Implements a self-synchronising stream cipher.
- Does not requires padding.
- IV should be a Nonce.
- Notice that block cipher primitive is always used in “encrypt” mode.
- Keystream depends on the key, IV, and all the previous plaintext.
- Number of bits in the feedback is variable (CFBn).
  - Between 1 and the block size;
  - Feedback transfers the n most significant bits to the least significant bits (with a shift of the remaining bits);
  - Number of bits in the feedback impact the resynchronisation time.

Output FeedBack mode (OFB)



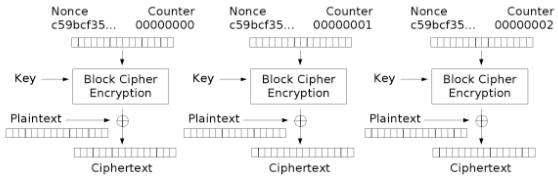
Output Feedback (OFB) mode encryption



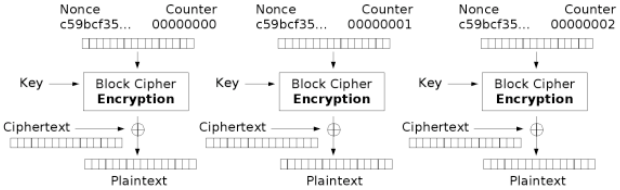
Output Feedback (OFB) mode decryption

- Implements a synchronous stream cipher (in output-feedback mode).
  - The block cipher is used as the *next-state function*;
  - The *output-function* is the identity function.
- Does not requires padding.
- IV should be a Nonce.
- Block cipher primitive is only used in “encrypt” mode.
- Keystream doesn’t depends on the message — is obtained by iterating the block cipher on the IV.
  - It can be expanded prior to the reception of the messsage;
  - but needs to be computed sequentially.
- The swapping of one bit of the ciphertext only impacts a single bit.

CounTeR mode (CTR)



Counter (CTR) mode encryption



Counter (CTR) mode decryption

- Implements a synchronous stream cipher (in counter mode).
  - The *next-state function* is a simple increment;
  - The block cipher is used as the *output-function*.
- Does not requires padding.
- IV should be a Nonce.
- Block cipher primitive is only used in “encrypt” mode.
- Keystream doesn’t depends on the message.
  - It can be expanded prior to the reception of the messsage;
  - and each block of the keystream can be processed independently (e.g. in parallel).
- The swapping of one bit of the ciphertext only impacts a single bit.

Authenticated Encryption

- A recent trend has been the adoption of block cipher modes of operation which provides both **confidentiality** and authenticity (**integrity**) of data.
- Usually supports also "*Associated Data*” — data which is not encrypted but is attached to the integrity guarantee (e.g. metadata).
- Some examples:
  - **EAX** (Encrypt-then-Mac-then-Translate)
  - **CCM** (Counter with CBC-MAC)
  - **GCM** (Galois/Counter Mode)
  - **OCB** (Offset CodeBook)

Advanced Encryption Standard (AES)

- NIST call for block cipher algorithm that would replace Data Encryption Standard (DES) — (announcement:1997, submissions: 1998, decision 2001).
- Requirements:
  - Blocks and keys with, at least, 128 bit;
  - Faster and more secure than TripleDES;
  - Detailed specification and design rational;
  - Suitable for software implementations (32bit architectures).
- Finalists:
  - **MARS** — complex, fast, high security margin
  - **RC6** — very simple, very fast, small security margin
  - **Rijndael** — clean design, fast, good security margin (**WINNER**)
  - **Serpent** — slow, clean, very high security margin
  - **Twofish** — complex, very fast, high security margin

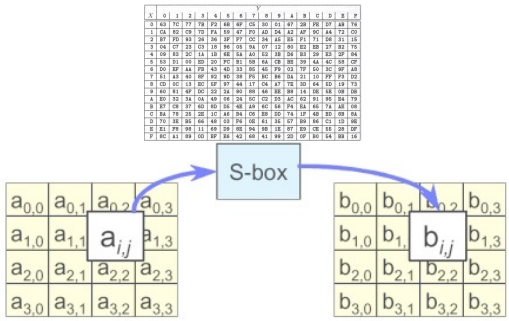
AES (RijnDael) description

- Block size of 128 bit (16 byte), organised as a 4x4 byte matrix;
- Iterated block cipher. Number of rounds: 10, 12 or 14 (depending on the key size);
- Supports keys of 128, 192 and 256 bit. KeyExpansion algorithm expands it into required 128 bit round keys;
- Detailed mathematical justification for specification details;
- Round processing steps:
  - SubBytes
  - ShiftRows
  - MixColumns
  - AddKey

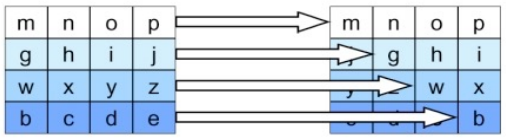
  (with corresponding *inverted* variants and subtle teaks to facilitate implementation)
- **Believed** to be resistant to all *known* attacks!

AES round

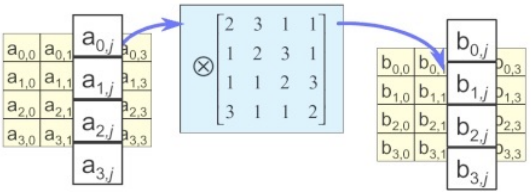
- ByteSub
  - A single S-box
  - Highly non-linear



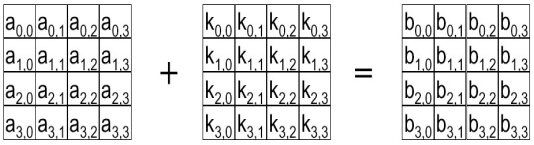
- ShiftRow
  - Rotate rows



- MixColumn
  - Interaction with ShiftRow promotes high diffusion in multiple rounds



- KeyAddition



One-Way Functions

One-Way Functions

- Security of cryptographic techniques often translates into establishing that a given function is **one-way.**

  A function $f:A \rightarrow B$ is **one-way** when, given some element $x:A$, it is *easy* to compute $f(x):B$, but giving any $y:B$ in the range of f, is very *difficult* to find some $x:A$ such that $y=f(x)$ (a *pseudo-inverse* of y).

- Notions of "easy" and "difficult" must be understood in the context of complexity theory, depending on the existence (or not) of *polynomial-time algorithms* for their calculation.

  Existence of one-way functions ⇒ P≠NP

  (obs.: but is indeed *a stronger assumption*)

- From a theoretical point of view, the existence of OW functions is usually identified as the complexity assumption that makes symmetric-cryptography feasible.

Cryptographic Hash functions

- A paradigmatic example of one-way functions are cryptographic hash functions.
- A **cryptographic hash function** is a one-way function that maps arbitrary length bit-strings into a fixed-length *n*-bit range.

$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

- Looking at the cardinalities of domain/codomain, we necessarily have that hash functions are non-injective — **collisions** always exist! (that is, distinct domain points m_1 and m_2 such that $H(m_1)=H(m_2)$).
  - …but, being OW functions, finding collisions should be difficult.
- Their use is prevalent in cryptography.

Defining properties

- The requirements of hash functions are usually expressed by the following hierarchy of properties:
  1. (**First) pre-image resistant**: given a hash value *h*, it should be unfeasible to obtain a message *m* such that $H(m)=h$.
  2. **Second pre-image resistant**: given a message m_1, it should be infeasible to obtain a message m_2 distinct from m_1 such that $H(m_2)=H(m_1)$.
  3. **Collision resistant**: it is unfeasible to find distinct messages m_1 and m_2 such that $H(m1)=H(m_2)$.
- The first simply asks for one-wayness of the hash function H.
- But when designing/choosing hash functions, we look at the stronger property of *collision resistance* (collisions cannot be found).
- However, the security of some applications only asks for one of the weaker properties…

Birthday paradox

- A standard result in probability theory tell us that we need reasonably sized codomain to achieve collision resistance.

  How many people need to meet to make it more likely than unlikely *that at least 2 share the same birthday?*

- …it would be enough to check around *sqrt(366)* random elements!
- If we consider typical hash codomain sizes ranging on 128 and 512 bits
- …we conclude that the *birthday attack* would require around 2^{64} and 2^{256} random picks to find a collision.

Examples

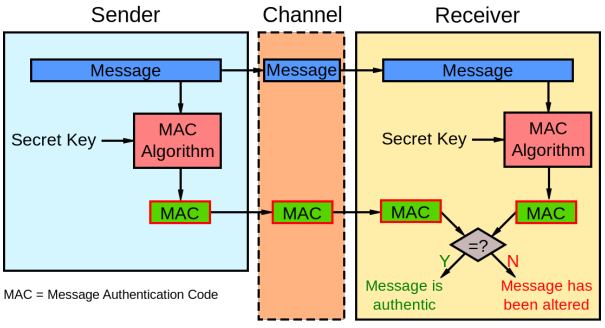
- Most common cryptographic hash functions:
  - **MD5** — codomain size: 128 bit. **Already broken!**
  - **SHA-1** (Standard Hash Algorithm) — codomain size: 160 bit. **Already broken!**
  - **SHA-2** — codomain sizes: ~~224~~, 256, 384 and 512 bits. Adopt the same design of SHA-1, with enhanced security.
  - **SHA-3** — codomain sizes: ~~224~~, 256, 384 and 512 bits. Follow a different design (sponge construction).
  - **SHAKE-128/256** — eXtendable Output Function (XOF). The same design of SHA-3.

Applications of Hash functions

- Password storage/verification;
- Building block of other cryptographic components:
  - Message Authentication Codes (MAC);
  - Key-Derivation Functions (KDF);
  - Secure Pseudo-Random Number Generators;
  - Commitment-schemes;
  - Block ciphers;
  - …

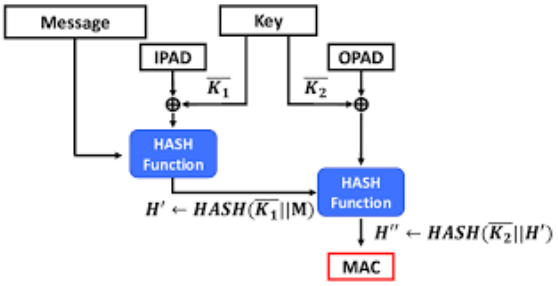
Message Authentication Codes (MAC)

- Hash functions, per se, do not ensure integrity of data — a malicious user might recompute its value for the modified data.
- A **Message Authentication Code** (**MAC**) should be used — it relies on a secret key that is known only to the legitimate parties (sender and receiver).
  - sender computes a **authentication tag** (from message and key);
  - receiver recomputes the tag, and compares it with the received one.
- Informally, it can be though as a Keyed-Hash function (Hash function with a secret key).
- Examples: `Poly1305`; `CMAC`.



HMAC

- A MAC can be easily constructed from any Hash function.
- A standard construction is the Hash-based Message Authentication Code (HMAC).
  - It performs two-passes on the hash function, to account for possible weakness of the used functions (length-extension attacks).
- Examples: HMAC-SHA256; etc.
- Remark: the SHA-3 design is not vulnerable to length-extension attacks. Hence, the HMAC nested construction is redundant for the SHA-3 family of hash functions — its enough to use $H(K||M)$.



Key-Derivation Functions (KDF)

- Key-Derivation Functions (KDF) allow the derivation of cryptographic keys from other (secret) material.
- KDFs are used in a variety of situations:
  - derive keys from weak secrets (passwords/passphrases);
  - derive keys for different cryptographic purposes from a single "*master secret*" (key diversification);
  - derive keys of different length from the ones provided;
- Sometimes, it accepts also non-secret inputs (`other-info`), allowing to bind the secret to application-specific data.

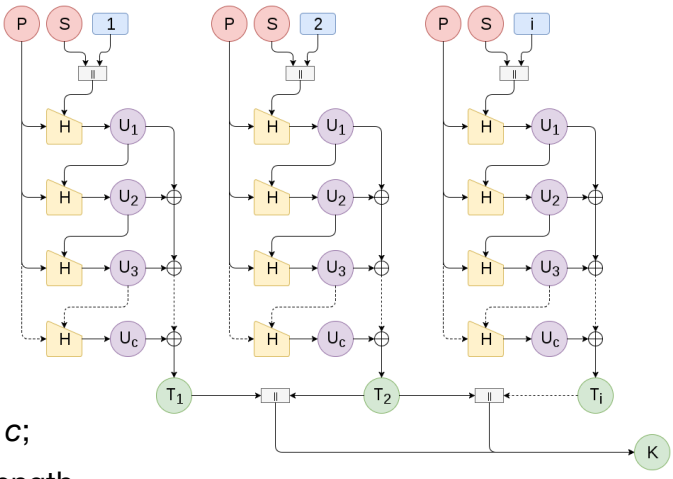
Password-Based KDF

- Passwords/Passphrases cannot be used directly as cryptographic keys
- …but we can derive cryptographic keys from those passwords (through a KDF).
- Main problem is that a KDF doesn't solve their inherent weaknesses:
  - low entropy;
  - vulnerable to *dictionary attacks*.
- Despite all their drawbacks, passwords remain a prevalent resource in information security (user authentication; keystore protection; etc.)

Password-Based KDF

- Passwords/Passphrases cannot be used directly as cryptographic keys
- …but we can derive cryptographic keys from those passwords (through a KDF).
- Main problem is that a KDF doesn't solve their inherent weaknesses:
  - low entropy;
  - vulnerable to *dictionary attacks*.
- Despite all their drawbacks, passwords remain a prevalent resource in information security (user authentication; keystore protection; etc.)

- It is recommended that a KDF, to be used with weak secrets, implement countermeasures that minimise the problems mentioned above:
  - use of a **salt** — a random number that randomises the output of the OW-function. It helps defending against attacks that use pre-computed tables (e.g. rainbow tables);
  - be **resource intensive** (CPU time and/or memory) to make dictionary attacks more difficult.
- Example: PBKDF2
  - Parametric on some MAC *H;*
  - uses a salt *S* and a iteration count *c*;
  - produces a secret of some given length.



Password protection

- Certain applications, and specifically *password protection*, recommend stricter counter-measures to dictionary attacks.
  - account for the possibility of attackers build dedicated password-cracking machines;
  - go beyond computational intensive: *memory-hard* designs.
- Some algorithms:
  - **bcrypt** (1999) — support adjustable cost;
  - **SCrypt** (2008) — combines PBKDF2 with a special construction (ROMix) making it resistant to hardware brute-force attacks;
  - **Argon2** (2015) — winner of the Password Hashing Competition.

Security balance

- …but in the end, security will always depend on the entropy of the passphrase!

Estimated cost of hardware to crack a password in 1 year.

KDF	6 letters	8 letters	8 chars	10 chars	40-char text
DES CRYPT	< $1	< $1	< $1	< $1	< $1
MD5	< $1	< $1	< $1	$1.1k	$1
MD5 CRYPT	< $1	< $1	$130	$1.1M	$1.4k
PBKDF2 (100 ms)	< $1	< $1	$18k	$160M	$200k
bcrypt (95 ms)	< $1	$4	$130k	$1.2B	$1.5M
scrypt (64 ms)	< $1	$150	$4.8M	$43B	$52M
PBKDF2 (5.0 s)	< $1	$29	$920k	$8.3B	$10M
bcrypt (3.0 s)	< $1	$130	$4.3M	$39B	$47M
scrypt (3.8 s)	$900	$610k	$19B	$175T	$210B

Key-Management

Cryptographic Keys

- A critical factor in the security of cryptographic techniques is the quality of the keys that are used.

  …always use Cryptographic Secure Random-Number Generators!

- Their shape (and specifically, their size) depend on the specific technique. For symmetric cryptography, keys are usually random bit strings (sizes ranging between 128 and 256 bits).
- Extreme care must be taken when handling cryptographic keys (storage; backup; disposal; etc.)

Key handling

- As a rule of thumb:
  - the *key lifetime* **(cryptoperiod)** shall be as short as possible,
  - and inversely proportional to their use and the criticality of the protected data.
- Keys are classified as
  - **Long-term** keys (aka **static** keys);
  - **Short-term** keys (aka **ephemeral** keys).
- As we will see, symmetric cryptography favours short-term keys.
- E.g. the establishment of a secure-channel should rely on session keys:

  A **session key** is a single-use symmetric key used for safeguarding communication during a specific interaction (a **session**).
- On the programming level, cryptographic APIs typically handle keys as an *abstract datatype*.
  - allow strict control on the available operations to manipulate keys;
  - support specific safeguards (e.g. clear the memory after use).

Key distribution

- The pre-distribution of keys is the major challenge in using symmetric crypto.
- They typically depend on secure-channels, which are costly.
- Notice the circularity:

  Cryptography can be used to establish a secure channel between two parties, but itself depends on the existence of a secure channel to distribute the secret key.
- Remembering the recommendation to use session keys makes the problem worse!
- We will see that asymmetric cryptography offers more attractive solutions, but is instructive to consider how the use of symmetric cryptography can obviate the problem.

Key distribution protocols

- In a community with N users, the number of keys each member needs to store to allow secure communication between any pair of members is (N * (N - 1))/2.
  - This is clearly not feasible in a community of considerable size (e.g. internet).
- **Key Distribution Protocols** make use of a "trust network" to distribute the keys between concerned parties.
  - E.g.: a Trusted Third-Party (TTP) shares a key with each member;
  - The TTP generates and distributes a session-key between any pair of agents on demand.
- But these protocols are still not suitable for open communities (such as the Internet).

Guidelines for key-management

- Keys shall not be used for multiple purposes (use instead KDFs to derive different keys);
- In security-critical operations, key-handling (generation; storage; use; etc.) should be restricted to **Hardware Security Modules** (**HSM**).
- Keys should be deleted when compromised or expired (but different applications can adopt different policies)
- If key-backup mechanism is needed, a threshold *secret-sharing scheme* should be used.

Cryptographic Key-Management Systems (CKMS)

- A CKMS consists of a set of policies, components and devices that are used to protect, manage and distribute cryptographic keys.
- A CKMS covers all states a key passes between its generation and its destruction — **key lifecycle**