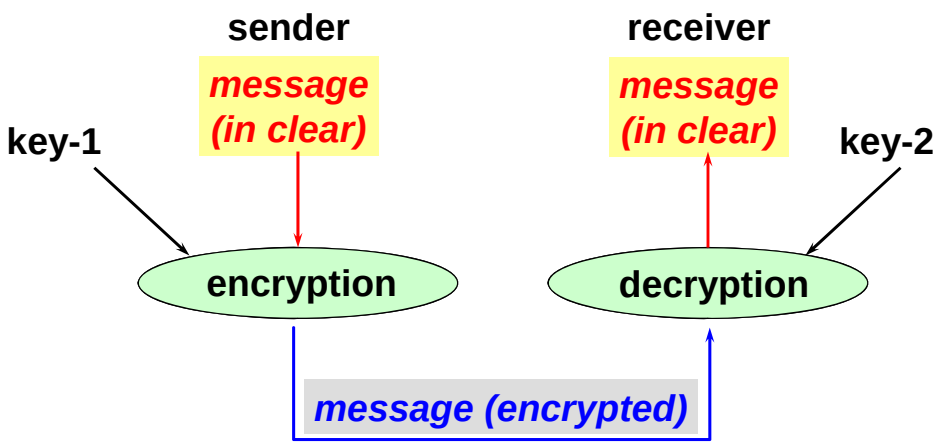
Basics of ICT security

Antonio Lioy
< lioy @ polito.it >

Politecnico di Torino
Dip. Automatica e Informatica

Cryptography



Terminology

- **message in clear:**
  - plaintext or cleartext
  - we will refer to it as P
- **encrypted message:**
  - ciphertext
  - we will refer to it as C
  - note that in some countries “encrypted” sounds offensive for religious reasons (cult of dead); in those cases “enciphered” is preferred

Cryptography’s strength (Kerchoffs’ principle)

- **if the keys:**
  - are kept secret
  - are managed only by trusted systems
  - are of adequate length
- **then ...**
- **... it has no importance that the encryption and decryption algorithms are kept secret**
- **... on the countrary it is better to make the algorithms public so that they can be widely analysed and their possible weaknesses identified**

Auguste Kerckhoffs, "La cryptographie militaire", Journal des sciences militaires, vol. IX, pp. 5–38, Janvier 1883, pp. 161–191, Février 1883.

Security through obscurity (STO)

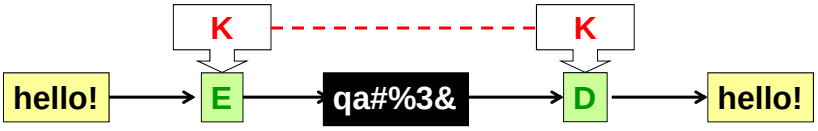
Security trough obscurity is a thing as bad with computer systems as it is with women

Secret key cryptography

- **key-1 = key-2**
- **symmetric algorithms**
- **low computational load**
- **used for data encryption**
- **main algorithms:**
  - DES, triple-DES
  - IDEA
  - RC2, RC5
  - RC4
  - AES

Symmetric cryptography

- single key
- key shared between sender and receiver (only!)
- C = enc (K , P) or C = { P } K
- P = dec (K , C) = enc^{-1} (K , C)



Symmetric algorithms

name	*key*	*block*	*note*
DES	56 bit	64 bit	obsolete
3-DES	112 bit	64 bit	56-112 bit strength
3-DES	168 bit	64 bit	112 bit strength
IDEA	128 bit	64 bit	
RC2	8-1024 bit	64 bit	usually K=64 bit
RC4	variable	stream	secret
RC5	0-2048 bit	1-256 bit	optimal when B=2W
AES	128-256 bit	128 bit	alias Rjindael

The EX-OR (XOR) function

- ideal "confusion" operator
- if the input is random (probability 0 : 1 = 50 : 50%) then also the output will be equally random
- primitive operation available on all CPU
- truth table:

$\oplus$	0	1
0	0	1
1	1	0

DES

- Data Encryption Standard
- standard FIPS 46/2
- mode of application standard FIPS 81
- 56 bits key (+ 8 parity bits) = 64 bits
- 64 bits data block
- designed to be efficient in hardware because it requires:
  - XOR
  - shift
  - permutation (!)

Triple DES (3DES, TDES)

- repeated application of DES
- uses two of three 56 bits keys
- usually applied in the EDE mode
  (for compatibility with DES when K1 = K2 = K3)
- 3DES with 2 keys (Keq=56 bit if 2**59B of memory is available, otherwise Keq=112 bit)
  $C' = enc(K_1,P)$ $C'' = dec(K_2,C')$ $C = enc(K_1,C'')$
- 3DES with 3 keys (Keq=112 bit)
  $C' = enc(K_1,P)$ $C'' = dec(K_2,C')$ $C = enc(K_3,C'')$
- standard FIPS 46/3 and ANSI X9.52

Double DES ?

- double application of encryption algorithms is subject to a known-plaintext attack named meet-in-the-middle which allows to decrypt data with at most 2^{N+1} attempts
- thus usually the double version of encryption algorithms is never used
- note: if the base symmetric algorithm would be a *group* then it would exist K_3 so that
  $enc(K_2, enc(K_1, P)) = enc(K_3, P)$

Meet-in-the-middle attack

- **hypothesis:**
  - N bit keys
  - known P and C so that C = enc (K_2, enc (K_1, P))
- **note:**
  - exists M so that M = enc(K_1,P) and C = enc(K_2,M)
- **actions:**
  - compute 2^N values X_i = enc (K_i, P)
  - compute 2^N values Y_j = dec (K_j, C)
  - search those values K_i and K_j such that $X_i = Y_j$
  - "false positives" can be easily discarded if more than one (P,C) couple is available

IDEA

- **International Data Encryption Algorithm**
- **patented but with low royalty (only for commercial use, ASCOM AG)**
- **128 bits key**
- **64 bits data block**
- **famous because used in PGP**
- **operations used:**
  - XOR
  - addition modulo 16
  - multiplication modulo $2^{16}+1$

An application of IDEA

courtesy of Ascom AG)

RC2, RC4

- developed by Ron Rivest
- RC = Ron's Code
- algorithms proprietary of RSA but not patented
- 3 or 10 times faster than DES
- RC2 is a block algorithm, RC4 is a stream one
- variable length key
- RC2:
  - published as RFC-2268 (mar 1998)
  - 8 to 1024 bits keys (usually 64 bits)
  - 64 bits data block
- RC4 reverse engineered (ARCFOUR)

RC5

- **RFC-2040**
- **B bits data block (0 < B < 257)**
- **b bytes fix/variable key (0 ≤ b < 256) that is between 0 and 2048 bits**
- **works best when B = 2 W**
- **operations used:**
  - shift
  - rotate (!)
  - modular addition
- **used in WAP**

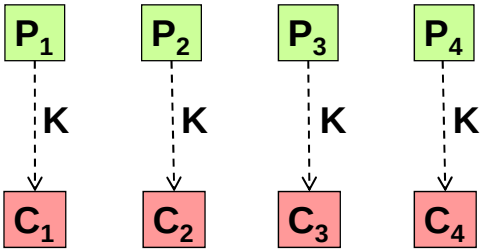
Application of block algorithms

How a block algorithm is applied to a data quantity different from the algorithm's block size?

- **to encrypt data of size > algorithm's block size:**
  - ECB (Electronic Code Book)
  - CBC (Cipher Block Chaining)
- **to encrypt data of size < algorithm's block size:**
  - padding
  - CFB (Cipher FeedBack), OFB (Output FeedBack)
  - CTR (Counter mode)

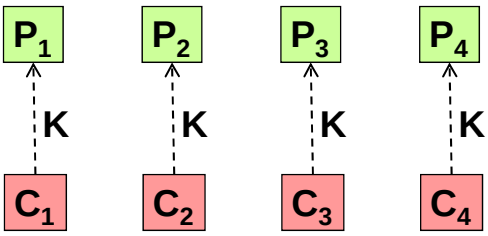
ECB (Electronic Code Book)

- formula for the i-th block:
  $C_i = enc (K, P_i)$
- NOT to be used on long messages because is vulnerable to *known-plaintext* attacks



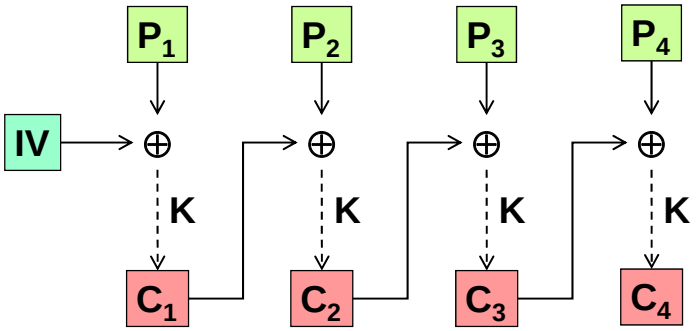
ECB - decrypt

- formula for the i-th block:
  $P_i = enc^{-1} (K, C_i)$
- an error in transmission generates an error at the decryption of one block



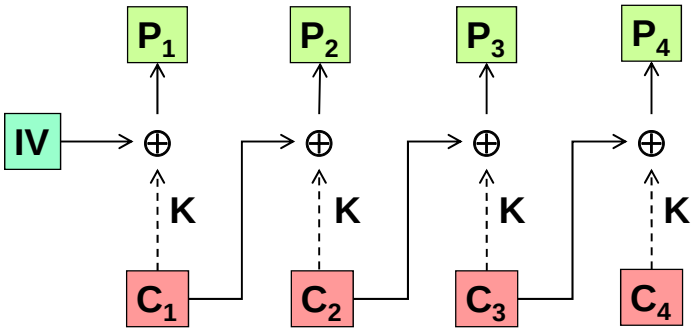
CBC (Cipher Block Chaining)

- formula for the i-th block:
  $C_i = enc\ (K, P_i \oplus C_{i-1})$
- requires C_0 = IV (Initialization Vector)



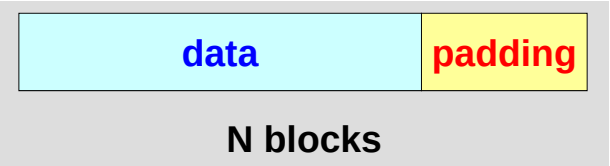
CBC - decryption

- formula for the i-th block:
  $P_i = enc^{-1}\ (K\ ,\ C_i\ ) \oplus C_{i-1}$
- requires C_0 (i.e. IV) to be known by the receiver
- an error in transmission generates an error at the decryption of two blocks



Padding (aligning, filling)

- **size of algorithm's block B**
- **size of data to process D < B**
- **add bits until size B is reached**



- **problems:**
  - transmit more data (B) than needed (D)
  - how many padding bits? (preferred D > 0.5 B)
  - value of padding bits?

Padding techniques

- **(if length is known or it can be obtained – e.g. a C string) add null bytes**
  - … 0x00 0x00 0x00
- **(original DES) one 1 bit followed by many 0**
  - … 1000000
- **one byte with value 128 followed by null bytes**
  - … 0x80 0x00 0x00
- **last byte's value equal to the length of padding**
  - … 0x?? 0x?? 0x03
  - what about the value of the other bytes?

Padding with explicit length (N)

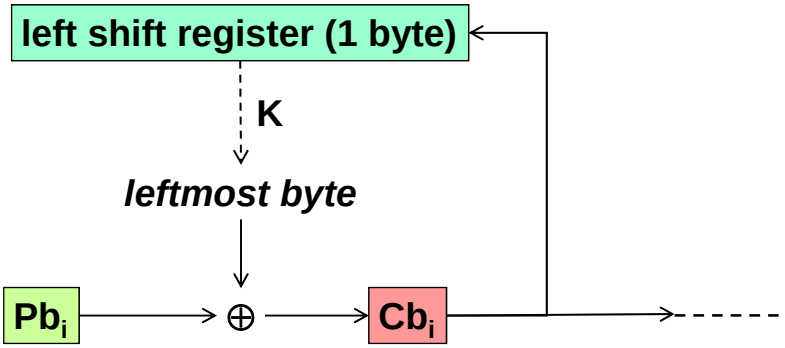
- **(Schneier) null bytes:**
  - e.g. … 0x00 0x00 0x03
- **(SSL/TLS) bytes with value N:**
  - e.g. … 0x03 0x03 0x03
- **(SSH2) random bytes:**
  - e.g. … 0x05 0xF2 0x03
- **(IPsec/ESP) progressive number:**
  - e.g. … 0x01 0x02 0x03
- **byte with value N-1:**
  - e.g. … 0x02 0x02 0x02

Padding – some notes

- **typically applied to large data, on the last fragment resulting from the division in blocks (e.g. for ECB or CBC)**
- **even if the plaintext splits into an exact number of blocks, padding bits must be added anyhow to avoid errors in the interpretation of the last block**
- **the SSH2 padding implies that equal data are encrypted to different ciphertexts**
- **the choice of the padding method for a certain algorithm determines the type of (some) possible attacks**

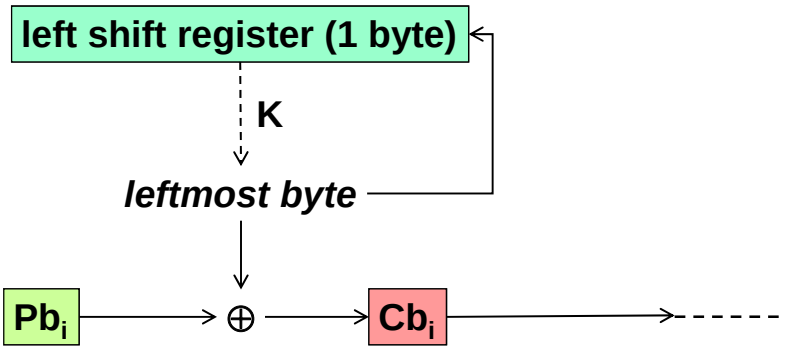
CFB (Cipher FeedBack)

- allows to encrypt N bits at a time (a group)
- requires an IV (to initialize the shift register)
- a transmission error ~ causes an error in the decryption of an entire block



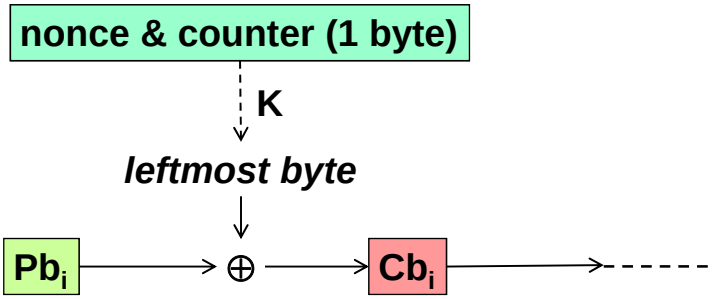
OFB (Output FeedBack)

- allows to encrypt N bits at a time (a group)
- requires an IV (to initialize the shift register)
- a transmission error ~ causes an error only in one group



CTR (Counter mode)

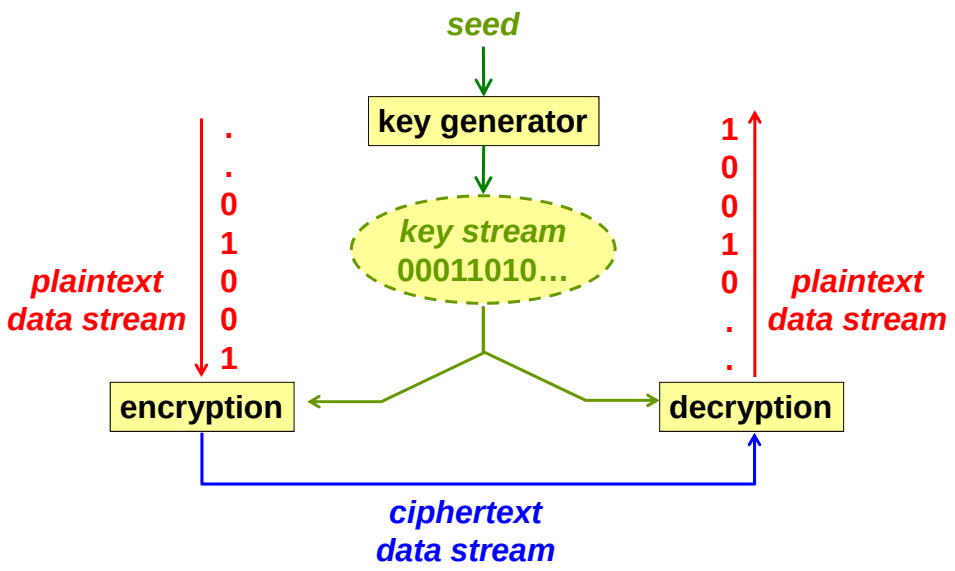
- allows to encrypt N bits at a time (a group)
- random direct access to any ciphertext group
- requires a nonce and a counter (concatenated, summed, XOR, ...)
- a transmission error ~ causes error only in one group



Symmetric stream algorithms

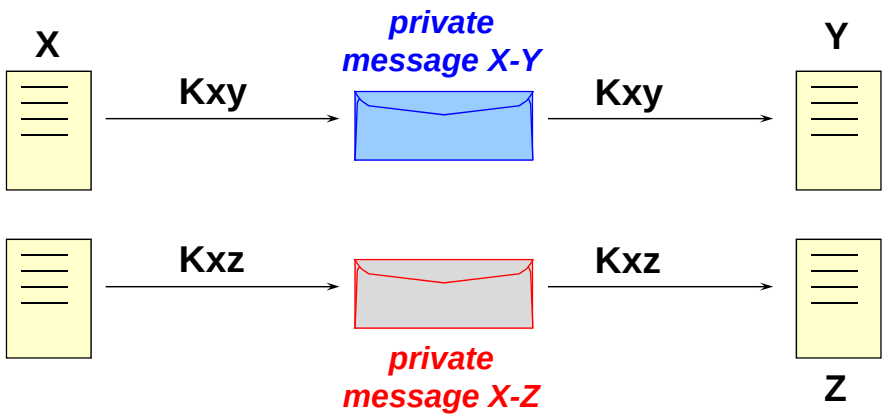
- operate on a stream of data without requiring the division on blocks, typically on one bit or one byte at a time
- ideal algorithm:
  - one-time pad (requires a key which is as long as the message to protect!)
- real algorithms:
  - use pseudo-random key generators, synchronized between the sender and the receiver
  - examples: RC4 and SEAL

Algorithms of type stream



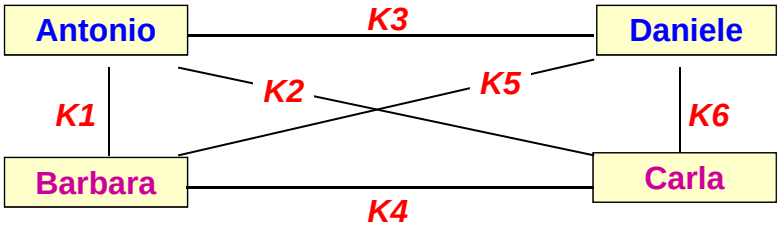
Symmetric cryptography

- single and secret key
- one key for each couple / group of users



Key distribution for symmetric cryptography

- **for a complete private communication between N parties N x (N-1) / 2 keys are necessary:**
  - distribution OOB (Out-Of-Band)
  - distribution by means of key exchange algorithms



Length of secret keys

- **if:**
  - the encryption algorithm was well designed
  - the keys – Nbit in length – are kept secret
- **… then the only possible attack is the brute force (exhaustive) attack which requires a number of trials equal to**

$$T = 2^{Nbit}$$

Length of cryptographic keys

symm.	40	64	128	...
asymm.	256	512	1024	...

low security → **high security**

DES challenges

- **2^{56} = 72.057.594 billions of possible keys**
- **DES challenge I**
  - start=18-feb-1997, fine=17-june-1997
  - 17.731.000 billions of tried keys (25%)
  - about 15.000 computer in network
- **DES challenge II**
  - start=13-jan-98, end=23-feb-98
  - 63.686.000 billion of tried keys (87%)
  - about 20.000 computer in network

The end (?) of DES

- **DES challenge III**
  - start=13-jul-98, end=15-jul-98
  - 17.903.000 billions of tried keys (25%)
  - 1 special-purpose system (DEEP CRACK) developed by the EFF at a cost of 250.000 $
- **it is thus possible to construct a computer system that can decrypt a generic DES message, but:**
  - it is necessary to know the type of data (e.g. ASCII)
  - the machine cannot decrypt 3DES messages
  - DES is not intrinsically weak, it uses only a short key!

Faster and faster

- **DES challenge IV**
  - start=18-jan-1999, end=after 22h 15m
  - 16.017.000 billions of tried keys (22%)
  - DEEP CRACK plus a few thousands of workstations
  - peak power: 250 Gkey/s
  - average power: 199 Gkey/s

What after DES?

- **IETF changes all RFC advising not to use DES and suggesting the use of triple DES**
- **RFC-4772 (security implications of using DES)**
- **a German bank sentenced for a fraud made by means of a system based on DES**
- **on 15-jan-1999 FIPS withdrawed DES (46/2) and replaced it with 3DES (46/3)**
- **competitive call of the US government for selecting a new symmetrical algorithm:**
  - AES (Advanced Encryption Standard)
  - key length at least 256 bits
  - block size at least 128 bits

AES (Advanced Encryption Standard)

- **15 candidates**
- **5 finalists (9 august 1999):**
  - MARS (IBM)
  - RC6 (RSA, i.e. Ron Rivest)
  - Rijndael (Joan Daemen, Vincent Rijmen)
  - Serpent (Ross Anderson, Eli Biham, Lars Knudsen)
  - Twofish (Bruce Schneier and others)
- **information about the selection process:**
  http://www.nist.gov/aes

AES = RIJNDAEL

- **2 October 2000**
- **RIJNDAEL chosen as winner**
- **published in November 2001 as FIPS-197**

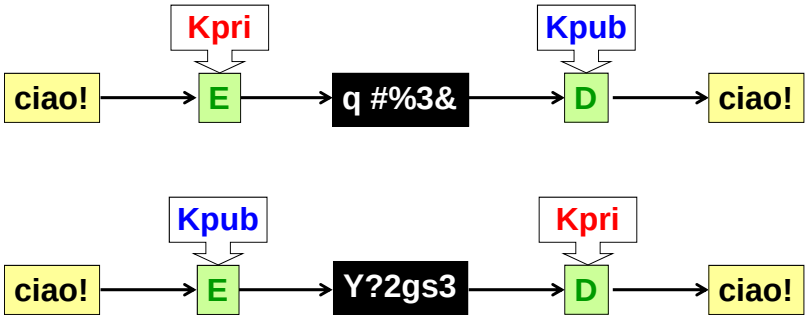
- **AES currently being implented in various applications (it takes so long because crypto algorithms are like wine: the best is the one aged for several years …)**

Public key cryptography

- **key-1 ≠ key-2**
- **asymmetric algorithms**
- **pair of keys (*public* and *private*)**
- **one of the keys is used for encryption and the other one is used for decryption**
- **processing load is high**
- **used to distribute secret keys and for the electronic signature (with hashing)**
- **principal algorithms:**
  - Diffie-Hellman, RSA, DSA, El Gamal, …

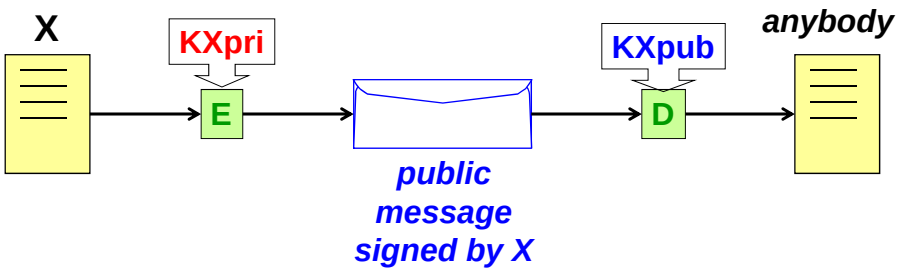
Asymmetric cryptography

- keys generated in pairs:
  private key (Kpri) + *public key* (Kpub)
- keys with inverse functionality: data encrypted with one key can be decrypted only with the other key



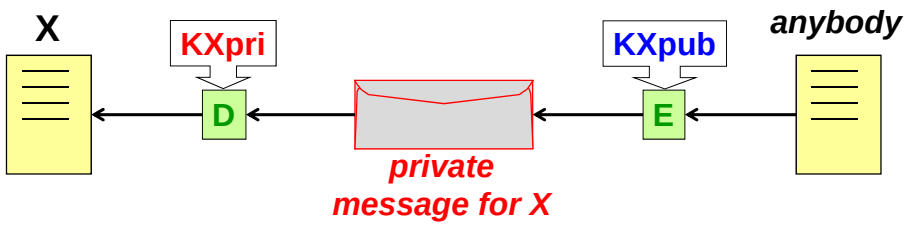
Digital signature

- digital signature = asymmetric encryption of data made with the private key of the author
- usually data is not directly encrypted but only its summary (*digest*)
- provides data authentication (and integrity)



Confidentiality without shared secrets

- it is possible to generate a secret message for a particular receiver given only its public key



Public key algorithms

- **RSA (Rivest - Shamir - Adleman)**
  - product of prime numbers, factoring of result
  - secrecy and digital signature
  - patented - only in USA - by RSA; patent expired on 20-set-2000
- **DSA (Digital Signature Algorithm)**
  - taking the power, logarithm of the result
  - digital signature only
    - for encryption use El-Gamal
  - standard NIST for DSS (FIPS-186)

RSA – the algorithm

- **public module $N = P \times Q$ known to anybody**
  P and Q are prime, large and secret
- **public exponent E arbitrarily chosen so that it is relatively prime with respect to P-1 and Q-1**
- **private exponent $D = E^{-1} \bmod (P-1) \times (Q-1)$**

- **plaintext to encrypt: $p < N$**
- **encrypt: $c = p^E \bmod N$**
- **decrypt: $p = c^D \bmod N$**
- **roles of E and D are interchangeable because**
  $(x^D)^E \bmod N = (x^E)^D \bmod N$

Modular arithmetic

- **X = A mod N**
  if X is the rest of the integer division of A by N
- **examples:**
  - 7 mod 5 = 2
  - 13 mod 5 = 3
- **optimal for security applications because it is not invertible in a unique way:**
  - given
    Y mod 5 = 2
    who is Y?
  - responses: 7, 12, 17, 22, 27, ...

Inversion in modular arithmetic

The inverse of a number X is that number X^{-1} that multiplied by X gives as result 1

- **in normal arithmetic:**
  $X = 5$ implies $X^{-1} = 1/5$
  because
  $5 \times 1/5 = 1$
- **in modular arithmetic (e.g. modulo 4):**
  $X = 5$ implies $X^{-1} \bmod 4 = \{ 5, 9, 13, ... \}$
  because
  $5 \times 5 \bmod 4 = 25 \bmod 4 = 1$
  $5 \times 9 \bmod 4 = 45 \bmod 4 = 1$
  ...

RSA - an example

- **chosen P=3, Q=5 we have N = 15**
- **E (relative prime to 2 and 4) = 7**
- **$D = 7^{-1} \bmod 8 = \{7, 15, 23, 31, \ldots\} = 23$**

- **text to encrypt: 1 2 3**
- **$c_1 = 1^7 \bmod 15 = 1$**
- **$c_2 = 2^7 \bmod 15 = 128 \bmod 15 = 8$**
- **$c_3 = 3^7 \bmod 15 = 2187 \bmod 15 = 12$**
- **$t_1 = 1^{23} \bmod 15 = 1$**
- **$t_2 = 8^{23} \bmod 15 = 2$**
- **$t_3 = 12^{23} \bmod 15 = 3$**

DoS attack on RSA

- **usually all public keys have exponent equal to 3 or to the Fermat number 65537 (0x00010001)**
  - the power operation is very easy because these numbers have only two bits set to one
    - (high) speed of the encryption operation
    - (high) speed in the operation of signature verification
  - optimized algorithms for this special case
- **attack: provide a signature made with a key whose exponent has many bits set to one, to generate a high computational load**

Length of the public keys

- **256 bits can be attacked in a couple of weeks**
- **512 bits can be attacked in a couple of months**
- **1024 bits offer an appropriate security level for a couple of centuries**

- **experimentally proved by means of the RSA challenges:**

```
http://www.rsasecurity.com/
  rsalabs/challenges/factoring/numbers.html
```

RSA challenges

- **solved challenges (old style):**
  - 10-apr-1996, RSA-130, 1000 MIPS-years
  - 2-feb-1999, RSA-140 (465 bits), 2000 MIPS-years
  - 22-aug-1999, RSA-155 (512 bits), 8000 MIPS-years
  - 9-may-2005, RSA-200 (663 bits), ~75 years Opteron 2.2 GHz
- **solved challenges (new style):**
  - 3-dec-2003, RSA-576 (174 decimal digits)
  - 2-nov-2005, RSA-640 (193 decimal digits)
  - 12-dec-2009, RSA-768 (232 decimal digits)

Twinkle (!?)

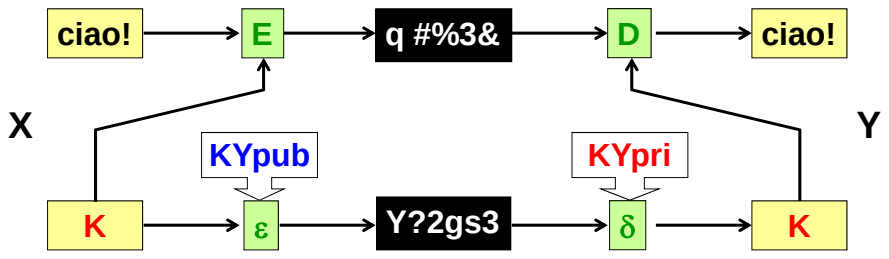
An Analysis of Shamir's Factoring Device
Robert D. Silverman
RSA Laboratories
May 3, 1999
At a Eurocrypt rump session, Professor Adi Shamir of the Weizmann Institute announced the design for an unusual piece of hardware. This hardware, called "TWINKLE" (which stands for The Weizmann INstitute Key Locating Engine), is an electro-optical sieving device which will execute sieve-based factoring algorithms approximately two to three orders of magnitude as fast as a conventional fast PC. The announcement only presented a rough design, and there are a number of practical difficulties involved with fabricating the device. It runs at a very high clock rate (10 GHz), must trigger LEDs at precise intervals of time, and uses wafer-scale technology. However, it is my opinion that the device is practical and could be built after some engineering effort is applied to it. Shamir estimates that the device can be fabricated (after the design process is complete) for about $5,000.

Key distribution for asymmetric cryptography

- **private key never disclosed!**
- **public key distributed as widely as possible**
- **problem: *who guarantees the binding (correspondence) between the public key and the identity of the person?***
- **solution #1: exchange of keys OOB (e.g. key party!)**
- **solution #2: distribution of the public key by means of a specific data structure named public key certificate (= digital certificate)**
  - format of the certificate?
  - trust in the certificate issuer?

Secret key exchange by asymmetric algorithms

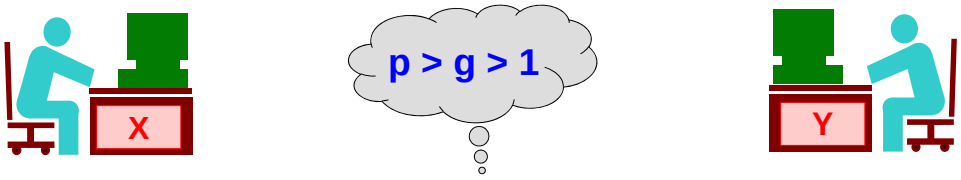
- **confidentiality without shared secrets is often used to send the secret key chosen for a symmetric algorithm**



Diffie-Hellman

- A and B choose two big integers p (prime) and g so that:
  $1 < g < p$
- A chooses a number x arbitrarily and computes:
  $X = g^x \bmod p$
- B chooses a number y arbitrarily and computes:
  $Y = g^y \bmod p$
- A and B exchange among themselves (publish) X and Y
- A computes $K = Y^x \bmod p$
- B computes $K' = X^y \bmod p$
- but $K = K' = g^{xy} \bmod p$

Diffie-Hellman (DH)



$A = g^X \bmod p$ $B = g^Y \bmod p$

$\xrightarrow{\quad A \quad}$

$\xleftarrow{\quad B \quad}$

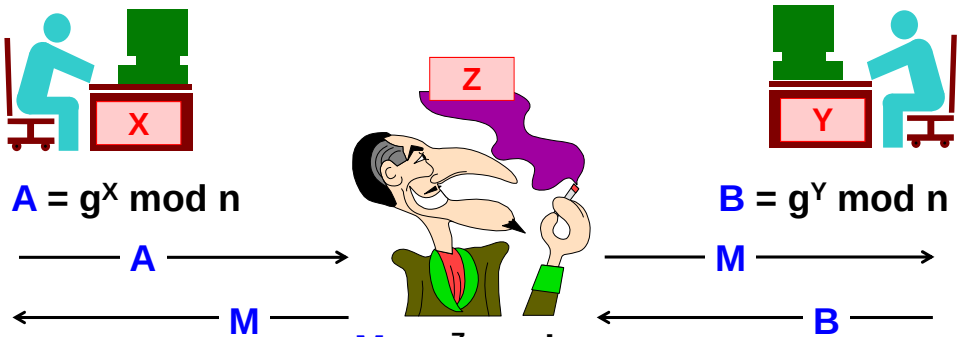
$K_A = B^X \bmod p$ $K_B = A^Y \bmod p$

$$K_A = K_B = g^{XY} \bmod p = K_{AB}$$

Diffie-Hellman

- **first public-key algorithm invented**
- **frequently used to agree on a secret key (*key agreement*)**
- **patented in the USA but the patent expired on 29 April 1997**
- **resistant to the sniffing attack**
- **if the attacker can manipulate the data then it is possible to make a *man-in-the-middle* attack; in this case it requires pre-authentication**
  - certificates for DH keys
  - authenticated DH = MQV (Menezes-Qu-Vanstone) patented by CertiCom

DH: man-in-the-middle attack



$A = g^X \bmod n$ $\qquad$ $B = g^Y \bmod n$

$\xrightarrow{\quad A \quad}$ $\qquad$ $\xrightarrow{\quad M \quad}$

$\xleftarrow{\quad M \quad}$ $\qquad$ $\xleftarrow{\quad B \quad}$

$M = g^Z \bmod n$

$K_A = M^X \bmod n$ $\qquad$ $K_B = M^Y \bmod n$

$$K_{AM} = g^{XZ} \bmod n \neq K_{BM} = g^{YZ} \bmod n$$

Elliptic curve cryptography

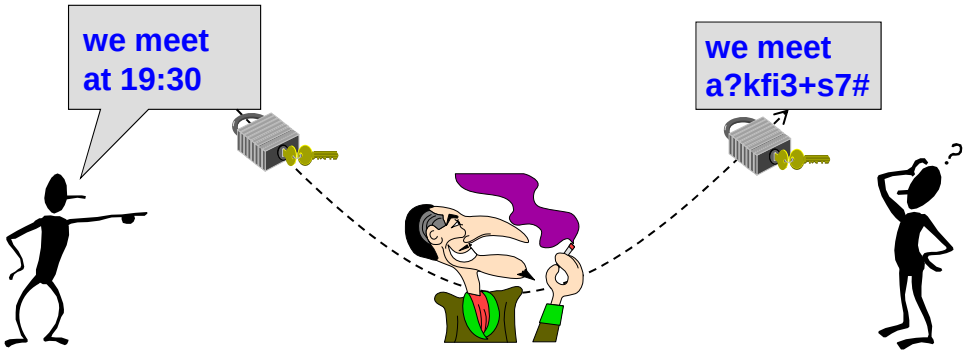
- **ECC (Elliptic Curve Cryptosystem)**
- **instead of working with modular arithmetic, the operations are done on the surface of a 2D curve (elliptic)**
- **problem of discrete logarithm on such a curve**
  - more complex than the same problem in modular arithmetic
  - possible to use shorter keys (about 1/10)
- **digital signature = ECDSA**
- **key agreement = ECDH**
- **authenticated key agreement = ECMQV (patented)**

Who's who in crypto

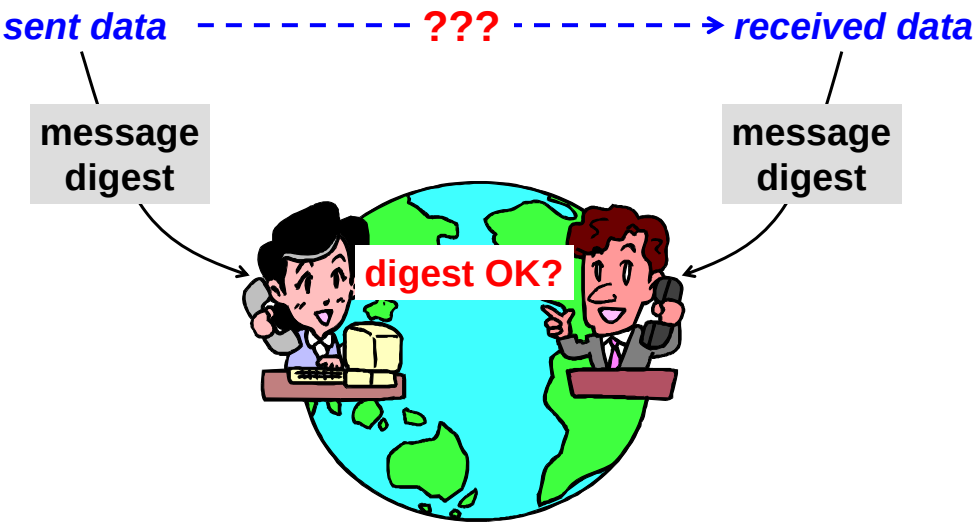
Adi Shamir **Ron Rivest** **Len Adleman** **Ralph Merkle** **Martin Hellman** **Whit Diffie**

Message integrity

- a person that intercepts an encrypted communication cannot read it ...
- ... but can modify it in an unpredictable way!



Digest

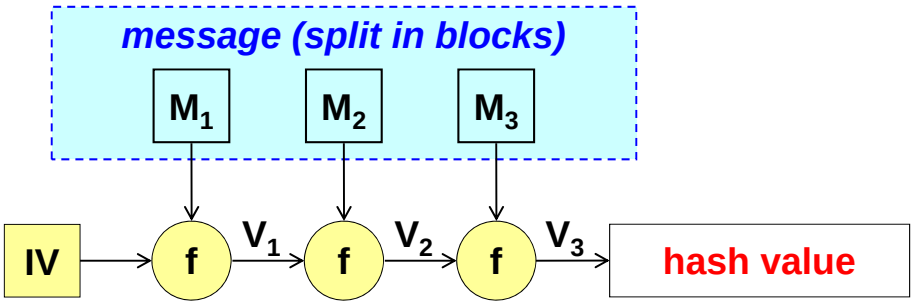


Message digest and hash functions

- **the message digest is a fixed-length "summary" of the message to be protected (of any length)**
- **it must be:**
  - fast to compute
  - impossible or very difficult to invert
  - difficult to create "collisions"
- **digest often used to avoid performing operations on the whole message, especially when the message is very large (e.g. because public-key cryptography is very slow)**
- **digest can be calculated in many ways, but usually via a (cryptographic) hash function**

Hash functions (dedicated)

- **usually:**
  - split the message M in N blocks $M_1 \ldots M_N$
  - iteratively apply a base function (f)
  - $V_k = f(V_{k-1}, M_k)$ with $V_0 = IV$ and $h = V_N$



Cryptographic hash algorithms

name	block	digest	definition	note
MD2	8 bit	128 bit	RFC-1319	obsolete
MD4	512 bit	128 bit	RFC-1320	obsolete
MD5	512 bit	128 bit	RFC-1321	semi-good
RIPEMD	512 bit	160 bit	ISO/IEC 10118-3	good
SHA-1	512 bit	160 bit	FIPS 180-1 RFC-3174	semi-good
SHA-224	512 bit	224 bit	FIPS 180-2 RFC-4634	optimal(?)
SHA-256	512 bit	256 bit	. . .	optimal(?)
SHA-384	512 bit	384 bit	. . .	optimal(?)
SHA-512	512 bit	512 bit	. . .	optimal(?)

SHA-1 broken

February 15, 2005

SHA-1 has been broken. Not a reduced-round version. Not a simplified version. The real thing.

The research team of Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu (mostly from Shandong University in China) have been quietly circulating a paper describing their results:

– collisions in the the full SHA-1 in 269 hash operations, much less than the brute-force attack of 2**80 operations based on the hash length.**

– collisions in SHA-0 in 239 operations.**

– collisions in 58-round SHA-1 in 233 operations.**

This attack builds on previous attacks on SHA-0 and SHA-1, and is a major, major cryptanalytic result. It pretty much puts a bullet into SHA-1 as a hash function for digital signatures (although it doesn't affect applications such as HMAC where collisions aren't important).

The paper isn't generally available yet. At this point I can't tell if the attack is real, but the paper looks good and this is a reputable research team.

`http://www.schneier.com/blog/archives/2005/02/sha1_broken.html`

Digest length

- **important to avoid *aliasing* (=collisions):**
  - md1 = h(m1)
  - md2 = h(m2)
  - if m1 ≠ m2 then we'd like to have md1 ≠ md2
- **if the algorithm is well designed and generates a digest of N bits, then the probability of aliasing is:**

$$P_A \propto 1 / 2^{Nbit}$$

- **thus, digests with many bits are required (because statistical events are involved)**

The birthday paradox

- **if there are at least 23 persons in the same room, then the probability that 2 of them were born in the same day is greater than 50 %; with 30 persons the probability is greater than 70%**
- **why? subtract from certainty (1) the probability that the 2nd, 3rd, 4th, … person was not born on the same day of any of the preceding ones**
  - $P(2) = 1 – 364/365$
  - $P(3) = 1 – 364/365 \cdot 363/365$
  - $P(N) = 1 – 364/365 \cdot 363/365 \cdot \ldots \cdot (365–N+1)/365$
  - $P(N) = 1 – [364 \cdot 363 \cdot 362 \cdot (365–N+1)] / 365^{N–1}$

The birthday attack

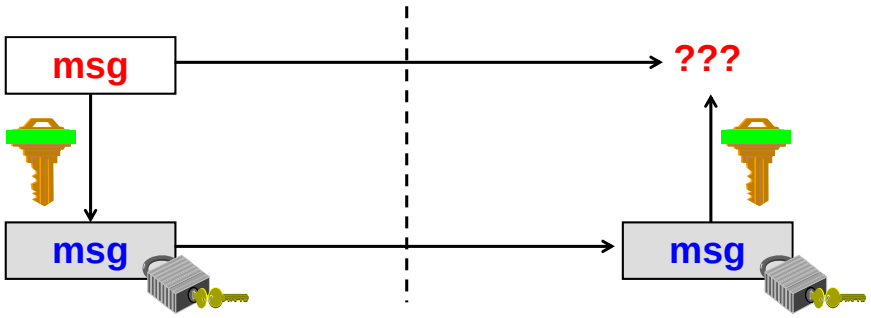
- a N-bits digest algorithm is not secure when more than 2**(N/2) digests are generated because the probability to have two messages with the same digest is P_A~50%
- SHA-256 and SHA-512 (SHA-354 is the cut off of SHA-512) have been designed for use respectively with AES-128 and AES-256
- note: SHA-1 (i.e. SHA-160) was designed to work with Skipjack-80
- usage example of SHA-256: key generation for AES-256 starting from a passphrase

MAC, MIC, MID

- to guarantee the integrity of messages, a code is added to the message:
  MIC (Message Integrity Code)
- often integrity is not useful without authentication, thus the code (ensuring both security properties) is named:
  MAC (Message Authentication Code)
- to avoid replay attacks, a unique identifier can be added to the message:
  MID (Message IDentifier)

Authentication by symmetric encryption

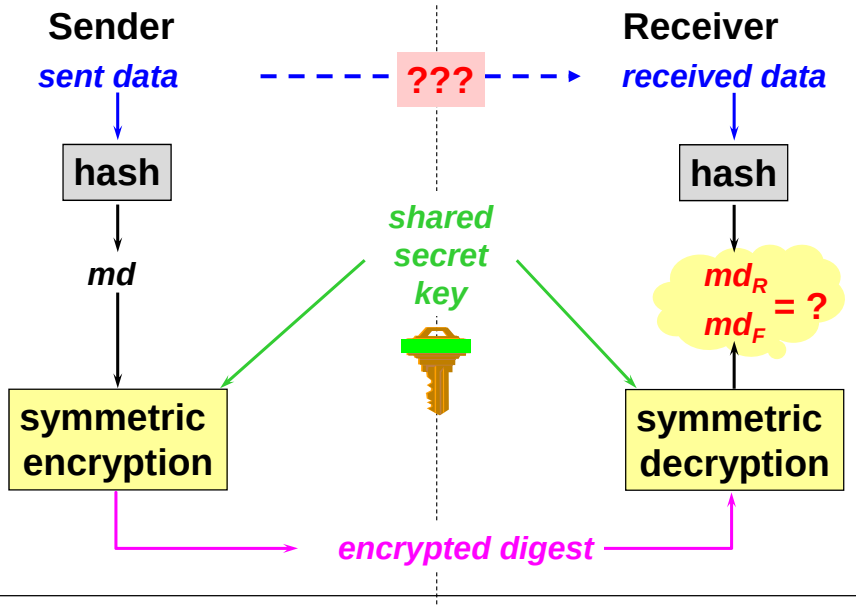
- send also an encrypted copy of data
- only who knows the (secret) key can compare the copy with the original data
- disadvantage: the same data are sent twice



Authentication by digest and symmetric encryption

- send also an (encrypted) digest of the data
- A → B : mex, { digest (mex) } S
- only who knows the (secret) key can compare the transmitted digest with the digest calculated on the received data
- disadvantage:
  - two operations (digest + encryption)
- advantage:
  - few additional data

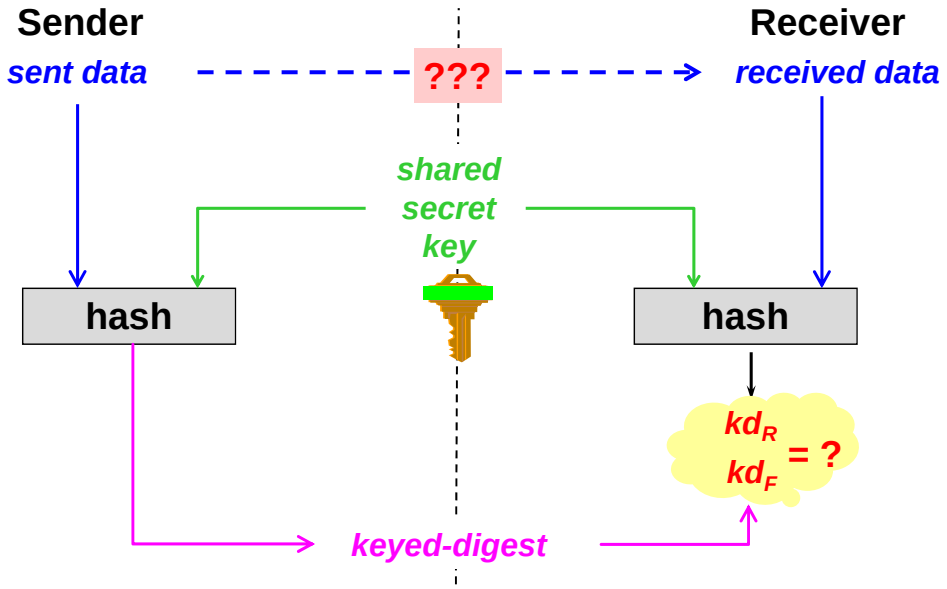
Digest + symmetric encryption



Authentication by means of keyed-digest

- **send also a digest calculated not only on data but also on a secret key**
- **A → B : mex, digest (mex, S)**
- **only who knows the key can compare the transmitted digest with the digest calculated on the received data**
- **advantages:**
  - only one operation (digest)
  - few additional data

Keyed-digest



Keyed-digest: possible mistakes

- **if kd = H (K || M) then I can change the message adding at its end one or more blocks:**
  - kd' = H (K || M || M') = f (kd, M')
- **if kd = H (M || K) then I can change the message adding before it a suitable block:**
  - kd = H (M' || M || K) choosing M' s.t. IV = f (IV, M')
- **protection:**
  - insert in the digested data also the length of M
  - define kd = H (K || M || K)
  - use a standard keyed-digest

Keyed-digest: some standards

- **RFC-1828 (historic) = keyed-md5**
  - md5(K || keyfill || data || K || MD5fill)
- **RFC-1852 (obsolete) = keyed-sha1**
  - sha1(K || keyfill || datagram || K || SHAfill)
- **RFC-2841 = keyed-sha1 (revised)**
  - sha1(K || keyfill || data || datafill || K || sha1fill)

Keyed-digest: HMAC

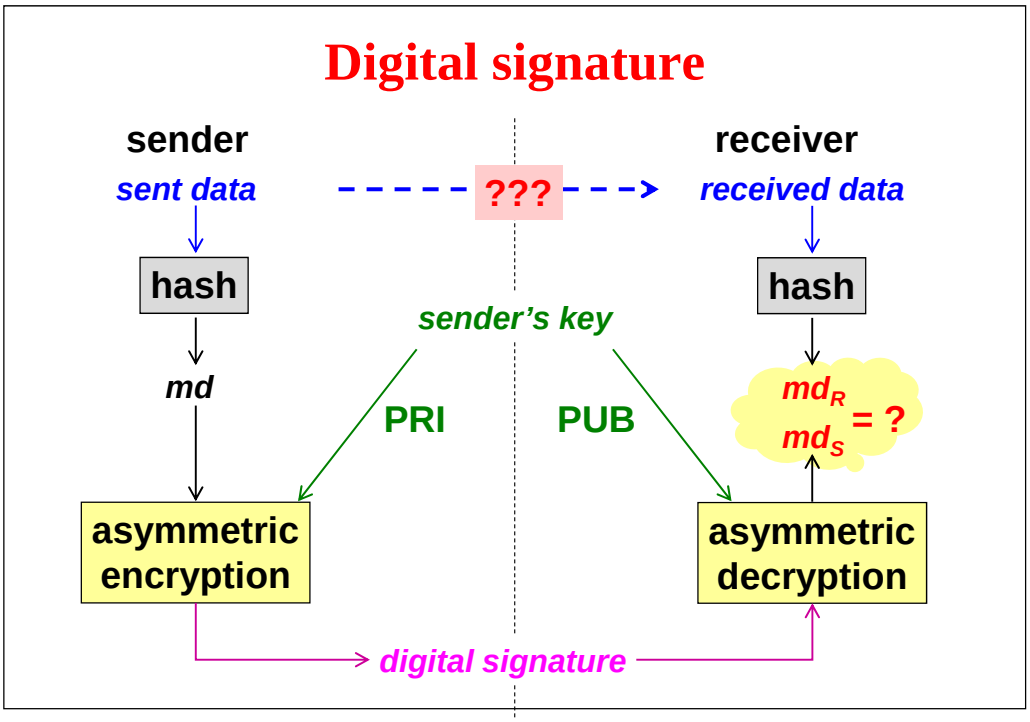
- **RFC-2104**
- **base hash function H: B byte block, L byte output**
- **definitions:**
  - ipad = 0x36 repeated B times
  - opad = 0x5C repeated B times
  - deprecated those keys s.t. | K | < L
  - if L < | K | < B, Kp = K padded with 0 until | Kp | = B
  - if | K | > B, Kp = H (K)
- **hmac = H(Kp ⊕ opad || H(Kp ⊕ ipad || data))**

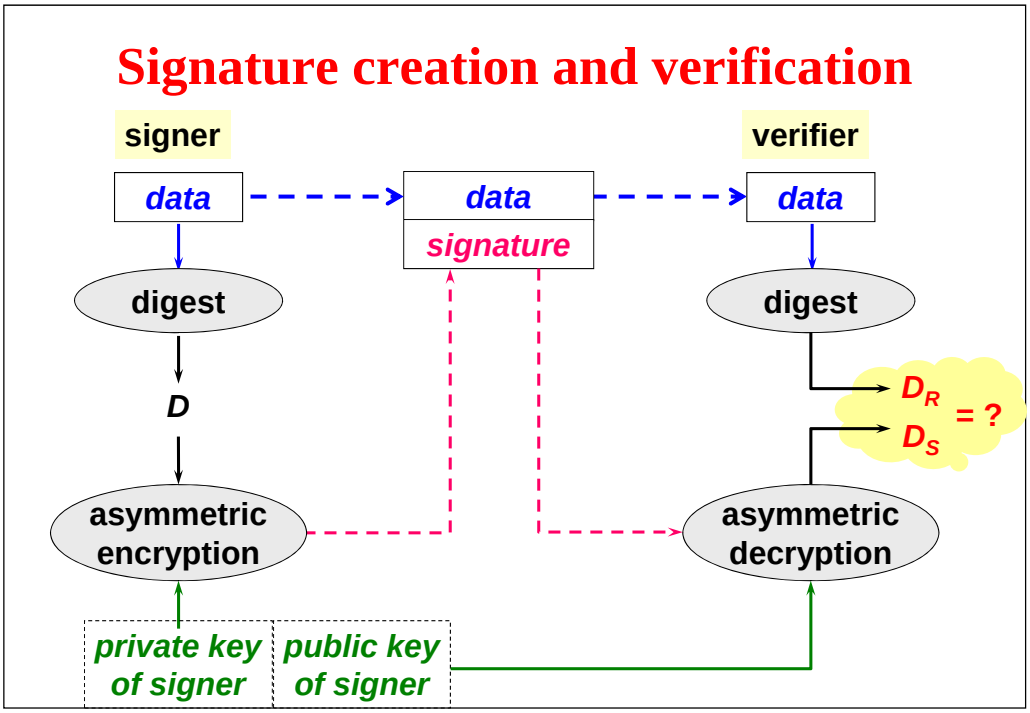
CBC-MAC

- **exploits a block-oriented symmetric encryption algorithm, in CBC mode with null IV, taking as MAC the last encrypted block**
- **message M split in N blocks $M_1 \ldots M_N$**
- **iterations:**
  - $V_0 = 0$
  - for (k=1…N) do $V_k = enc(K, M_k \oplus V_{k-1})$
- **cbc-mac = V_N**
- **DES-based CBC-MAC is the Data Authentication Algorithm (standard FIPS 113, ANSI X9.17)**

Authentication by digest and asymmetric cryptography

- **send also a digest (encrypted with the private key of the sender)**
- **those who know the public key can compare the transmitted digest with the digest calculated on the received data**
- **A → B : mex, { digest(mex) } priA**
- **DIGITAL SIGNATURE !!!**


Digital signature
sender
receiver
sent data
???
received data
hash
hash
sender's key
md
PRI
PUB
md_R
md_S = ?
asymmetric encryption
asymmetric decryption
digital signature


Signature creation and verification
signer
verifier
data
data
signature
data
digest
digest
D
D_R
D_S = ?
asymmetric encryption
asymmetric decryption
private key of signer
public key of signer

RSA signatures and hash functions

- **the hash function to be used in a RSA-based signature schema must be:**
  - resistant to collisions (obvious, even just to avoid generating accidentally the same signature)
  - difficult to invert (less obvious)
    - to create a fake signature of the key (E, N)
    - … choose S randomly
    - … compute $R = S^E \bmod N$
    - … find X such that $h(X) = R$, that is $X = h^{-1}(R)$
    - … we may state that S is the digital signature of X verifiable with the public key (E,N)

Authentication and integrity: analysis

- **by means of a shared secret:**
  - useful only for the receiver
  - cannot be used as a proof without disclosing the secret key
  - not useful for non repudiation
- **by means of asymmetric encryption:**
  - being slow it is applied to the digest only
  - can be used as a formal proof
  - can be used for non repudiation
  - = *digital signature*

Digital vs. handwritten signature

- digital signature = authentication + integrity
- handwritten signature = authentication
- thus the digital signature is better, because it is tightly bound to the data
- note: each user does not have a digital signature but a private key, which can be used to generate an infinite number of digital signatures (one for each different document)

Public key certificate

"A data structure used to securely bind a public key to some attributes"

- typically it binds a key to an identity ... but other associations are possible too (e.g. IP address)
- digitally signed by the issuer: the Certification Authority (CA)
- limited lifetime
- can be revoked on request both by the user and the issuer

Formats for public key certificates

- **X.509:**
  - v1, v2 (ISO)
  - v3 (ISO + IETF)
- **non X.509:**
  - PGP
  - SPKI (IETF)
- **PKCS-6:**
  - RSA, partly compatible with X.509
  - obsolete

Structure of a X.509 certificate

version	**2**
serial number	**1231**
signature algorithm	**RSA with MD5, 1024**
issuer	**C=IT, O=Polito, OU=CA**
validity	**1/1/97 - 31/12/97**
subject	**C=IT, O=Polito, CN=Antonio Lioy Email=lioy@polito.it**
subjectPublicKeyInfo	**RSA, 1024, xx...x**
CA digital signature	**yy...y**

PKI (Public-Key Infrastructure)

- **is the infrastructure ...**
- **technical and administrative ...**
- **put in place for the creation, distribution and revocation of public key certificates**

Certificate revocation

- **any certificate can be revoked before its expiration date:**
  - on request from the owner (subject)
  - autonomously by the creator (issuer)
- **when a signature is verified, the receiver must check that the certificate was valid at signature time**
- **this kind of check is the responsibility of the receiver (relying party, RP)**

Revocation mechanisms

- **CRL (Certificate Revocation List)**
  - list of revoked certificates
  - signed by the CA or by a delegated party
- **OCSP (On-line Certificate Status Protocol)**
  - response containing information about the certificate status
  - signed by the server

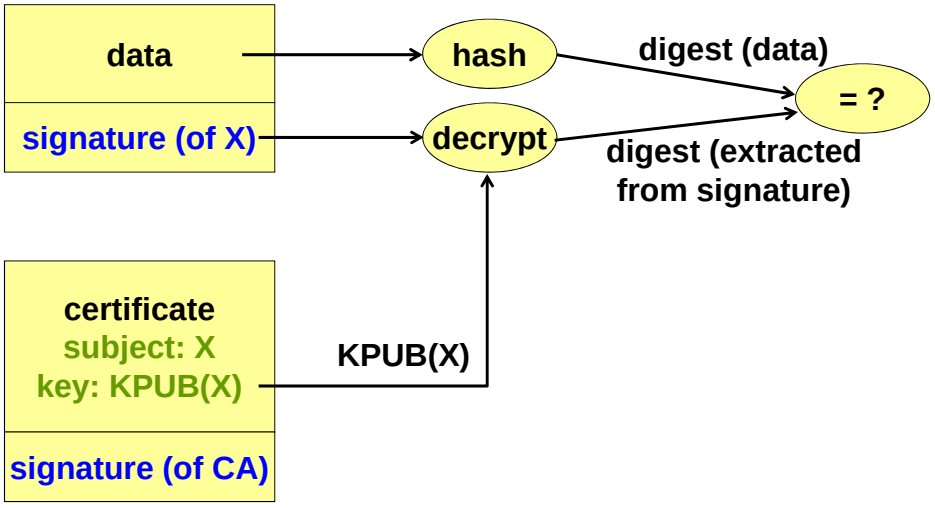
Structure of a X.509 CRL

version	**1**
signature algorithm	**RSA with MD5, 1024**
issuer	**C=IT, O=Polito, OU=CA**
thisUpdate	**15/10/2000 17:30:00**
userCertificate revocationDate	**1496 13/10/2000 15:56:00**
userCertificate revocationDate	**1574 4/6/1999 23:58:00**
CA digital signature	**yy...y**

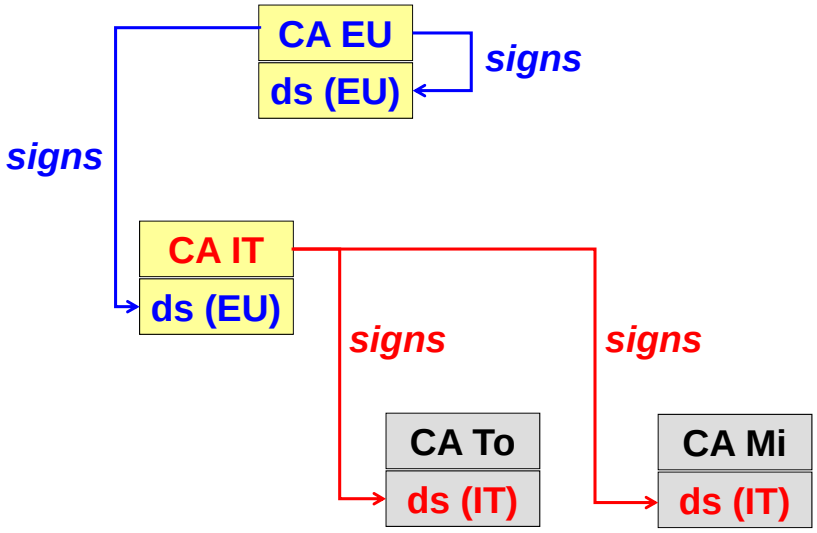
Verification of a signature / certificate

- how to verify that a public-key certificate (signed by CA1) is authentic?
- … the public-key certificate of CA1 is required (which will be signed by CA2)
- how to verify the last one ?
- … the public-key certificate of CA2 is required (which will be signed by CA3)
- … and so on …
- it becomes thus necessary to have an infrastructure (hierarchical?) for certification and distribution of public-key certificates

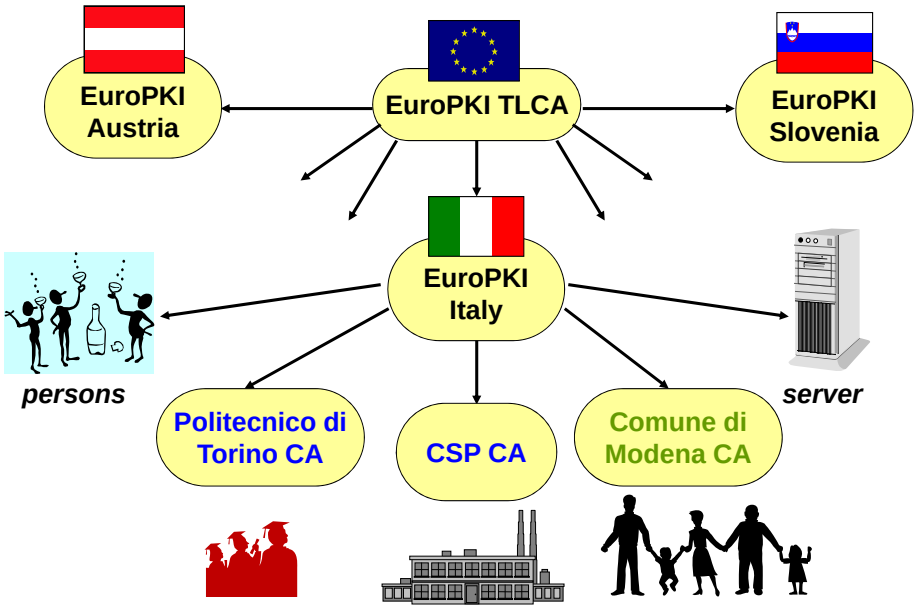
Verification of a signature / certificate



Certification hierarchy



EuroPKI



Performance

- **cryptographic performance does not depend on RAM but on CPU (architecture and instruction set) and cache size**
- **performance is not a problem on clients (except when they are overloaded by local applications)**
- **performance can become a problem on the servers and/or on the network nodes (e.g. router):**
  - use cryptographic accelerators
  - special-purpose accelerators (e.g. SSL, IPsec) or generic ones

Performance (P4 @ 1.7 GHz)

	[64 B/packet]	[1024 B/packet]
hmac(md5)	31.5 MB/s	152.1 MB/s
des cbc	28.7 MB/s	28.9 MB/s
des ede3	10.8 MB/s	10.9 MB/s
aes-128	38.0 MB/s	37.8 MB/s
rc4-128	61.2 MB/s	62.0 MB/s

rsa 1024	133.7 signs/s	2472.1 verifies/s

Performance (P3 @ 800 MHz)

	[64 B/packet]	[1024 B/packet]
hmac(md5)	19.2 MB/s	83.6 MB/s
des cbc	14.4 MB/s	14.5 MB/s
des ede3	5.2 MB/s	5.2 MB/s
aes-128	15.6 MB/s	15.9 MB/s
rc4-128	80.9 MB/s	86.4 MB/s

rsa 1024	94.8 signs/s	1682.0 verifies/s

NSA suite B

- **for COTS products that treat SBU (Sensitive But Unclassified) and Classified information**
- **contains the following algorithms:**
  - (symmetric encryption) AES-128 e AES-256
  - (hash) SHA-256 e SHA-384
  - (key agreement) ECDH e ECMQV
  - (digital signature) ECDSA
- **for information up to Secret level:**
  - AES-128 + SHA-256 + EC-256
- **for information at Top Secret level:**
  - AES-256 + SHA-384 + EC-384

Length of keys and digest (NIST, 2007)

- equivalence defined in NIST SP800-57
- FFC = Finite Field Cryptography (e.g. DSA, D-H)
- IFC = Integer Factorization Cryptography (e.g. RSA)

symm.	FFC	IFC	ECC	hash	years
80	1024	1024	160	160	< 2010
112	2048	2048	224	224	< 2030
128	3072	3072	256	256	> 2030
192	7680	7680	384	384	> 2030
256	15360	15360	512	512	> 2030

Length of keys and digest (ECRYPT, 2008)

symm.	80	96	112	128	256
hash	160	192	224	256	512
asymm.	1248	1776	2432	3248	15424
ECC	160	192	224	256	512
	very short term	legacy	medium term	long term (2008-'38)	foreseaable future (quantum computers)

- www.keylength.com

Why don't we buy everything from USA?

- **export of cryptographic material is subject to the same restrictions as nuclear material (!)**
- **… unless the protection level is very low :**
  - symmetric key restricted to 40 bits (2^{40} trials = few CPU hours)
  - asymmetric key restricted to 512 bits
- **example: Netscape, Internet Explorer, … (export version)**

Key escrow: a novelty?

- **December 1996: the USA government authorizes the export of semi-robust (56 bits) cryptographic products if they incorporate key-escrow functions**
- ***does not*** **apply to internal USA products**
- **key escrow = possibility to recover a key even without the consent of the owner**
- **example: Lotus Notes 4.x was using 64 bits symmetric keys, but 24 bits of them were encrypted with the NSA public-key**
- **problem: who decides when it is necessary to recover a key?**

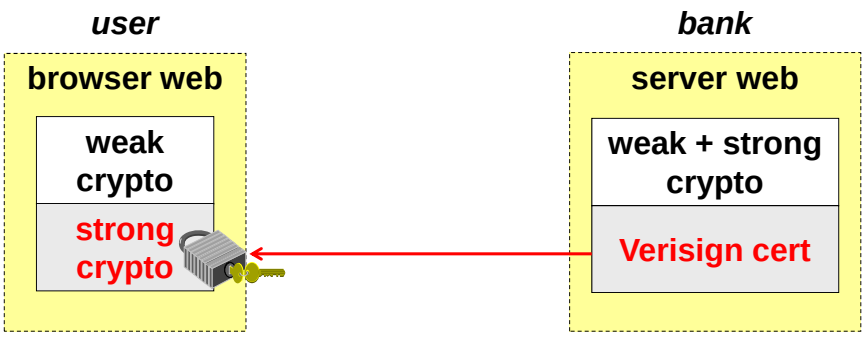
The many editions of Notes

Aside from encryption process time, U.S. government export laws limit encryption key length. These laws are the driving force behind the three major editions of Notes: North American, International, and French. Despite the different names, the product functionality is exactly the same. The difference, however, lies in the length of the keys used for encryption.
The North American edition uses encryption keys that are 64-bits long. The U.S. Government, for reasons of national security, limits the length of encryption keys for export to 40 bits. To comply with these restrictions, we have the International edition. When we generate a 64-bit key for the International edition, the top 24 bits are encrypted using the U.S. Government's public key and stored in what is called the Workfactor Reduction Field (WRF). Splitting the key in this manner results in a key that's 40 bits for the U.S. Government and 64 bits for everyone else. This approach maintains a high level of security worldwide without violating the export laws of the U.S. Government.
Most countries are content with the way the International edition complies with U.S. encryption key export laws. The government of France, however, found the International edition unacceptable. To comply with French law, we created the French edition, which uses a plain 40-bit encryption key and can therefore be "broken" by attackers willing to apply considerable computing power (presumably, including the French government).

Changes in the USA cryptographic export regulations

- **June 1997:**
  - permission to export secure web client and server web only if used by foreign branches of USA companies or in financial environment (transactions)
  - to verify the real use, special certificates issued by Verisign must to be used
- **September 1998:**
  - permission extended to insurance and health institutions
  - no permission for keys up to 56 bits

Step-up (gated) cryptography



Key recovery: another novelty?

- **export from USA of products with strong cryptography (e.g. 128 bits symmetric keys) is allowed if:**
  - they incorporate key-recovery functions
  - with a recovery centre authorized by the USA government
- **the symmetric keys used by the users are encrypted with the public key of the recovery centre**
- **in this way it becomes a political problem but key-recovery is a real problem**

USA cryptographic export regulations (December 1999)

- **symmetric algorithms with 56 bits keys**
- **asymmetric algorithms with keys:**
  - 1024 bits if used only for authentication
  - 512 bits if used also for key exchange
- **not all products conformed to these rules:**
  - Netscape has 56 bits keys from version 4.6
  - IE 5.0 has 56 bits keys only in the Win2k version
  - both generate 512 bits asymmetric keys

New USA export regulation

- **January 2000**
- **permission to export ...**
  - off-the-shelf products ...
  - that passed a “one-time review”
- **or**
  - products whose source code is freely available in Internet
- **upgrades available for the main commercial products**
  - doubts about the existence of back-doors

Bibliography
(http://security.polito.it/books.html)

- B.Schneier: "Applied cryptography"
- W.Stallings:
  "Cryptography and network security" (3rd ed.)
- S.Garfinkel, G.Spafford:
  "Practical Unix and Internet security"
- W.R.Cheswick, S.M.Bellovin:
  "Firewalls and Internet security" (2nd ed.)
- W.Ford, M.S.Baum:
  "Secure electronic commerce"
- C.P.Pfleeger, S.Pfleeger:
  "Security in computing" (3rd ed.)

Bibliografia (in Italiano)

- W.Stallings
  "Sicurezza delle reti - applicazioni e standard"
  Addison-Wesley Italia, 2004
- C.Pfleeger, S.Pfleeger
  "Sicurezza in informatica"
  Pearson Education Italia, 2004
- Fugini, Maio, Plebani
  "Sicurezza dei sistemi informativi"
  Apogeo, 2001
- S.Singh
  "Codici e segreti"
  BUR saggi, 2001