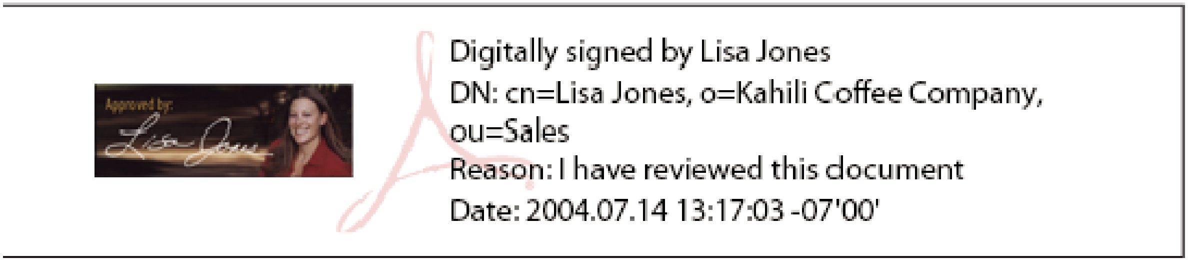
# COMPUTER SECURITY

# Cryptography: the security mechanism [1]

«Sxw#pruh#frgh/#jhu#pruh#exjv1#»

(Caeser's cipher adapted to Latin 1 (ISO 8859-1) table;
French quotes are just delimiters.)




1 However, keep in mind: «*Cryptography is rarely ever the solution to a security problem*.» (D. Gollmann, Computer Security, p. 203)

## Basics

- Originally:
  - science (and art) of secret writing
  - aimed to hinder the knowledge of sensitive information
- Currently:
  - science (and art) of providing protection mechanisms to ensure security properties (confidentiality, integrity...)
  - aims to permit control of access to information
- Relevant types of professionals:
  - cryptographers - try to master and enhance that access control
  - cryptanalysts - try to break the enabled access control

## Practical uses

- Traditional:
  - **conceal** information *P*, by making it unintelligible (*C*)
- Modern:
  - unambiguously **identify** information *P*, by means of its *fingerprint* (hash, *h*)

## Traditional use of Cryptography

- information *P is* enciphered, i.e. made unintelligible, *C*
- elsewhere or later, *C* will be deciphered, to retrieve original information *P*
- both operations are usually assisted by (cryptographic) keys[1], *K*.

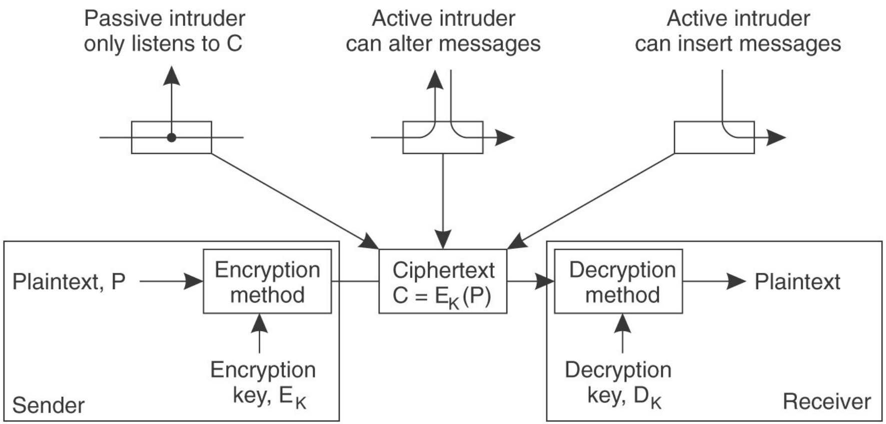



Fig. Original Cryptography: basic model of concealment and recovery of info & examples of attacks (*in* several of Tanenbaum's books).

1 pieces of information necessary for using cryptographic security mechanisms (more info, later!)

## Added newer usage of Cryptography

- information *P* is *fingerprinted*, by calculating its *digest*, or *hash*[1] *h* (array of bytes)
- elsewhere or later, *h* will be used again to detect the adulteration of the original information *P*
- usually, *hashing*[2]:
  - produces same-size values (different for each *P*!)
  - does without (cryptographic) keys

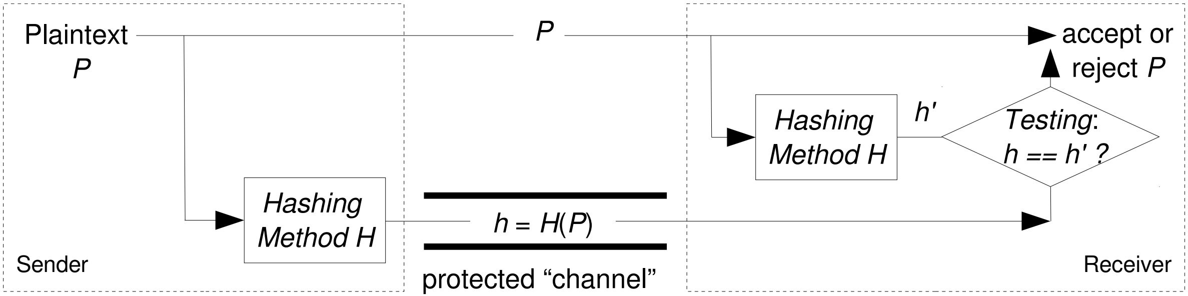



Fig. Modern Cryptography: basic model for the validation of info (e.g. integrity protection). Note the need for a protected channel!

1 PT: *síntese*, *sumário*
2 Note: *cryptographic* hashing is different from *database* hashing.

## Generic Cryptographic System

***So, involves:***

- ***P***, plaintext[1] - original, uncovered information
- ***E***, enciphering algorithm - method to conceal the info
  - or ***H***, hashing algorithm - method to transform (hash) the info
- $K_e$ , cipherkey - parameter of the concealment methods
  - normally non-existent, for hashing
- ***C***, ciphertext - hidden information
  - or ***h***, hash - transformed info
- ***D***, deciphering algorithm - method to recover the original info
  - (does not apply for hashing)
- $K_d$, deciphering key - parameter of the recovering methods
  - (does not apply for hashing)

- Note that, sometimes,
  - ***E*** = ***D***
  - $K_e = K_d$ (symmetric cryptography)

1 PT: texto inteligível

## *...Generic Cryptographic System (cont.)*

***Notation:***

- Encipher operation: $C = E_{Ke}(P)$ or $C = E(P, K_e)$ or $C = K_e(P)$
- Decipher operation: $P = D_{Kd}(C)$ or $P = D$ (C, $K_d$) or $P = K_d(C)$
    - obviously, $D_{Kd}(E_{Ke}(P)) = P$
- Cryptographic hashing: $h = H(P)$ or $F = H(P)$ or $F = h(P)$

- If
    - $K_e = K_d$ --> symmetric cryptography
    - $K_e \neq K_d$ --> asymmetric cryptography
        - $K_e = K^+$ ; $K_d = K^-$ --> public-key cryptography

# Breaking cryptographic systems

***Attacks in traditional use***

- normal
  - only the ciphertext is available
  - try to grasp the original text or, preferably, the deciphering key
- known original text (“passively” obtained)
  - both the original text and its enciphered counterpart are available
  - try to grasp the deciphering key
- planned original text (“actively” prepared)
  - specific original texts are made to be enciphered
  - try to grasp the deciphering key

***Attacks in added recent usage***

- find collisions
  - get different documents with same fingerprint
    - any *P1*, *P2* pair (more easy - *birthday attack*)
    - another *P'* for a specific *P* original (more difficult)

### *...Breaking cryptographic systems (cont.)*

***in all cases, attack methods involve***

- mathematics
- statistics
- intuition
- for an example, see Bishop: *Introduction*, Chap.8; *Art & Science*, chap.9!

## Ideal cryptographic system's requirements

- hard to break
  - in a reasonable future horizon
  - formal proof would be nice...
- easy to use
  - otherwise will be rejected or bypassed
- if broken, easily replaceable
  - this is a must, as systems **will** be broken!
  - depends on what was broken (type of secret)

# Classification of cryptographic systems

| *Perspective* | *Variant* | *Sub-variant* | *Examples* |
|---|---|---|---|
| on the secret | secret algorithm | | RC4 (originally) |
| | secret key(s) | single key, shared-key, symmetric | AES |
| | | two-key, public key, asymmetric | RSA |
| on the method | stream[1] | | RC4, One-time pad |
| | block | (pure) | AES, RSA[2] in ECB |
| | | *mixed* | AES in CBC |
| on the purpose | bidirectional, reversible, two-way | confidentiality[3] | AES |
| | | authentication[4] | RSA |
| | unidirectional, irreversible, one-way | | MD5, SHA-2 |
| | mixed | (confidentiality & integrity) | AES-CBC-HMAC-SHA1 |

1 PT: *contínuo*, *sequencial*
2 Many authors do not ever classify asymmetric systems (e.g. RSA) as "block", because of their inherent inefficiency...
3 Keys are temporary and efficient
4 Keys are personal and durable (long-lasting)

# Classification of cryptographic systems: *on the secret*

## Types of secret

- secret algorithm
- secret key(s)

## Secret algorithm systems

***Example:***

- Discover the algorithm[1] that turns the phrase (French quotes are just delimiters):

  `«Put more code, get more bugs.»`

  into

  `«Wklt!jx%f){xm ~v”cojrvmx|!54w»`

***Use:***

- typically in military systems; also in commercial ones

1 and then tell me about it, because I have forgotten the algorithm!

## Secret key's systems

- single key
- two-key

***Example:***

- Knowing that a variant of "Caesar's cipher"[1] is being used (adapted to Latin 1, ISO 8859-1, table), find the "key" that turns the phrase:

  «`Put more code, get more bugs.`»

  into

  «`Sxw!npwj%lxmp7+igv#pruh#j}ou0`»

***Use:***

- common in many military, commercial and personal applications
- often, the two variants are used in conjunction (more on this later...)

1 apparently, the original Caesar's cipher used a simple "3" as key

## Enciphering systems with key

***Symmetric, secret key, shared key***

- $K_e = K_d = K$
- heuristic constructions:
  - very efficient computation; so, very suitable for large amounts of data
- difficult combination and sharing of key; so, preferred for closed environments
- *e.g. AES (Advanced Encryption Standard)*

***Asymmetric, public key, double-key***

- $K_e = K^+ \neq K_d = K^-$
- math-based constructions:
  - very heavy computation; so, not suitable for large amounts of data
- easy combination and exchange of keys; so, ideal for open environments
- e.g. RSA (*Rivest-Shamir-Adleman*)

# Classification of cryptographic systems: *on the method*

## Enciphering methods for “long” texts

- Encipher (and decipher)[1] operations have to be done in pieces (blocks)
  - pieces could be of 1 b, 1 B, 8 B,…
    - typical: 8 B (64 b) and 16 B (128 b)
- So, *plaintext P* is divided into parts of equal size:
  - P = $P_1 P_2$...
  - each, is separately enciphered by one of the methods:
    - stream
    - block
    - “mix” of previous...

***Exercise:***

- In practice, almost any text is "long". Why?

1 and hash

## Stream method

- each part is enciphered with a different key $K = K_1K_2...$
- $C = K(P) = K_1(P_1)K_2(P_2)...$
- if the number of keys is smaller than the number of parts, the method could be periodic (e.g. Vigenère's algorithm)
- Examples: Ronald Rivest's RC4 (ARC4), *one-time pad*

***Example:***

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| P: | Put | mo | re | cod | e, | get | mo | re | bug | s. |
| key: | 3 | 1 | 5 | 9 | 11 | 2 | 3 | 3 | 8 | 2 |
| C: | Sxw | !np | wj% | lxm | p7+ | igv | #pr | uh# | j}o | u0 |

## ...Classification of cryptographic systems: *on the method – stream (cont.)*

### Example of cryptographic technique: *One-time Pad*

- stream-type system
- random key (or cryptographically secure pseudo-random...)
- size of key equal to the the original text's
- key used only once
- $E = D =$ XOR ($\oplus$)
- Advantages: proved unbreakable
- Disadvantages: exercise!

***enciphering:***

- original text: $P$
- key: $K$
- enciphered text: $C = P \oplus K$

***deciphering:***

- $P = C \oplus K$

***Exercise:***

- The system is considered unbreakable; why is it not very much used?

## ...Classification of cryptographic systems: *on the method*

# Block Method

- each part of text, block, is enciphered with the same key *K*: ECB[1] mode
- $C = K(P) = K(P_1)K(P_2)$...
- Examples: AES, RSA[2]

***Example:***

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| P: | Put | mo | re | cod | e, | get | mo | re | bug | s. |
| key: | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| C: | Sxw | #pr | uh# | frg | h/# | jhu | #pr | uh# | exj | v1# |

1 Electronic Code Book

2 Many texts do not consider RSA to be a block cipher, as it is not efficient enough to be used consecutively (block after block) in long documents. E.g., see section 3.5 of Peter Gutmann, *Lessons Learned in Implementing and Deploying Crypto Software*, 11th USENIX Security Symposium, 2002.

## ...Classification of cryptographic systems: *on the method - block (cont.)*

***Serious problem:***

- with this method, identical blocks give identical codes!
- visual example:

a) 


b) 


Fig. a) original picture; b) enciphered with AES 256b, ECB mode

***Solutions to the problem[1]:***

- mixing additional (and different) information per block!
  - Exs: CBC – Cipher Block Chaining; OFM – Output Feedback Mode; CTR – Counter Mode; ...
  - use random initialization values: IV (*initialization vector*)
- "solutions" are usually called "<u>modes of operation</u>" (of block ciphers...)

1 see More Advanced Topics

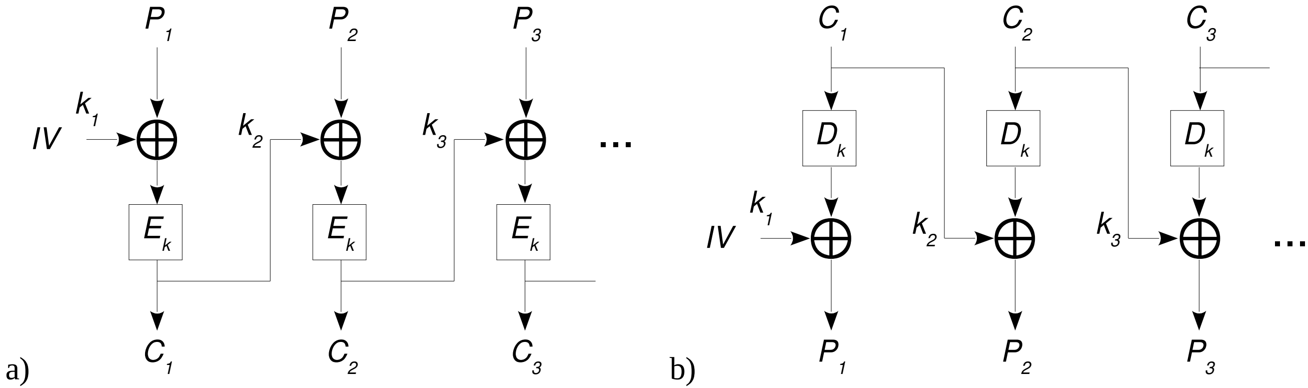



Fig. Use of CBC (*cipher block chaining*): a) enciphering; b) deciphering.

***Exercise:***

- Write the formulas for the encipherment (C as function of P) and vice versa.

## ...Classification of cryptographic systems: *on the method… CTR, Counter Mode*

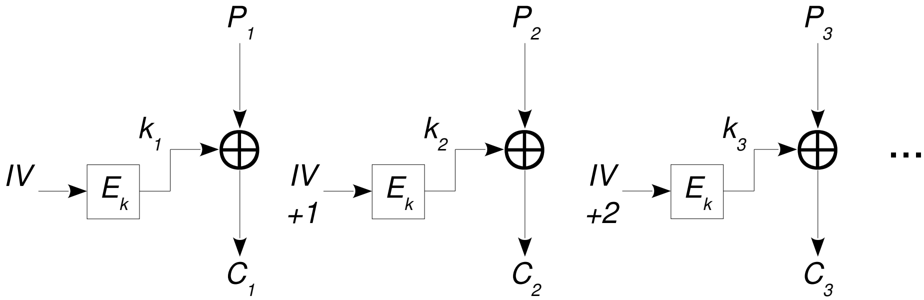



Fig. Use of CTR (*counter mode*) in enciphering mode.

***Exercises:***

- Write the formulas for the encipherment (*C* as function of *P*) and vice versa.
- Draw a corresponding picture for the decipherment in CTR.

## Padding

- size of block varies (in bits or bytes)
  - so, final block might need to be "padded"!
- important topic as padding is an attack vector!
- several schemes
  - e.g. PKCS[1] #7 – grain is byte; add (block_size - P_length mod block_size) bytes; all with value equal to number of added bytes: e.g. if 3 bytes are needed to complete last block, each added byte's value is 3
- some "modes of operation" do not need padding (why?)[2]

***Final comment on solutions for the Block method:***

- several of these methods are still vulnerable (e.g. see Kaufman et. al, Network Security, pp. 98-101)
- counter-measures: use authenticated modes[3], safer algorithms...

1 Public Key Cryptography Standards
2 see More Advanced Topics
3 see More Advanced Topics

# Classification of cryptographic systems: *on the purpose*

## Purpose types

- bidirectional, reversible (*two*-*way*)
- unidirectional, irreversible (*one*-*way*)
- “mix” of previous

# Reversible (or bidirectional, *two-way*) encipherment:

***Usage area***

- Confidentiality
- (Authentication)[1]
- ((Integrity checking))[2]

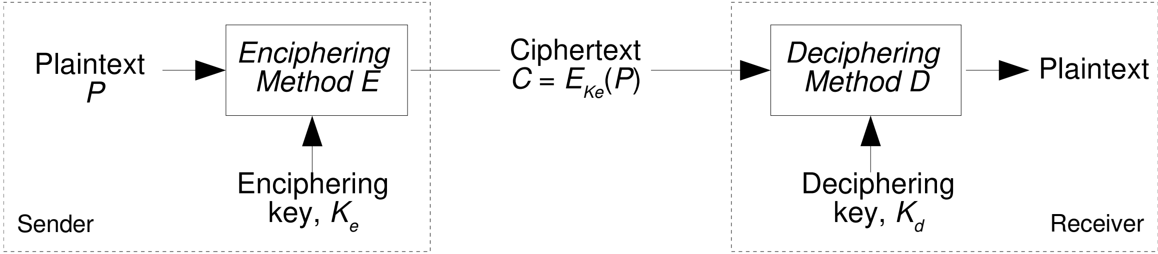



Fig. Usage of two-way cryptography.

1 not main purpose
2 difficult to use in practice, as vulnerable to attacks

## (Desired) Properties of the bidirectional algorithm:

***Simplicity:***

- the enciphering of the *plaintext* $P$ (with $K_e$) is (relatively) easy;
- the deciphering of the *ciphertext* $C$ (with $K_d$) also is.

***Resistance:***

- given a plaintext $P$ and its ciphered counterpart $C$, it is impractical[1] to compute the key $K$, used to produce $C = E_K(P)$

***Uniqueness:***

- given a plaintext $P$ and a key $K$, it is impractical to compute another key $K'$ such as $E_K(P) = E_{K'}(P)$

1 impractical = currently, computationally infeasible

# Irreversible (or unidirectional, *one-way*) "encipherment"[1]:

***Use area***

- Authentication
- Integrity checking

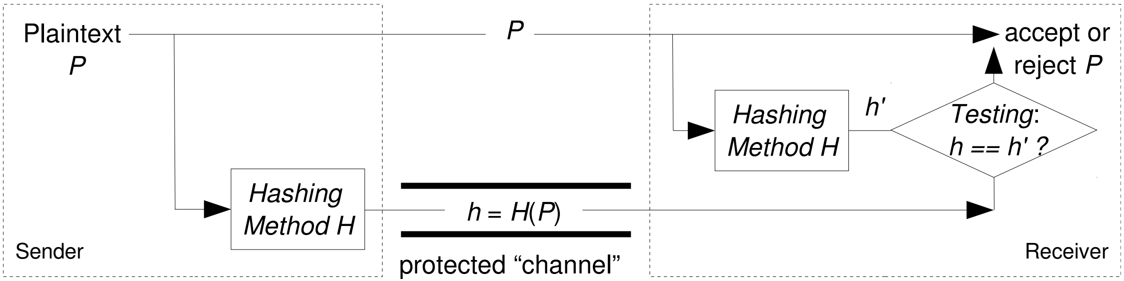



Fig. Usage of one-way cryptography for integrity checking.

1 more like *transformation*

## ...Classification of cryptographic systems: *on the purpose… Irreversible (cont.)*

***Basic idea:***

- from an original text, compute an array of bytes that is characteristic of the text (*hash value*, *digest*, *fingerprint* [, *checksum*[1]])
- (The original text is not recoverable from the hash!)

***Usually,***

- a key is not necessary: $h = H(P)$
- the hash value has a fixed length
- the hashing function is somewhat akin to database dispersion functions, but has very different features and purpose

1 *checksum*'s common use (in communication coding) does not convey needed cryptographic strength (e.g. uniqueness)

## (Desired) Properties of the unidirectional algorithm:

***Simplicity:***

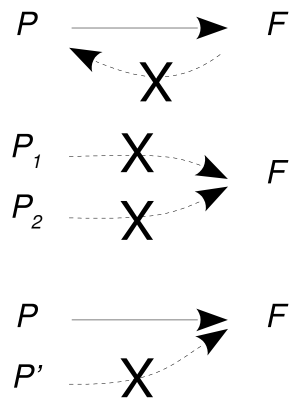
- the transformation of the original text is easy

***No reversibility (or pre-image resistance):***

- it is impractical[1] to invert the function $H$: $P \neq H^{-1}(F)$ [2]

***Uniqueness (or collision resistance):***

- it is impractical to find two texts *P1* and *P2* such that $H(P1) = H(P2)$
- **Variant**: *weak collision resistance*[3] or *2nd pre-image resistance*
  - for a given specified text $P$, it is impractical to find a text $P'$ such that $H(P) = H(P')$.




1 impractical = currently, computationally infeasible
2 *F* from Fingerprint
3 "weak", because this type of collision resistance is more easily achieved

# Mix of bidirectional encipherment and unidirectional transformation:

***Use area***

- Confidentiality **&** Integrity protection
  - so called *Authenticated Encipherment*[1]

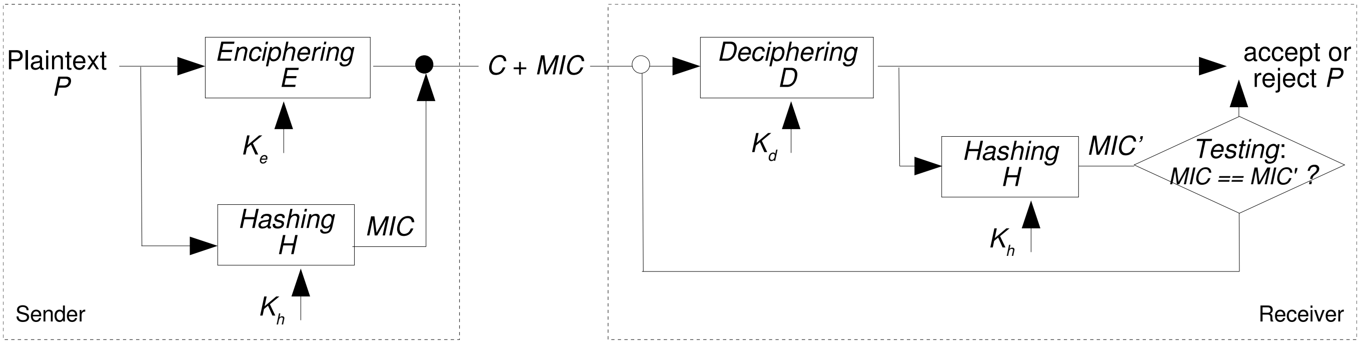



Fig. Usage of mixed two-way and one-way cryptography.

1 see More Advanced Topics

## (Desired) Properties of the *mixed* algorithm:

***Simplicity:***

- the enciphering/enciphering of the *plaintext P*/*ciphertext C* is (relatively) easy;
- the unidirectional transformation for integrity protection also is.

***Resistance:***

- given *P* / *C* pairs, it is impractical[1] to compute the key *K*, used to produce $C = E_K(P)$

***Uniqueness:***

- given a *P* / *C* pair it is impractical to compute another *C'* such that will, unnoticed, decipher into a different *P'*

1 impractical = currently, computationally infeasible

# Randomness

- essential in Cryptography!
  - one time pad, IV (initialization values), stream cipher seeds
  - hashes
  - *nonces*, key generation (TLS, RSA…) ...
- generation
  - excellent: physical source
    - inherent
      - radioactive decay, Brownian movement, ...
    - depending on initial conditions
      - (non-biased) roulette or dice, ...
  - reasonable: algorithmic-based with physical seed
    - *cryptographically secure pseudorandom number generators*
      - use physical (hopefully random) sources (e.g. mouse movements)
      - Linux's `getrandom()` (`/dev/random`, `/dev/urandom`)
  - bad: algorithmic-based
    - *pseudorandom number generators*
      - POSIX's `random()`

## *...Randomness*

- random oracle
  - (ideal) function that
    - for each input, outputs a unique and (truly) random value, uniformly distributed in the infinite output codomain;
    - is deterministic: always outputs the same value every time the same input is submitted.
  - idealized cryptographic hash function (for finite output codomain)
  - used as a reference for proofs of security of algorithms, protocols...

# Cryptographic libraries

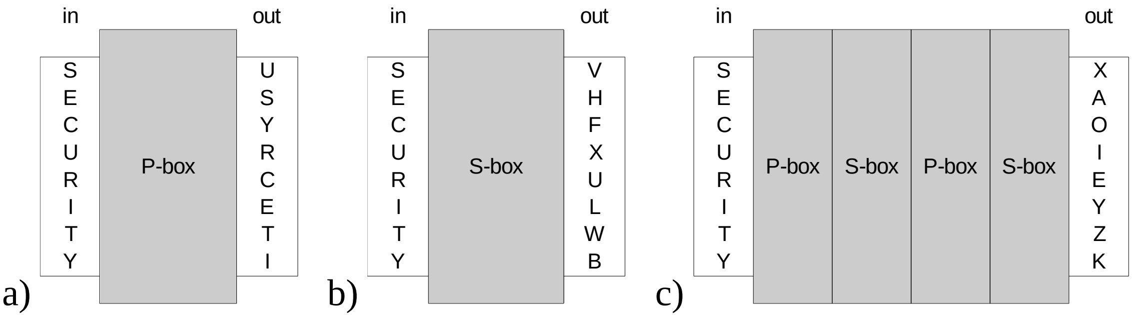
- essential in cryptographic programming
  - encryption, hashing, signing… different algorithms... all ready to use
    - perhaps, coupled with a "*cryptographically secure pseudorandom number generator*"
  - why not build a library?
    - it is not just implementing algorithms, it's how they are used, how "random" numbers are generated and chosen, etc.
- examples
  - OpenSSL: the reference![1]
    - components: application & C library
    - EVP (envelope) API level (lab classes)
  - WebCrypto: JavaScript (by W3C)
  - Bouncy Castle: Java and C# (by Australia's Legion...)
  - Libgcrypt: C (OpenPGP library)  (by GnuPG community)
  - PyCryptodome: Phyton

1 in spite of same infamous bugs, such as *The Heartbleed Bug* (heartbleed.com)

# Cryptographic transformations

- Transposition – exchange (swapping) of positions of elements – *P-box*
- Substitution – exchange of elements (e.g. Caeser's cipher) – *S-box*
- Combination - transposition and substitution cascade – *product cipher*

- Shannon: operations should cause
  - *diffusion* – each plaintext char affecting many ciphertext chars
  - *confusion* – ciphertext depending complexly on key




Cryptographic transformations: a) permutation box; b) substitution box; c) "complete" system.
Exercise: find out the algorithms for P- and S- boxes and validate them with c).

# Some famous cryptographic algorithms

- RC4: stream key generation (1987, needs medication)
- DES: reversible system, secret key (1975, defunct)
- AES: reversible system, secret key (1998, still healthy)
- RSA: reversible system, public key (1977, still healthy)
- MD5: irreversible system (1992, defunct)
- SHA-2[1]: irreversible system (2001, still healthy)
- SHA-3[2]: irreversible system (2015, yet in phase of wide adoption)

1 About SHA-1 end of life, see sha-mbles.github.io

2 Is based on new paradigm - sponge construction (keccak.team/sponge_duplex.html).

# Some numbers...

- $2^8 = 256$ — number of values represented by a byte
- $2^{32} = 4\ 294\ 967\ 296$ — maximum number of IPv4 addresses
  $\simeq$ 0,6 * number of people on Earth in 2018
- $2^{56} = 72\ 057\ 594\ 037\ 927\ 936$ — number of different keys for DES algorithm
- $2^{64} = 18\ 446\ 744\ 073\ 709\ 551\ 616$
  1+ number of grains of wheat in chess board (from 1, doubled in each square)
- $2^{76} \simeq 7 \times 10^{22}$ — mass of the Moon in kg
- $2^{79} \simeq 6 \times 10^{23}$ — Avogadro's constant
- $2^{82} \simeq 6 \times 10^{24}$ — mass of the Earth in kg
- $2^{101} \simeq 2 \times 10^{30}$ — mass of the Sun in kg
- $2^{128} = 340\ 282\ 366\ 920\ 938\ 463\ 463\ 374\ 607\ 431\ 768\ 211\ 456$
  maximum number of IPv6 addresses
- $2^{256} \simeq 10^{77}$ — number of values of SHA-256 hash
- $2^{280} \simeq 10^{84}$ — number of fundamental particles in the observable universe

# Pointers...

- The “**Public-key cryptography paper**”, 1976 – W. Diffie , M. E. Hellman
  - www-ee.stanford.edu/~hellman/publications/24.pdf
- The “**RSA paper**”, 1978 - R. L. Rivest, A. Shamir, and L. Adleman
  - dx.doi.org/10.1145/359340.359342
- The “**ElGamal Signature Scheme**”, 1985 - Taher Elgamal
  - ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01057074
- The “**DES Cryptanalysis paper**”, 1977 – W. Diffie , M. E. Hellman
  - www-ee.stanford.edu/~hellman/publications/27.pdf
- The “**Rijndael, AES Proposal**”, 1999 - Joan Daemen, Vincent Rijmen
  - citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.36.640
- The “**MD5 Message Digest Algorithm**”, 1992, R. Rivest
  - tools.ietf.org/html/rfc1321
- The “**The Keccak SHA-3 submission**”, 2011, G. Bertoni et al.
  - keccak.team/files/Keccak-submission-3.pdf
- The “**Crypto Mini-FAQ**”, Internet FAQ Archives, -2014, Roger Schlafly
  - www.faqs.org/faqs/crypto/faq/