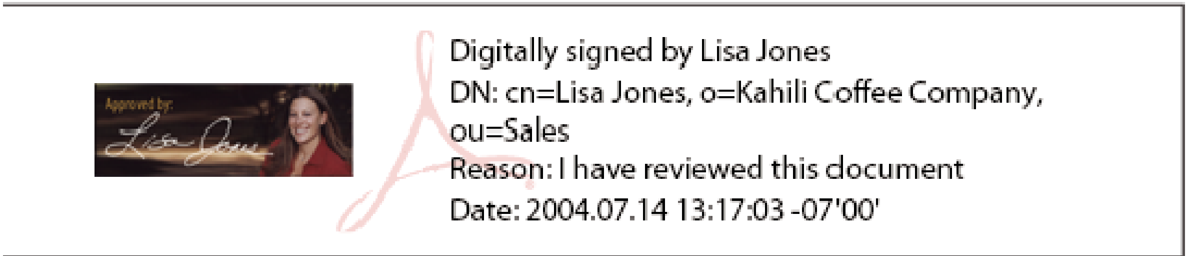
COMPUTER SECURITY

Cryptography: THE security mechanism [1]

"Security!" --> "1676c0cf7e901d443bd9cad6c5253fee"

(AES cipher, ECB mode, PKCS#7 padding, 128-b key: "I am JohnDoe 007"; 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

History

- Originally:
  - science (and art) of secret writing
  - aimed at hinder the knowledge of sensitive information
- Currently:
  - science (and art?) of providing mechanisms to ensure security properties (confidentiality, integrity...)
  - aims to control the access to information

Practical uses

- Traditional:
  - control access to information by **concealing** it, i.e. making it unintelligible
- Modern:
  - the traditional, plus
  - control access to information by **identifying** it with a *fingerprint* (or *hash*[1])
  - **support** all above uses
    - produce (almost) random numbers
    - derive secret numbers (keys)[2]

Relevant types of professionals:

- *cryptographers* - try to master and enhance that access control
- *cryptanalysts* - try to break the enabled access control

1 PT: *síntese*, *sumário*
2 pieces of data necessary for using cryptographic security mechanisms

Notation

Symbol	*Name of symbol*	*Meaning of symbol*
$\boldsymbol{P}$	plaintext[1]	original, uncovered information
$\boldsymbol{E}$	enciphering algorithm	method to conceal the info
$\boldsymbol{K_e}$	enciphering key	parameter of the concealment methods
$\boldsymbol{C}$	ciphertext	hidden information
$\boldsymbol{D}$	deciphering algorithm	method to recover the original info
$\boldsymbol{K_d}$	deciphering key	parameter of the recovering methods
$\boldsymbol{H, h}$	hash algorithm, hash value	method to transform (hash) the info, transformed info
$\boldsymbol{F}$	fingerprint, hash value	transformed info

1 PT: texto inteligível

...Basics: Notation...

Operation	*Symbolic representation*
ciphering	$C = E_{Ke}(P)$ $C = E(P, K_e)$ $C = K_e(P)$
deciphering	$P = D_{Kd}(C)$ $P = D(\mathrm{C}, K_d)$ $P = K_d(C)$
(cyptographic) hashing[1]	$h = H(P)$ $F = H(P)$ $F = h(P)$
reversing	$D_{Kd}(E_{Ke}(P)) = P$

If...	*Cryptography type*
$K_e = K_d$	symmetric
$K_e \neq K_d$	asymmetric
$K_e = K^+$ $K_d = K^-$	public-key (asymmetric)

Advance notice for Digital Signature:

$[Doc]_E$ <==> K_E^- (Doc) <==> K_E^- (H(Doc))

1 Note: *cryptographic* hashing is different from *database* hashing.

Traditional use of Cryptography

- confidentiality protection:
    - conceal information, by making it unintelligible
    - elsewhere or later, retrieve original information

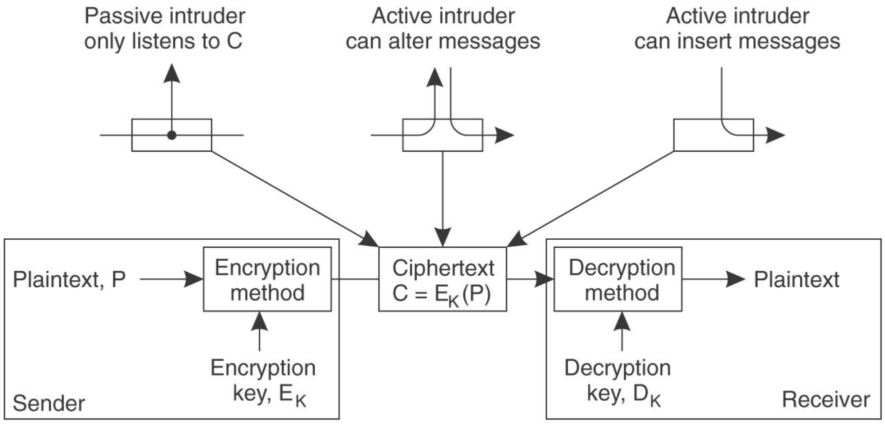


Fig. Original Cryptography: basic model of concealment and recovery of info with examples of attacks (*in* several of Tanenbaum's books).

Added, newer, usage of Cryptography

- integrity protection:
  - information is *fingerprinted*,[1] by calculating its *hash (or digest)*
  - elsewhere or later, the hash will be used to detect the adulteration of the original information

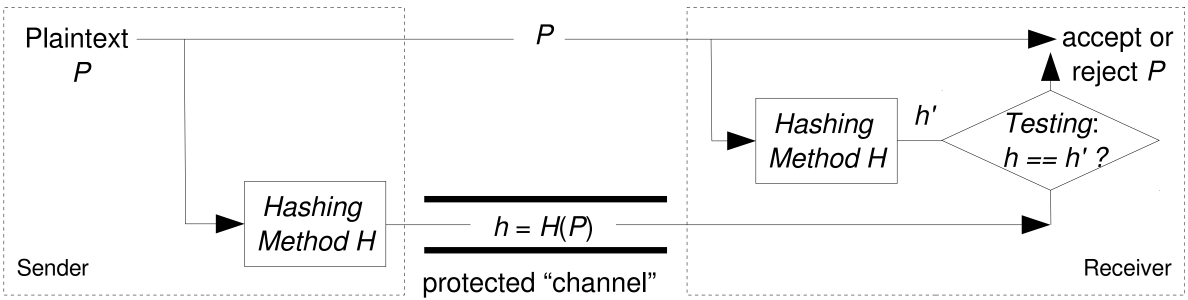


Fig. Newer use of Cryptography: basic model for the validation of info (e.g. integrity protection). Note the need for a protected channel!

1 small array of bytes that represents the original information

Breaking cryptographic systems

- Professionals: cryptanalysts, random crackers
- Methods: mathematics, statistics, intuition[1]
- Goals: depend on type of usage

Attacks in traditional use

- Goal: grasp the deciphering key! Sometimes, at least, grasp plaintexts.
- Approaches (in descending order of difficulty):
  - normal
    - only ciphertexts are available
  - known original text (“passively” obtained)
    - both some original texts and their enciphered counterparts are available
  - planned original text (“actively” prepared)
    - specific original texts are made to be enciphered

1 For an example, see Bishop: "*Introduction*", Chap.8; "*Art & Science*", chap.9.

Attacks in added recent usage

- Goal: break integrity protection
- Approaches[1] (in descending order of difficulty):
  - find collisions[2]
    - produce chosen document pairs (*birthday attack*[3])
    - produce another document for a specific original

1 The special case of "digital signatures" will be seen elsewhere.
2 meaning: different documents with same fingerprint
3 https://en.wikipedia.org/wiki/Birthday_attack

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 by users
- if broken, easily replaceable
  - this should be 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, Crypto1 ([1])
	secret key(s)	single key, shared-key, symmetric	AES
		two-key, public key, asymmetric	RSA
on the method	stream[2]		RC4, One-time pad
	block	(pure)	AES, RSA[3] in ECB ; SHA-2
		mixed	AES in CBC
on the purpose	bidirectional, reversible, two-way	symmetric (for confidentiality[4])	AES
		asymmetric (for authentication[5])	RSA
	unidirectional, irreversible, one-way	(for integrity)	SHA-2, SHA-3
	mixed	(for confidentiality & integrity)	AES-CBC-HMAC-SHA1

1 Originally, both were secret; now, they are **not**!
2 PT: *contínuo*, *sequencial*
3 Many authors do not ever classify asymmetric systems (e.g. RSA) as "block"... (more on this later)
4 usually, with temporary keys
5 main usage, with personal and durable (long-lasting) keys

On the secret

Perspective	*Variant*	*Comments*	*Exs*
on the secret ...	secret algorithm	• used in closed applications: military, commercial • not recommended by academics[1]	RC4[2], Crypto1
	secret key(s) ...	• used everywhere: military, commercial, personal applications • recommended by academics[3]	AES[4], RSA[5]

1 because, sooner or later, the secret will be discovered and a replacement is always difficult
2 Rivest Cipher 4
3 and by common sense as well, if history is something to go by
4 Advanced Encryption Standard
5 Rivest-Shamir-Adleman

...Classification of cryptographic systems: on the secret

<table>
<tr><th>Perspective</th><th>Variant</th><th>Sub-variant[1]</th><th>Comments</th><th>Exs</th></tr>
<tr><td rowspan="2">...
on the secret</td><td rowspan="2">...
secret key(s)</td><td>single key, shared-key, symmetric:
$K_e = K_d = K$</td><td>• heuristic constructions
  • very efficient computation:
    • very suitable for large amounts of data
• difficult combination and sharing of key:
  • preferred for closed environments</td><td>AES</td></tr>
<tr><td>two-key, public key, asymmetric:
$K_e = K^+$
$\neq$
$K_d = K^-$</td><td>• math-based constructions
  • very heavy computation:
    • not suitable for large amounts of data
• easy combination and exchange of keys:
  • ideal for open environments</td><td>RSA</td></tr>
</table>

1 often, the two sub-variants are used in conjunction (more on this later...)

On the method

“Long” texts[1]

- cryptographic operations[2] have to be done on (equal sized) pieces (blocks)[3]
  - typical size: 8 B (64 b) and 16 B (128 b)[4]
- enciphering (and deciphering)
  - *modes of operation*,[5] are ways to use keys in the processing of each piece
- hashing
  - does not use keys (in general)
- final piece might need to be "*padded*"[6]
  - as, in general, data size is not a multiple of the piece size

1 In practice, almost any text is "long"...
2 ciphering, deciphering, hashing
3 $P = P_1 P_2 \ldots$
4 could be 1 b, 1 B, ...
5 to be discussed later
6 to be discussed later

...Classification of cryptographic systems: on the method

Perspective	*Variant*	*Sub-variant*	*Comments*[1]	*Exs*
on the method	stream		• each piece is (de)ciphered with a different key, $K = K_1\,K_2$ ... • e.g. $C = K(P) = K_1(P_1)\,K_2(P_2)$ ...	RC4, One-time pad
	block	(pure)	• each piece is (de)ciphered with the same key, K • e.g. $C = K(P) = K(P_1)\,K(P_2)$ ... • fingerprinting does not use keys • in general, $F = H(P) = H(P_1)\,H(P_2)$ ...	AES, RSA[2] (both) in ECB[3] mode ; SHA-2, SHA-3[4]
		mixed	• each piece is (de)ciphered with a "virtual" different key, combination of the same key with additional (and different) information per block	AES in CBC[5] mode

1 will apply to (de)ciphering, as hashing does not use keys
2 Many authors 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*.
3 Electronic Code Book
4 SHA: Secure Hash Algorithms
5 Cipher Block Chaining

On the purpose

Perspective	*Variant*	*Sub-variant*	*Comments*	*Exs*
on the purpose	bidirectional, reversible, two-way	symmetric	• usage: mostly, confidentiality[1] • (see picture C below)	AES
		asymmetric	• usage: mostly, authentication[2] ; also confidentiality[3] • (see picture A1 below)	RSA
	unidirectional, irreversible, one-way		• usage: authentication, integrity • (see pictures A2, I below)	SHA-2, SHA-3
	mixed		• usage: mostly, both confidentiality & integrity • (see picture CI below)	AES-CBC-HMAC-SHA1

1 with temporary keys
2 with personal and durable (long-lasting) keys
3 of small amounts of data, e.g. symmetric keys

...Classification of cryptographic systems: on the purpose...

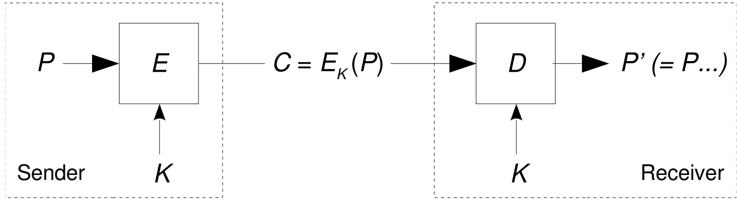


Fig C. Confidentiality with symmetric cryptography.

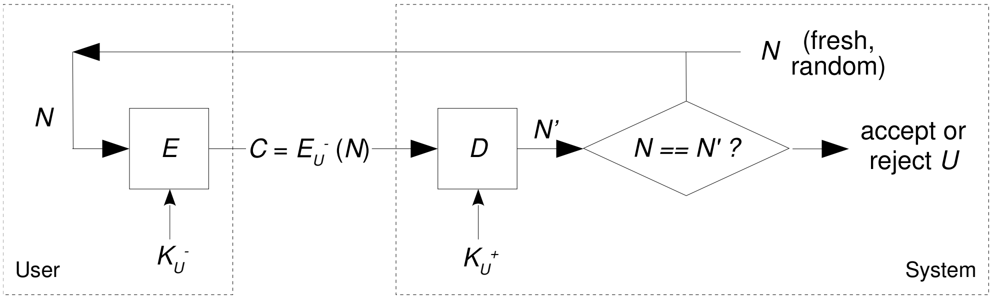


Fig A1. Authentication with asymmetric cryptography.

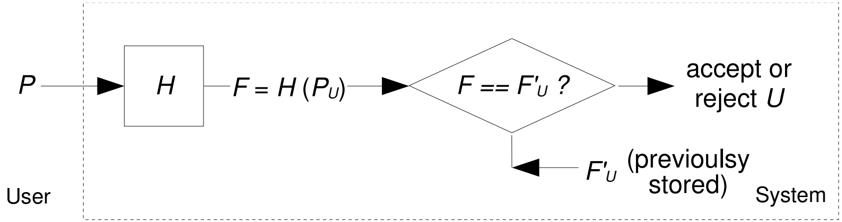


Fig A2. Authentication with cryptographic hashing.

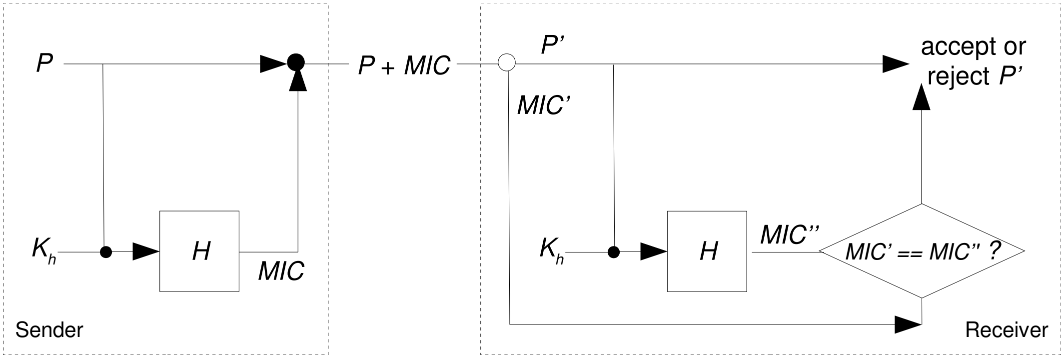


Fig I. Integrity with cryptographic hashing.

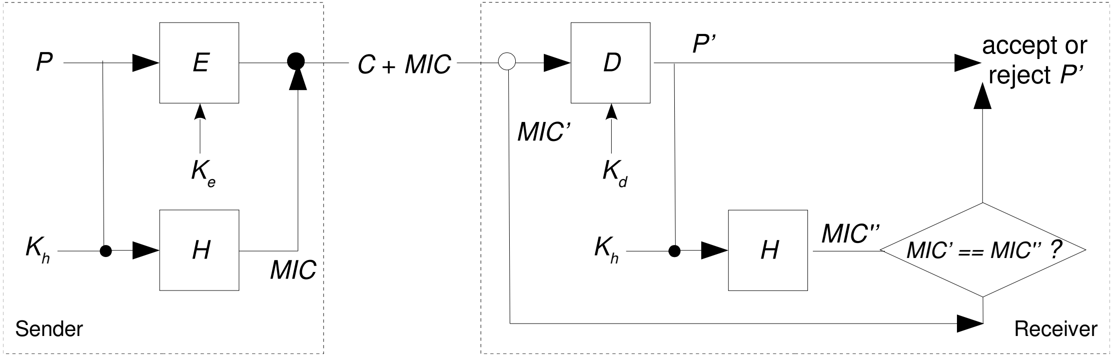


Fig CI. Confidentiality & Integrity (*Authenticated Encipherment*).

Cryptographic Keys

Definition

- cryptographic key – piece of data needed for cryptographic operations
  - usually: number or string hard to memorize
  - some times: fit to a mathematical procedure (algorithm)[1]
  - most of the times: secret

Key types of cryptographic keys

Designation	***"Owner" entity***	***Main application***	***Cryptographic type***	***Longevity***	***Efficiency***
personal	human	authentication	public-key	extended	low
session	communication channel	confidentiality	shared-key	short[2]	high[3]

1 so, user cannot "choose" it: a “cryptographic key generator” is needed
2 to be use-resistant (prevent brute-force search and repetition attacks)
3 so, can accommodate heavy traffic

Key Management

- generation
  - problem solved: just take care with choosing of seed values[1]
- storage
  - many "solutions", but still a problem swept under the rug!
- distribution
  - big problem:
    - physically separated entities must exchange/agree on cryptographic keys
  - solutions:
    - several, depending on type of cryptography (symmetric or asymmetric)
    - specific of asymmetric:
      - *public key* is distributed mostly by digital certificates

1 here, randomness is essential

Digital certificate

- document that maps an entity to a cryptographic public key
  - the mapping is guaranteed by *T*,[1] by digital signing the document[2]

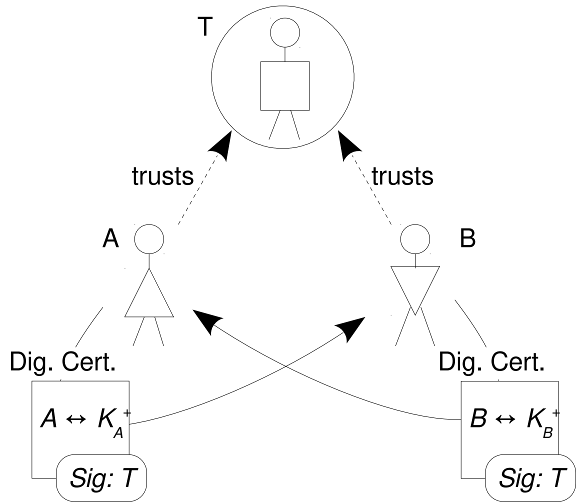


Fig. Representation of users, trusted certificate emitter and digital certificate's essential content.

1 *T* is entity trusted by *A* and *B*: 1- they believe *T* operates in an honest way; 2- they have previously exchanged cryptographic info with *T*. Usually, but not necessarily, *T* is connoted with a *Certification Authority* (CA).

2 so, assuring the legitimacy of the certificate's content; technique will be discussed later

...Key Management: Digital Certificate

Typical content

I hereby certify that the public key
19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A
belongs to
Robert John Smith
12345 University Avenue
Berkeley, CA 94702
Birthday: July 4, 1958
Email: bob@superdupernet.com

SHA-1 hash of the above certificate signed with the CA’s private key

Fig. *Really* relevant content type of a digital certificate (*in* several of Tanenbaum's books).

- identity of key's owner
- his/her public key
- identity of emitter[1]
- digital signature of emitter
- expiration date of certificate
- serial number
- specific purpose
- etc.

1 e.g. tipically, a Certificate Authority

Randomness

- essential in Cryptography!
  - one time pad, IV (initialization values), stream cipher seeds
  - hashes
  - *nonces*, key generation (e.g. asymmetric keys)...
- generation
  - excellent: physical source
    - inherent: radioactive decay, Brownian movement, ...
    - depending on initial conditions: (non-biased) roulette, 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()`

Cryptographic libraries

- essential in cryptographic programming
  - encryption, hashing, signing… different algorithms... all ready to be used
    - coupled with a "*cryptographically secure pseudorandom number generator*"
  - why not write your own library?
    - Highly dangerous! It is not just implementing algorithms, it is how they are used, how "random" numbers are generated and chosen, etc.[1]
- examples
  - OpenSSL: the reference![2]
    - components: application & C library
    - EVP (envelope) "high level" API (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 see, for instance, https://security.stackexchange.com/questions/18197/why-shouldnt-we-roll-our-own
2 in spite of same infamous bugs, such as *The Heartbleed Bug* (heartbleed.com)

Cryptographic algorithms

- RC4: stream key generation (1987, survives with medication)
- DES[1]: reversible system, secret key (1975, defunct)
- AES: reversible system, secret key (1998, still healthy)
- RSA[2]: reversible system, public key (1977, still healthy)
- MD5[3]: irreversible system (1992, defunct)
- SHA-1[4]: irreversible system (1995, defunct)
- SHA-2: irreversible system (2001, still healthy)
- SHA-3[5]: irreversible system (2015, yet in phase of wide adoption)

1 Data Encryption Standard, a landmark of cryptography
2 another landmark of (public-key) cryptography
3 yet another landmark of cryptography
4 about SHA-1 end of life, see sha-mbles.github.io
5 based on new paradigm - sponge construction (keccak.team/sponge_duplex.html)

Cryptographic transformations

- diverse, generally follow some major "patterns" for each type of cryptography
- in general, Shannon's recommended properties[1] are followed:
  - *diffusion* – each plaintext unit (e.g. char) affects many transformed units
  - *confusion* – transformed output depends complexly on key (if it exists)

Common patterns[2] - symmetric cryptography

- Transposition – exchange (swapping) of positions of elements – *P-box*
- Substitution – exchange of elements (e.g. Caesar's cipher) – *S-box*
- Combination - transposition and substitution cascade – *product cipher* [Fig.C]
- Feistel construct (e.g. 3DES) [Fig.F]
- ...

1 Those were thought to symmetric cryptography, the only that existed at the time (1949); however, they are applied, at least partially, to other, contemporary cryptographic systems.

2 or transformations

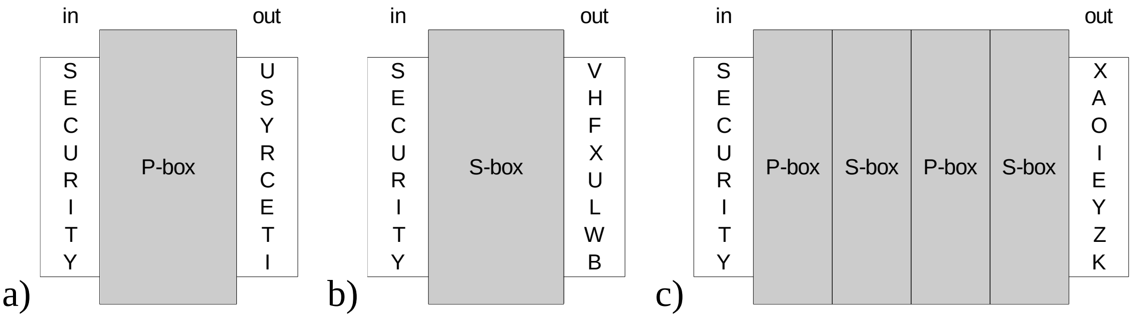


Fig C. Symmetric transformations: a) permutation box; b) substitution box; c) “complete”, product cipher. Exercise: find out the algorithms for P- and S- boxes and validate them with c).

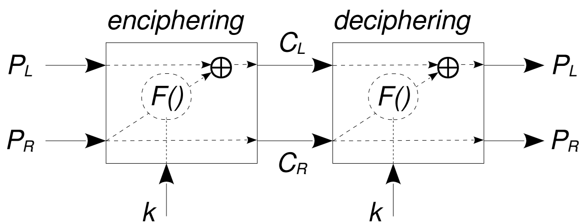
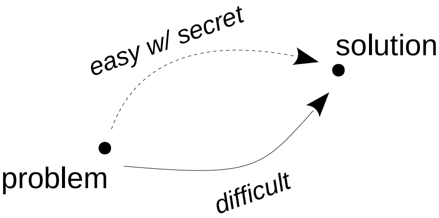


Fig F. Feistel symmetric construct "in action". *F* is Feistel (or *round*) function.

Common patterns - asymmetric constructs

- based on apparent intractability of computational problems:
  - easy to compute with knowledge of some data;
    otherwise, most difficult (intractable) to compute[1]
  - common computational problems:
    - integer factorization (e.g. RSA)
    - discrete logarithm problem (e.g. DSA)
    - ...



Common patterns - "hash" cryptography

- most common are *iterated hash functions*
  - Merkle-Damgård construct (e.g. SHA-2)
  - sponge construct (e.g. SHA-3)
  - ...

1 Informally, an easy or tractable problem can be solved in less than n^k (polynomial) time units, k being an integer and n the size of the problem (e.g. number of items to sort); a hard or intractable problem will need k^n (exponential) time units to be solved.

Some numbers...

- $2^8 = 256$ number of values represented by a byte
- $2^{32} = 4\,294\,967\,296$ maximum number of IPv4 addresses
  $\simeq$ 0,5 * number of people on Earth in 2023
- $2^{56} = 72\,057\,594\,037\,927\,936$ 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)
- $2^{76} \simeq 10^{23}$ mass of the Moon in kg
- $2^{79} \simeq 10^{24}$ Avogadro's constant
- $2^{82} \simeq 10^{25}$ mass of the Earth in kg
- $2^{101} \simeq 10^{30}$ mass of the Sun in kg
- $2^{128} = 340\,282\,366\,920\,938\,463\,463\,374\,607\,431\,768\,211\,456$
  $\simeq 10^{38}$ 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/