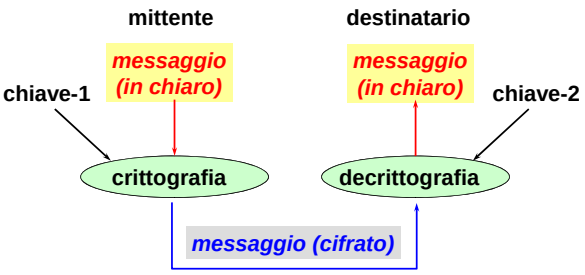
Concetti base di sicurezza informatica

Antonio Lioy
< lioy @ polito.it >

Politecnico di Torino
Dip. Automatica e Informatica

Crittografia



Terminologia

- **messaggio in chiaro:**
  - plaintext o cleartext
  - noi lo indicheremo con P
- **messaggio cifrato:**
  - ciphertext
  - noi lo indicheremo con C
- **cifrare / decifrare:**
  - to encrypt / decrypt
  - to cipher / decipher
  - per alcuni popoli usare “crypt” in questo senso è offensivo (culto dei morti); preferire quindi “cipher”

Forza della crittografia (principio di Kerchoffs)

- **se le chiavi:**
  - sono tenute segrete
  - sono gestite solo da sistemi fidati
  - sono di lunghezza adeguata
- **allora ...**
- **... non ha alcuna importanza che gli algoritmi di crittografia e decrittografia siano tenuti segreti**
- **... anzi è bene che siano pubblici affinché siano studiati ampiamente e siano rivelate eventuali debolezze**

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

Security trough obscurity (STO)

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

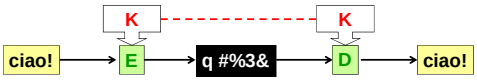
Crittografia a chiave segreta

- **chiave-1 = chiave-2**
- **algoritmi simmetrici**
- **basso carico di elaborazione**
- **usata per crittografia dei dati**
- **principali algoritmi:**
  - DES, triplo-DES
  - IDEA
  - RC2, RC5
  - RC4
  - AES

© *Antonio Lioy - Politecnico di Torino (1995-2012)*

Crittografia simmetrica

- chiave unica
- chiave comune a mittente e destinatario
- C = enc (K , P) oppure C = { P } K
- P = dec (K , C) = enc^{-1} (K , C)



Algoritmi simmetrici

nome	*chiave*	*blocco*	*note*
DES	56 bit	64 bit	obsoleto
3-DES	112 bit	64 bit	resistenza 56-112 bit
3-DES	168 bit	64 bit	resistenza 112 bit
IDEA	128 bit	64 bit	
RC2	8-1024 bit	64 bit	solitamente K=64 bit
RC4	variabile	stream	segreto
RC5	0-2048 bit	1-256 bit	ottimale se B=2W
AES	128-256 bit	128 bit	alias Rjindael

La funzione EX-OR (XOR)

- operatore di “confusione” ideale
- se l’input è casuale (probabilità 0 : 1 = 50 : 50%) allora anche l’output sarà casuale allo stesso modo
- operazione primitiva disponibile su tutte le CPU
- tavola di verità:

⊕	0	1
0	0	1
1	1	0

DES

- Data Encryption Standard
- standard FIPS 46/2
- modalità di applicazione standard FIPS 81
- chiave a 56 bit (+ 8 bit di parità = 64 bit)
- blocco dati da 64 bit
- pensato per essere efficiente in hardware perché richiede:
  - XOR
  - shift
  - permutazioni (!)

Triplo DES (3DES, TDES)

- applicazione ripetuta di DES
- usa due o tre chiavi a 56 bit
- solitamente usato in modalità EDE (per compatibilità con DES se K1 = K2 = K3)
- 3DES con 2 chiavi (Keq=56 bit se si hanno 2^{59}B di memoria, altrimenti Keq=112 bit)
  $C'=enc(K_1,P)$ $C''=dec(K_2,C')$ $C=enc(K_1,C'')$
- 3DES con 3 chiavi (Keq= 112 bit)
  $C'=enc(K_1,P)$ $C''=dec(K_2,C')$ $C=enc(K_3,C'')$
- standard FIPS 46/3 e ANSI X9.52

Doppio DES ?

- l’applicazione doppia di algoritmi di cifratura è soggetta ad un attacco di tipo known-plaintext chiamato meet-in-the-middle che permette di decifrare i dati con al più 2^{N+1} tentativi (per chiavi da N bit)
- quindi solitamente non si usa mai la versione doppia degli algoritmi di cifratura
- nota: se l’algoritmo simmetrico di base fosse un *gruppo* allora esisterebbe K_3 tale che
  enc (K_2 , enc (K_1 , P)) = enc (K_3 , P)

Attacco meet-in-the-middle

- **ipotesi:**
  - chiavi da N bit
  - sono noti P e C tali che C = enc (K_2, enc (K_1, P))
- **nota:**
  - esiste M tale che M = enc(K_1,P) e C = enc(K_2,M)
- **azioni:**
  - calcolo 2^N valori X_i = enc (K_i, P)
  - calcolo 2^N valori Y_j = dec (K_j, C)
  - cerco quei valori K_i e K_j tali che $X_i = Y_j$
  - "falsi positivi" eliminabili se ho più coppie P,C

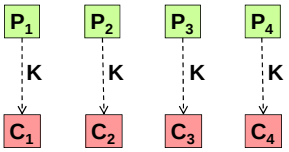
Applicazione degli algoritmi a blocchi

Come si applica un algoritmo a blocchi a dati in quantità diversa da quella del blocco base?

- **per cifrare dati in quantità superiore:**
  - ECB (Electronic Code Book)
  - CBC (Cipher Block Chaining)
- **per cifrare dati in quantità inferiore:**
  - padding
  - CFB (Cipher FeedBack), OFB (Output FeedBack)
  - CTR (Counter mode)

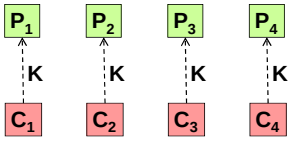
ECB (Electronic Code Book)

- **formula per il blocco i-esimo:**
  C_i = enc (K, P_i)
- **fortemente sconsigliato (su messaggi lunghi):**
  - possibile scambiare due blocchi qualunque
  - si presta ad attacchi *known-plaintext*



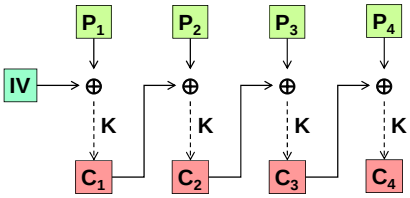
ECB - decifratura

- **formula per il blocco i-esimo:**
  P_i = enc^{-1} (K, C_i)
- **un errore in trasmissione provoca errore nella decifratura di un solo blocco**



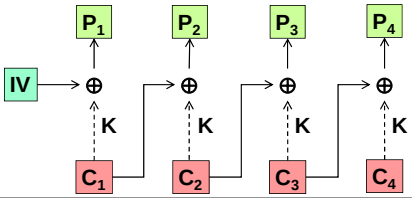
CBC (Cipher Block Chaining)

- **formula per il blocco i-esimo:**
  C_i = enc (K, $P_i \oplus C_{i-1}$)
- **richiede C_0 = IV (Initialization Vector)**



CBC - decifratura

- **formula per il blocco i-esimo:**
  P_i = enc^{-1} (K , C_i) $\oplus$ C_{i-1}
- **C_0 (ossia IV) noto al ricevente (in chiaro o cifrato)**
- **un errore in trasmissione provoca errore nella decifratura di due blocchi**



Padding (allineamento)

- **dimensione del blocco algoritmico B**
- **quantità di dati da elaborare D (non multiplo di B)**
- **aggiungo bit per far diventare i dati un multiplo B**

- **problemi:**
  - trasmetto più dati (NxB) del minimo necessario (D)
  - valore dei bit di padding?

Metodi di padding

- **(se lunghezza è nota o ricavabile - es. stringa C) aggiungere byte nulli**
  - … 0x00 0x00 0x00
- **(DES originale) un bit a 1 seguito da tanti 0**
  - … 1000000
- **un byte con valore 128 seguito da byte nulli**
  - … 0x80 0x00 0x00
- **ultimo byte pari alla lunghezza del padding**
  - … 0x?? 0x?? 0x03
  - valore degli altri byte?

Padding con lunghezza esplicita (N)

- **(Schneier) byte nulli:**
  - es. … 0x00 0x00 0x03
- **(SSL/TLS) byte con valore N:**
  - es. … 0x03 0x03 0x03
- **(SSH2) byte random:**
  - es. … 0x05 0xF2 0x03
- **(IPsec/ESP) numero progressivo:**
  - es. … 0x01 0x02 0x03
- **byte con valore N-1:**
  - es. … 0x02 0x02 0x02

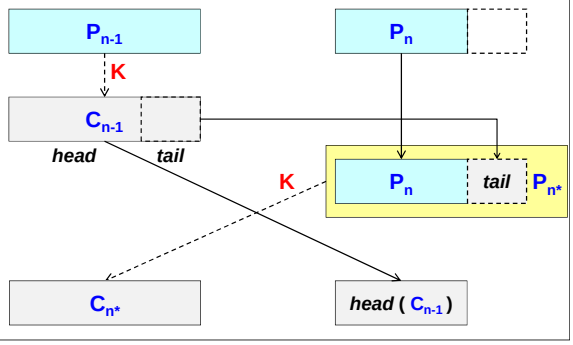
Padding – alcune note

- **tipicamente il padding si applica a dati molto grandi, sull'ultimo frammento che avanza dopo la suddivisione in blocchi (es. per ECB o CBC)**
- **per |D| < |B| si preferiscono tecniche ad-hoc (CFB, OFB, CTR, …)**
- **anche se il plaintext è un numero esatto di blocchi bisogna comunque inserire il padding per evitare errori di interpretazione dell'ultimo blocco**
- **il padding di SSH2 implica che dati uguali vengono cifrati in modo diverso**
- **la scelta del tipo di padding per un certo algoritmo determina il tipo di alcuni attacchi possibili**

Ciphertext stealing (CTS)

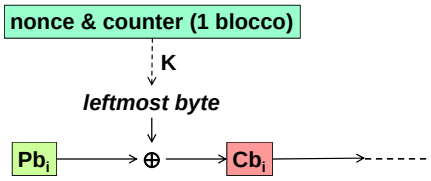
- **per usare algoritmi a blocchi senza fare padding**
  - ultimo blocco (parziale) riempito con byte del penultimo blocco
  - byte rimossi dal penultimo blocco (diventa parziale)
  - scambio di posto tra ultimo e penultimo blocco dopo la cifratura
- **utile quando non si può/vuole aumentare la dimensione dei dati dopo la cifratura**
- **aumenta un po' il tempo di elaborazione**

CTS – esempio con ECB (cifratura)



CTR (Counter mode)

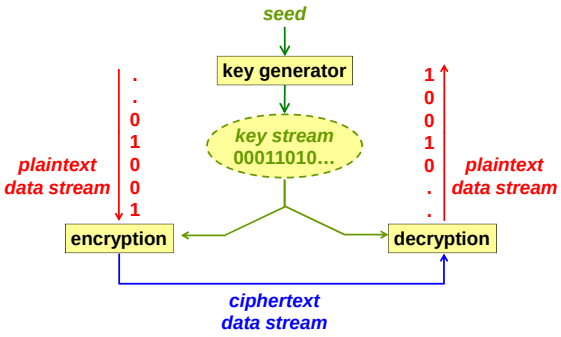
- permette di cifrare N-bit alla volta (gruppo)
- random access al ciphertext
- richiede un nonce ed un contatore (concatenati, sommati, XOR, …)
- un errore in trasmissione ~ errore solo in un gruppo



Algoritmi simmetrici di tipo stream

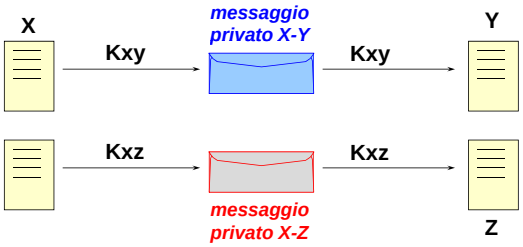
- operano su un flusso di dati senza richiederne la divisione in blocchi, tipicamente su un bit o un byte alla volta
- algoritmo ideale:
  - one-time pad (richiede chiave lunga tanto quanto il messaggio da proteggere!)
- algoritmi reali:
  - usano generatori pseudo-casuali di chiavi, sincronizzati tra mittente e ricevente
  - esempi: RC4 e SEAL

Algoritmi di tipo stream



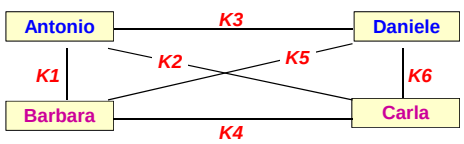
Crittografia simmetrica

- chiave singola e segreta
- una chiave per ogni coppia / gruppo di utenti



Distribuzione chiavi per crittografia simmetrica

- per una comunicazione privata completa tra N persone occorrono N x (N-1) / 2 chiavi:
  - distribuzione OOB (Out-Of-Band)
  - distribuzione tramite algoritmi per scambio chiavi

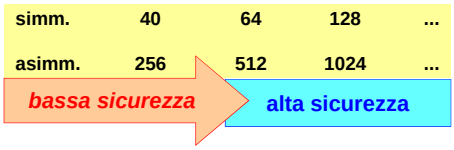


Lunghezza delle chiavi segrete

- se:
  - l'algoritmo di crittografia è stato ben progettato
  - le chiavi - lunghe Nbit - sono tenute segrete
- … allora l'unico attacco possibile è l'attacco esaustivo (o brute force) che richiede un numero di tentativi pari a

$$T = 2^{Nbit}$$

Lunghezza delle chiavi crittografiche



Sfide DES

- **2^{56} = 72.057.594 miliardi di chiavi possibili**
- **DES challenge I**
  - inizio=18-feb-1997, fine=17-giu-1997
  - 17.731.000 miliardi di chiavi provate (25%)
  - circa 15.000 computer in rete
- **DES challenge II**
  - inizio=13-gen-98, fine=23-feb-98
  - 63.686.000 miliardi di chiavi provate (87%)
  - circa 20.000 computer in rete

La fine (?) del DES

- **DES challenge III**
  - inizio=13-lug-98, fine=15-lug-98
  - 17.903.000 miliardi di chiavi provate (25%)
  - 1 sistema special-purpose (DEEP CRACK) sviluppato dalla EFF col costo di 250.000 $
- **è quindi possibile costruire una macchina che decifri un generico messaggio DES, ma:**
  - bisogna conoscere il tipo di dati (es. ASCII)
  - la macchina non riesce a decifrare messaggi 3DES
  - il DES non è intrinsecamente debole, ha solo una chiave corta!

Sempre più in basso

- **DES challenge IV**
  - inizio=18-gen-1999, fine=dopo 22h 15m
  - 16.017.000 miliardi di chiavi provate (22%)
  - DEEP CRACK più alcune migliaia di workstation
  - potenza di picco: 250 Gkey/s
  - potenza media: 199 Gkey/s

Cosa c'è dopo il DES?

- **IETF cambia tutti gli RFC sconsigliando l'uso del DES e suggerendo l'uso del triplo DES**
- **RFC-4772 (security implications of using DES)**
- **banca tedesca condannata per una truffa fatta tramite un sistema basato su DES**
- **il 15-gen-1999 il FIPS ha ritirato il DES (46/2) e l'ha rimpiazzato col 3DES (46/3)**
- **gara del governo USA per nuovo algoritmo simmetrico:**
  - AES (Advanced Encryption Standard)
  - chiave da almeno 256 bit
  - blocchi da almeno 128 bit

AES (Advanced Encryption Standard)

- **15 candidati**
- **5 finalisti (9 agosto 1999):**
  - MARS (IBM)
  - RC6 (RSA, ossia Ron Rivest)
  - Rijndael (Joan Daemen, Vincent Rijmen)
  - Serpent (Ross Anderson, Eli Biham, Lars Knudsen)
  - Twofish (Bruce Schneier ed altri)
- **informazioni circa il processo di selezione:**
  http://www.nist.gov/aes

AES = RIJNDAEL

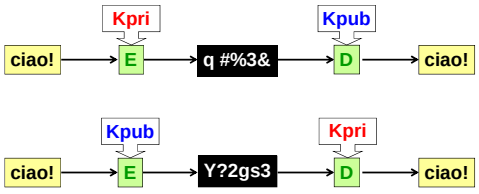
- **2 ottobre 2000**
- **RIJNDAEL scelto come vincitore**
- **pubblicato nel novembre 2001 come FIPS-197**
- **in corso di adozione (perché gli algoritmi sono come il vino: i migliori sono quelli invecchiati per alcuni anni …)**

Crittografia a chiave pubblica

- **chiave-1 ≠ chiave-2**
- **algoritmi asimmetrici**
- **coppie di chiavi (*pubblica* e *privata*)**
- **chiavi con funzionalità reciproca**
- **alto carico di elaborazione**
- **usato per distribuire chiavi segrete e per la firma elettronica (con hashing)**
- **principali algoritmi:**
  - Diffie-Hellman, RSA, DSA, El Gamal, …

Crittografia asimmetrica

- **chiavi generate a coppie:**
  ***chiave privata* (Kpri) + *chiave pubblica* (Kpub)**
- **chiavi con funzionalità reciproca: dati cifrati con una chiave sono decifrabili solo con l'altra**

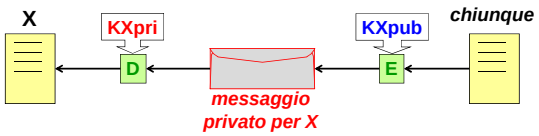


Firma digitale

- **firma digitale ~ cifratura asimmetrica dei dati con la chiave privata dell'autore**
- **solitamente non si cifrano direttamente i dati ma un loro riassunto (*digest*)**
- **fornisce autenticazione (e integrità) dei dati**

Riservatezza senza segreti condivisi

- **possibile generare un messaggio segreto per uno specifico destinatario conoscendone solo la chiave pubblica**



Algoritmi a chiave pubblica

- **RSA (Rivest - Shamir - Adleman)**
  - prodotto di numeri primi, fattorizzazione del risultato
  - segretezza e firma digitale
  - brevettato - solo in USA - da RSA; scaduto il 20-set-2000
- **DSA (Digital Signature Algorithm)**
  - elevamento a potenza, logaritmo del risultato
  - solo per firma digitale
    - per cifratura si usa El-Gamal
  - standard NIST per DSS (FIPS-186)

Attacco DoS su RSA

- **solitamente tutte le chiavi pubbliche hanno esponente pari a 3 o al numero di Fermat 65537 (0x00010001)**
  - è molto facile fare l'elevamento a potenza perché contiene solo due bit a uno e si ottiene
    - velocità in cifratura
    - velocità nella verifica della firma
  - algoritmi ottimizzati per questo caso speciale
- **attacco: fornire una firma realizzata con una chiave con esponente con molti bit a uno, così da provocare un grosso sovraccarico di calcolo**

Lunghezza delle chiavi pubbliche

- **256 bit sono attaccabili in alcune settimane**
- **512 bit sono attaccabili in alcuni mesi**
- **1024 bit offrono una sicurezza ragionevole per vari secoli**
- **provato tramite sfide RSA:**

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

Sfide RSA

- **sfide risolte (vecchio stile):**
  - 10-apr-1996, RSA-130, 1000 MIPS-years
  - 2-feb-1999, RSA-140 (465 bit), 2000 MIPS-years
  - 22-ago-1999, RSA-155 (512 bit), 8000 MIPS-years
  - 9-mag-2005, RSA-200 (663 bit), ~75 anni Opteron 2.2 GHz
- **sfide risolte (nuovo stile):**
  - 3-dic-2003, RSA-576 (174 cifre decimali)
  - 2-nov-2005, RSA-640 (193 cifre decimali)
  - 12-dic-2009, RSA-768 (232 cifre decimali)

Twinkle (!?)

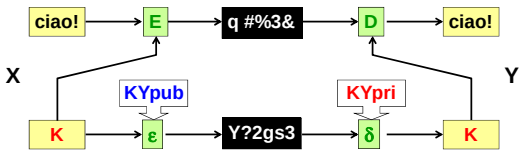
An Analysis of Shamir's Factoring Device
Robert D. Silverman
RSA Laboratories
May 3, 1999
At a Eurocrypt rump session, Professor Adi Shamir of the Weizmann Institute announced the design for an unusual piece of hardware. This hardware, called "TWINKLE" (which stands for The Weizmann INstitute Key Locating 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.

Distribuzione delle chiavi per crittografia asimmetrica

- **chiave privata mai divulgata!**
- **chiave pubblica distribuita il più ampiamente possibile**
- **problema: *chi garantisce la corrispondenza tra chiave pubblica ed identità della persona?***
- **soluzione #1: scambio di chiavi OOB (es. key party)**
- **soluzione #2: distribuzione della chiave pubblica all'interno di un certificato a chiave pubblica (= certificato d'identità digitale)**
  - formato del certificato?
  - fiducia nell'emettitore del certificato?

Scambio chiave segreta mediante algoritmi asimmetrici

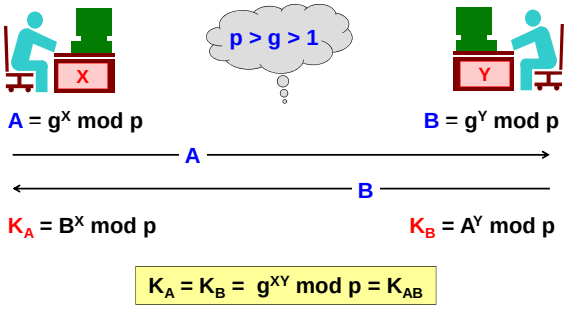
- **la riservatezza senza segreti condivisi viene spesso usata per comunicare la chiave crittografica scelta per un algoritmo simmetrico**



Diffie-Hellman

- A e B scelgono due interi grandi g e p (primo) tali che:
  $1 < g < p$
- A, scelto un numero a caso x, calcola:
  $X = g^x \bmod n$
- B, scelto un numero a caso y, calcola:
  $Y = g^y \bmod n$
- A e B si scambiano (pubblicano) X e Y
- A calcola $K = Y^x \bmod n$
- B calcola $K' = X^y \bmod n$
- ma $K = K' = g^{xy} \bmod n$

Diffie-Hellman (DH)

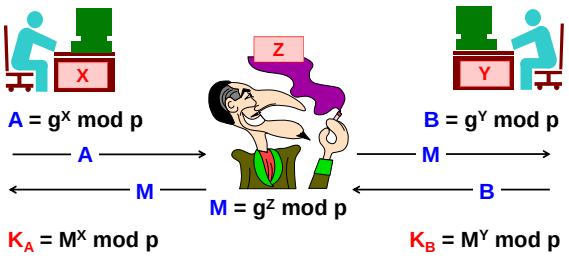


$K_A = K_B = g^{XY} \bmod p = K_{AB}$

Diffie-Hellman

- primo algoritmo a chiave pubblica inventato
- solitamente usato per concordare su una chiave segreta (*key agreement*)
- brevettato negli USA ma il brevetto è scaduto il 29 aprile 1997
- resistente allo sniffing
- se l'attaccante può manipolare i dati allora è possibile un attacco ***man-in-the-middle***; in questo caso richiede pre-autenticazione
  - certificati per chiavi DH
  - authenticated DH = MQV (Menezes-Qu-Vanstone) brevettato da CertiCom

DH: attacco man-in-the-middle



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

Crittografia a curve ellittiche

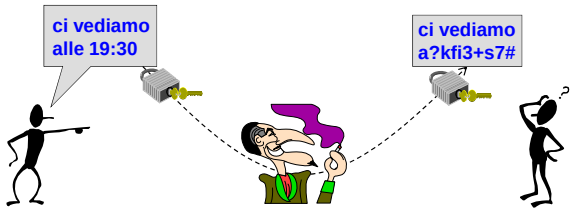
- ECC (Elliptic Curve Cryptosystem)
- invece di lavorare con l'aritmetica modulare, si lavora sulla superficie di una curva ellittica 2D
- problema del logaritmo discreto su tale curva
  - più complesso dello stesso problema in aritmetica modulare
  - possibile usare chiavi più corte (circa 1/10)
- firma digitale = ECDSA
- key agreement = ECDH
- authenticated key agreement = ECMQV (brevettato)
- cifratura asimmetrica = ECIES

Who's who in crypto

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

Integrità dei messaggi

- una persona che intercetti una comunicazione cifrata non può leggerla ...
- ... ma può modificarla in modo imprevedibile!



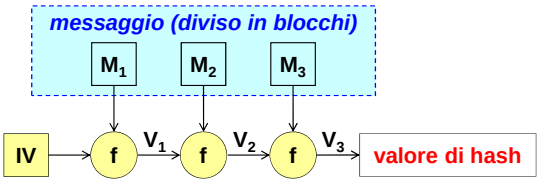
Digest

Message digest e funzioni di hash

- è un "riassunto" a lunghezza fissa del messaggio da proteggere (che può avere qualsiasi lunghezza)
- deve essere:
  - veloce da calcolare
  - impossibile (o molto difficile) da invertire
  - difficile generare "collisioni"
- spesso usato per non lavorare sul messaggio completo quando è molto grosso (es. perché la crittografia a chiave pubblica è molto lenta)
- digest calcolabili in molti modi, ma solitamente tramite una funzione di hash (crittografica)

Funzioni di hash (dedicate)

- solitamente:
  - dividono il messaggio M in N blocchi $M_1 \ldots M_N$
  - applicano ripetutamente una funzione base (f)
  - $V_k = f(V_{k-1}, M_k)$ con $V_0 = IV$ e $h = V_N$



Algoritmi di hash crittografici

nome	*blocco*	*digest*	*definizione*	*note*
MD2	8 bit	128 bit	RFC-1319	obsoleto
MD4	512 bit	128 bit	RFC-1320	obsoleto
MD5	512 bit	128 bit	RFC-1321	buono
RIPEMD	512 bit	160 bit	ISO/IEC 10118-3	ottimo
SHA-1	512 bit	160 bit	FIPS 180-1 RFC-3174	buono
SHA-224	512 bit	224 bit	FIPS 180-2 RFC-4634	ottimo (?)
SHA-256	512 bit	256 bit	. . .	ottimo (?)
SHA-384	512 bit	384 bit	. . .	ottimo (?)
SHA-512	512 bit	512 bit	. . .	ottimo (?)

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 2**69 hash operations, much less than the brute-force attack of 2**80 operations based on the hash length.
– collisions in SHA-0 in 2**39 operations.
– collisions in 58-round SHA-1 in 2**33 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`

Lunghezza del digest

- **importante per evitare *aliasing* (=collisioni):**
  - md1 = H(m1)
  - md2 = H(m2)
  - se m1 ≠ m2 si vorrebbe md1 ≠ md2
- **se l'algoritmo è ben ideato e genera un digest di N bit, allora la probabilità di aliasing è:**

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

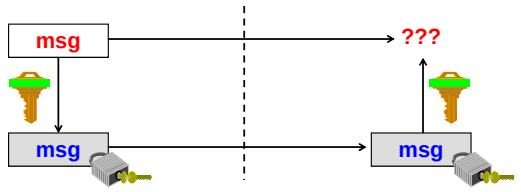
- **occorrono quindi digest con molti bit (perché si tratta di eventi statistici)**

MAC, MIC, MID

- **per garantire l'integrità dei messaggi si aggiunge agli stessi un codice:**
  MIC (Message Integrity Code)
- **spesso l'integrità non è utile senza l'autenticazione e quindi il codice (con doppia funzione) è anche detto:**
  MAC (Message Authentication Code)
- **per evitare attacchi di tipo replay si usa anche aggiungere ai messaggi un identificatore univoco:**
  MID (Message IDentifier)

Autenticazione tramite cifratura simmetrica

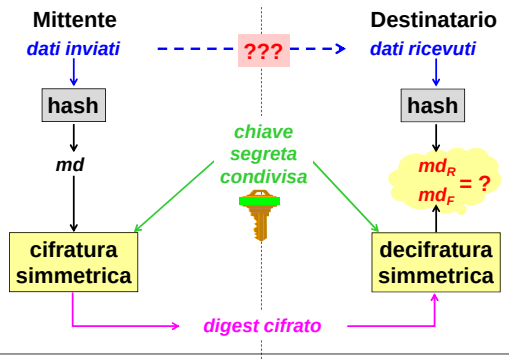
- **si invia anche una copia cifrata dei dati**
- **solo chi conosce la chiave può confrontare la copia con l'originale**
- **svantaggi: raddoppio dei dati trasmessi + tempo di calcolo**



Autenticazione tramite digest e cifratura simmetrica

- **si invia anche un digest cifrato dei dati**
- **A → B : mex, { digest (mex) } S**
- **solo chi conosce la chiave può confrontare il digest trasmesso con quello calcolato sui dati ricevuti**
- **svantaggio:**
  - due operazioni (digest + crittografia)
- **vantaggio:**
  - pochi dati aggiuntivi

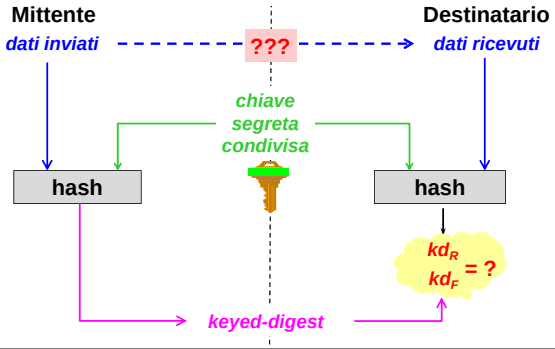
Digest + cifratura simmetrica



Autenticazione tramite keyed-digest

- **si invia anche un digest calcolato non solo sui dati ma anche su un segreto (chiave)**
- **A → B : mex, digest (mex, S)**
- **solo chi conosce la chiave può confrontare il digest trasmesso con quello calcolato sui dati ricevuti**
- **vantaggi:**
  - una sola operazione (digest)
  - pochi dati aggiuntivi

Keyed-digest



Keyed-digest: possibili errori

- **se kd = H (K || M) allora posso cambiare il messaggio post-ponendo uno o più blocchi:**
  - kd' = H (K || M || M') = f (kd, M')
- **se kd = H (M || K) allora posso cambiare il messaggio premettendo un opportuno blocco:**
  - kd = H (M' || M || K) scegliendo M' t.c. IV = f (IV, M')
- **difese:**
  - inserire tra i dati la lunghezza di M
  - definire kd = H (K || M || K)
  - usare un keyed-digest standard

Keyed-digest: alcuni standard

- **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 (rivisto)**
  - sha1(K || keyfill || data || datafill || K || sha1fill)

Keyed-digest: HMAC

- **RFC-2104**
- **funzione di hash H: blocco B byte, output L byte**
- **definizioni:**
  - ipad = 0x36 ripetuto B volte
  - opad = 0x5C ripetuto B volte
  - sconsigliate chiavi t.c. | K | < L
  - se L < | K | < B, Kp = K padded 0 fino a | Kp | = B
  - se | K | > B, Kp = H (K)
- **hmac = H(Kp ⊕ opad || H(Kp ⊕ ipad || data))**

CBC-MAC

- **sfrutta un algoritmo di cifratura simmetrico a blocchi, in modalità CBC con IV nullo, prendendo come MAC la cifratura dell'ultimo blocco**
- **messaggio M diviso in N blocchi M_1 … M_N**
- **iterazioni:**
  - $V_0 = 0$
  - for (k=1…N) do $V_k = enc (K, M_k \oplus V_{k-1})$
- **cbc-mac = V_N**
- **se basato su DES si chiama Data Authentication Algorithm (standard FIPS 113, ANSI X9.17)**

Integrità e riservatezza: come combinarle?

- **ipotesi di operazioni separate:**
  - riservatezza = cifratura simmetrica con K_1
  - integrità = keyed-digest (MAC) con K_2
- **opzione 1 – authenticate-and-encrypt (A&E)**
  - enc(p, K_1) || mac(p, K_2)
  - si deve sempre decifrare prima di verificare l'integrità (possibile un attacco DoS)
  - può fornire informazioni indirette sul plaintext
  - es. usato da SSH

Integrità e riservatezza: come combinarle?

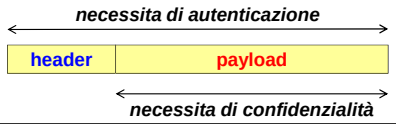
- **opzione 2 – authenticate-then-encrypt (AtE)**
  - enc(p || mac(p, K_2) , K_1)
  - si deve sempre decifrare prima di verificare l'integrità (possibile un attacco DoS)
  - es. usato da SSL e TLS
- **opzione 3 – encrypt-then-authenticate (EtA)**
  - enc(p, K_1) || mac(enc(p, K_1) , K_2)
  - si può evitare di decifrare se il MAC è errato
  - es. usato da IPsec

Integrità e riservatezza: sicurezza?

- **una combinazione scorretta di algoritmi sicuri può condurre a … un risultato insicuro!**
- **autenticate-and-encrypt (A&E)**
  - insicura a meno che sia fatta in un passo solo
- **authenticate-then-encrypt (AtE)**
  - sicura solo con CBC o cifratura stream
- **encrypt-then-authenticate (EtA)**
  - la combinazione più sicura … ma bisogna prestare attenzione agli errori di implementazione
    - includere sempre nel MAC l'IV e gli algoritmi
- **iniziative per lo sviluppo di un algoritmo unico AE**

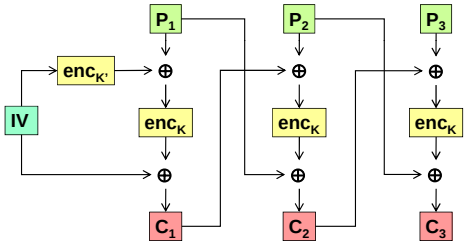
Authenticated encryption

- **unica operazione per riservatezza+autenticazione (ed integrità)**
  - una sola chiave ed un solo algoritmo
  - maggior velocità
  - meno probabilità di errori nel combinare le funzioni
- **necessità applicative (es. messaggi di posta elettronica, pacchetti di rete):**



IGE (Infinite Garble Extension)

- **errore su tutti i blocchi dopo quello manipolato**
- **facile quindi aggiungere un ultimo blocco di controllo (es. tutti zero, contatore del n. di blocchi)**



Authenticated encryption: standard

- **ISO/IEC 19772:2009 definisce 6 modi standard:**
  - OCB 2.0 (Offset Codebook Mode)
  - AESKW (AES Key Wrap)
  - CCM (CTR mode with CBC-MAC)
  - EAX (Encrypt then Authenticate then X(trans)late) = CTR + OMAC
  - Encrypt-then-MAC
  - GCM (Galois/Counter Mode)
- **altri modi esistono e sono/verranno raccomandati da altri organismi (es. NIST, IETF)**

Authenticated encryption: applicazioni

- **802.11i usa CCM**
- **ZigBee usa CCM* (=CCM + auth-only + enc-only)**
- **ANSI C12.22 (trasmissione in rete di misure elettroniche, es. contatori di abitazione) usa EAX'**

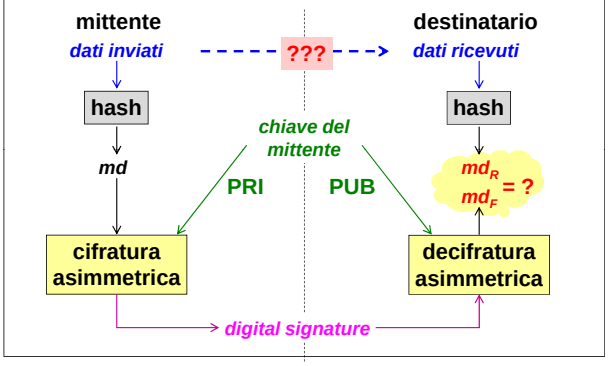
Raccomandazioni NIST per uso di algoritmi di cifratura a blocchi

- **NIST SP 800-38**
- **parte A (dic'01) = cinque modi per confidenzialità**
  - ECB, CBC, CFB, OFB e CTR
- **addendum (ott'10) = tre CBC ciphertext stealing**
- **parte B (mag'05) = un modo di autenticazione**
  - CMAC (~OMAC > XCBC > CBC-MAC)
- **parte C (lug'07) = auth. encryption = CCM**
- **parte D (nov'07) = high-speed auth. enc. = GCM**
- **parte E (gen'10) = confidenzialità per memorie a blocchi = XTS-AES**
- **draft = uso di AESKW per aut. enc. di chiavi**

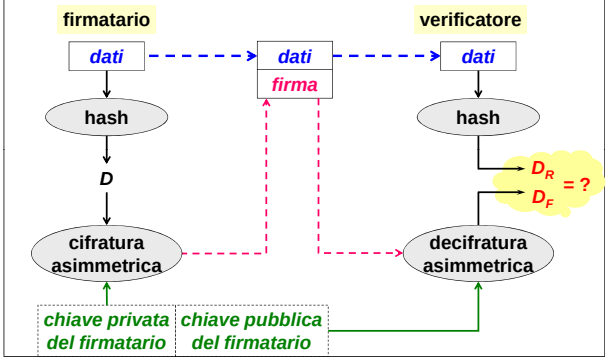
Autenticazione tramite digest e cifratura asimmetrica

- **si invia anche un digest (cifrato con la chiave privata del mittente)**
- **chi conosce la chiave pubblica può confrontare il digest trasmesso con quello calcolato sui dati ricevuti**
- **A → B : mex, { digest(mex) } priA**
- **FIRMA DIGITALE !!!**

Firma digitale



Firma e verifica



Autenticazione e integrità: analisi

- **tramite segreto condiviso:**
  - utile solo per il ricevente
  - non usabile come prova senza rivelare la chiave segreta
  - non usabile per il non ripudio
- **tramite crittografia asimmetrica:**
  - essendo lenta la si applica al digest
  - usabile come prova formale
  - usabile per il non ripudio
  - = firma digitale (*digital signature*)

Firma digitale o firma autografa?

- **firma digitale = autenticazione + integrità**
- **firma autografa = autenticazione**
- **meglio quindi la firma digitale, perché indissolubilmente legata ai dati**
- **nota bene: ogni utente non è dotato di una firma digitale ma di una chiave privata con cui può generare infinite firme digitali (una per ogni documento diverso)**

Certificato a chiave pubblica

"Una struttura dati per legare in modo sicuro una chiave pubblica ad alcuni attributi"

- **tipicamente lega chiave a identità ... ma sono possibili altre associazioni (es. indirizzo IP)**
- **firmato in modo elettronico dall'emettitore: l'autorità di certificazione (CA)**
- **con scadenza temporale**
- **revocabile sia dall'utente sia dall'emettitore**

Formati per certificati a chiave pubblica

- **X.509:**
  - v1, v2 (ISO)
  - v3 (ISO + IETF)
- **non X.509:**
  - PGP
  - SPKI (IETF)
- **PKCS-6:**
  - RSA, in parte compatibile con X.509
  - obsoleto

Struttura di un certificato X.509

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)

- **è l'infrastruttura ...**
- **tecnica ed organizzativa ...**
- **preposta alla creazione, distribuzione e revoca dei certificati a chiave pubblica**

Revoca dei certificati

- **un certificato può essere revocato prima della sua scadenza naturale**
  - su richiesta del titolare (subject)
  - autonomamente dall'emettitore (issuer)
- **quando si valida una firma si deve verificare che il certificato fosse valido all'atto della firma**
- **verifica a carico del ricevente (relying party, RP)**

Meccanismi di revoca

- **CRL (Certificate Revocation List)**
  - elenco di certificati revocati
  - firmato dalla CA o da un delegato
- **OCSP (On-line Certificate Status Protocol)**
  - risposta puntuale su un singolo certificato
  - firmata dal server

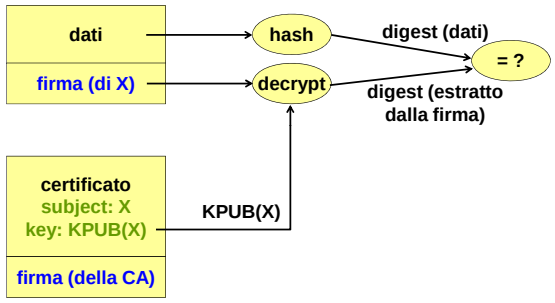
Struttura di una CRL X.509

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

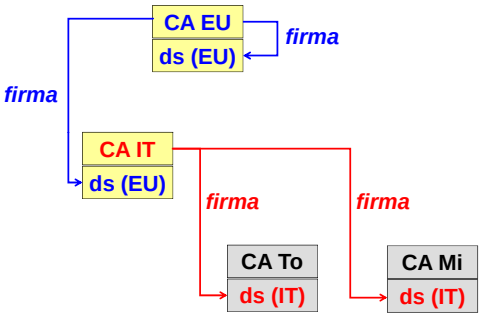
Verifica di una firma / certificato

- come verificare che un certificato a chiave pubblica (firmato da CA1) sia autentico?
- … occorre il certificato della chiave di CA1 (che sarà firmato da CA2)
- e come si verifica quest'ultimo?
- … occorre il certificato della chiave di CA2 (che sarà firmato da CA3)
- … e così via …
- occorre un'infrastruttura (gerarchia?) di certificazione e distribuzione

Verifica di una firma / certificato



Gerarchia di certificazione



Valutazione delle prestazioni

- le prestazioni crittografiche non dipendono dalla RAM ma dalla CPU (architettura ed instruction set) e dalla sua cache
- le prestazioni non sono un problema sui client (a meno che siano particolarmente carichi per applicazioni locali)
- le prestazioni possono essere un problema sui server o sui nodi di rete (es. router):
  - uso di acceleratori crittografici
  - acceleratori specifici (es. SSL, IPsec) o generici

Prestazioni (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 firme/s	2472.1 verifiche/s

Prestazioni (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 firme/s	1682.0 verifiche/s

NSA suite B

- **per prodotti COTS che trattano informazioni SBU (Sensitive But Unclassified) e Classified**
- **comprende i seguenti algoritmi:**
  - (cifratura simmetrica) AES-128 e AES-256
  - (hash) SHA-256 e SHA-384
  - (key agreement) ECDH e ECMQV
  - (firma digitale) ECDSA
- **per informazioni fino a livello Secret:**
  - AES-128 + SHA-256 + EC-256
- **per informazioni al livello Top Secret:**
  - AES-256 + SHA-384 + EC-384

Lunghezza chiavi e digest (NIST, 2011)

- **equivalenze definite in NIST SP800-57 (part 1, rev.3)**
- **FFC = Finite Field Cryptography (es. DSA, D-H)**
- **IFC = Integer Factorization Cryptography (es. RSA)**

symm.	FFC	IFC	ECC	hash	anni
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

Lunghezza chiavi e digest (ECRYPT II, 2011)

simm.	80	96	112	128	256
hash	160	192	224	256	512
asimm.	1248	1776	2432	3248	15424
ECC	160	192	224	256	512
	very short term	legacy	medium term	long term (2011-'40)	foreseeable future (quantum computers)

- **www.keylength.com**

Perchè non compriamo tutto dagli USA?

- **esportazione di materiale crittografico soggetta alle medesime restrizioni del materiale nucleare (!)**
- **… a meno che il livello di protezione sia molto basso:**
  - chiave simmetrica limitata a 40 bit (2^{40} tentativi = poche ore di CPU)
  - chiave asimmetrica limitata a 512 bit
- **esempio: Netscape, Internet Explorer, … (versione esportazione)**

Key escrow: una novità?

- **dicembre 1996: l'amministrazione USA autorizza l'esportazione di prodotti di crittografia semi-robusti (56 bit) se incorporano funzioni di key-escrow**
- ***non* si applica ai prodotti interni USA**
- **key escrow = possibilità di recuperare una chiave anche senza il consenso del proprietario**
- **esempio: Lotus Notes 4.x usava chiavi simmetriche da 64 bit ma ne cifrava 24 bit con la chiave pubblica della NSA**
- **problema: chi decide quando è necessario recuperare una chiave?**

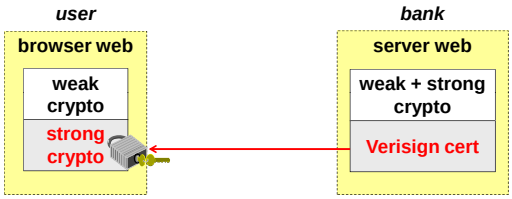
The many editions of Notes

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).

Cambiamenti della politica crittografica USA

- **giugno 1997:**
  - permesso di esportare server Web sicuri purché usati da filiali estere di aziende USA oppure in ambito finanziario
  - per verificare il reale uso si ricorre a speciali certificati rilasciati solo da VeriSign
- **settembre 1998:**
  - permesso esteso ad assicurazioni ed istituzioni sanitarie
  - nessun permesso per chiavi sino a 56 bit

Step-up (gated) cryptography



Key recovery: un'altra novità?

- **è permessa l'esportazione dagli USA di prodotti con crittografia forte (es. 128 bit per le chiavi simmetriche) se:**
  - incorporano funzioni di key-recovery
  - con un centro di recovery autorizzato dal governo USA
- **le chiavi simmetriche usate dagli utenti sono cifrate con la chiave pubblica del centro di recovery**
- **così diventa una problematica politica mentre il key-recovery è un problema pratico reale che deve affrontare chiunque usi strumenti crittografici**

Novità nella politica di esportazione USA

- **gennaio 2000**
- **permesso di esportare ...**
  - prodotti off-the-shelf ...
  - che abbiano passato una "one-time review"
- **oppure**
  - prodotti il cui codice sorgente sia liberamente disponibile in Internet
- **disponibili upgrade per i principali prodotti**
  - dubbi sull'esistenza di back-door

Import / export / domestic controls

- **controlli esistono ancora in molti paesi**
- **Bert-Jaap Koops ha creato un'ottima survey:**
  http://rechten.uvt.nl/koops/cryptolaw/cls-sum.htm

Chaos Computer Club analyzes government malware

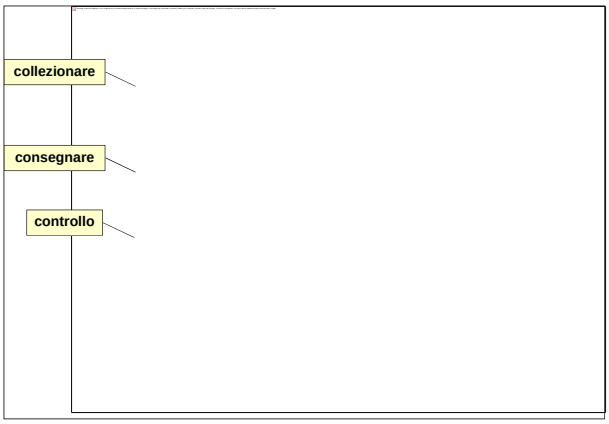
The largest European hacker club, "Chaos Computer Club" (CCC), has reverse engineered and analyzed a "lawful interception" malware program used by German police forces.

It has been found in the wild and submitted to the CCC anonymously.

The malware can not only siphon away intimate data but also offers a remote control or backdoor functionality for uploading and executing arbitrary other programs.

Significant design and implementation flaws make all of the functionality available to anyone on the Internet.

www.ccc.de/en/updates/2011/staatstrojaner (8/10/11)



Bibliografia

- **B.Schneier:**
  "Applied cryptography"
- **A.Menezes, P. van Oorschot, S.Vanstone:**
  "Handbook of Applied Cryptography"
- **W.Stallings:**
  "Cryptography and network security"
- **C.P.Pfleeger, S.Pfleeger:**
  "Security in computing"
- **S.Garfinkel, G.Spafford:**
  "Practical Unix and Internet security"
- **W.R.Cheswick, S.M.Bellovin:**
  "Firewalls and Internet security"

Bibliografia (in Italiano)

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