

Sicurezza Informatica

Crittografia a Chiave Pubblica

RSA



RSA

- Schema di Cifratura a Chiave Pubblica ideato da Ron Rivest, Adi Shamir e Leonard Adleman nel 1977
 - Vincitori del Turing Award nel 2002
 - Turing Award istituito nel 1966
- Brevettato negli Stati Uniti fino al 2000

Standard

- PKCS#1
 - Versione corrente 2.2, 27 ottobre 2012
- IEEE P1363 (IEEE Std 1363-2000 e1363a-2004)
 - Invece di RSA lo schema è chiamato *Integer Factorization Encryption Scheme*
- ANSI X9.31-1998
 - Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry (rDSA)
 - RSA usato come schema di firma digitale

Utilizzo di RSA

- RSA è il crittosistema a chiave pubblica usato maggiormente
 - La crittografia basata su curve ellittiche (ECC) sta diventando sempre più popolare
- RSA è usato principalmente per due applicazioni
 - Trasporto di chiavi simmetriche
 - Firme digitali

RSA

- Dobbiamo definire i tre algoritmi di uno schema di cifratura a chiave pubblica
- Gen
 - Genera la coppia di chiavi (pubblica, privata)
- Enc
 - Cifra un messaggio usando la chiave pubblica
- Dec
 - Decifra un messaggio usando la chiave privata

Chiave RSA

- Si scelgono due numeri primi p e q e si calcola $N = p \cdot q$
- Lo spazio dei messaggi è $Z_N = \{0, 1, \dots, n-1\}$
- Si sceglie un valore e relativamente primo a $\phi(N) = (p-1)(q-1)$
 - $1 = \text{mcd}(e, (p-1)(q-1))$ e può essere anche maggiore di $\phi(N)$
- Si calcola d come l'inverso moltiplicativo di e modulo $\phi(N)$
 - $ed = 1 \text{ mod } (p-1)(q-1)$
- Chiave pubblica (e, N) – Chiave privata (d, N)

Generazione Chiavi

GenRSA

Input: Security parameter 1^n

Output: N, e, d as described in the text

$(N, p, q) \leftarrow \text{GenModulus}(1^n)$

$\phi(N) := (p-1)(q-1)$

find e such that $\gcd(e, \phi(N)) = 1$

compute $d := [e^{-1} \bmod \phi(N)]$

return N, e, d

Scegliere due numeri primi grandi non è banale

e è scelto relativamente primo a $(p-1)(q-1)$ in modo da ammettere un inverso moltiplicativo

Se si sceglie il valore e primo, allora e è anche relativamente primo a $(p-1)(q-1)$

GenModulus genera il modulo N e la sua fattorizzazione

I numeri primi p e q devono essere grandi (512, 1024, 2048 bit)

I numeri primi p e q devono avere circa la stessa dimensione (in bit)

Se N deve essere di modLen bit, p e q si scelgono di $\text{modLen}/2$ bit

Cifrare/Decifrare con RSA

Si usa l'elevamento a potenza modulo n

RSA Encryption Given the public key $(n, e) = k_{pub}$ and the plaintext x , the encryption function is:

$$y = e_{k_{pub}}(x) \equiv x^e \pmod{n} \quad (7.1)$$

where $x, y \in \mathbb{Z}_n$.

RSA Decryption Given the private key $d = k_{pr}$ and the ciphertext y , the decryption function is:

$$x = d_{k_{pr}}(y) \equiv y^d \pmod{n} \quad (7.2)$$

where $x, y \in \mathbb{Z}_n$.

La sicurezza dello schema si basa sul fatto che difficile ricavare la chiave privata d a partire dalla chiave pubblica (e, n)

Un piccolo esempio

Alice

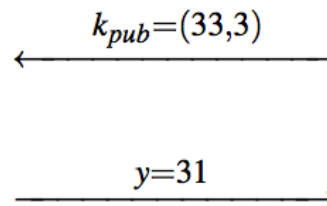
message $x = 4$

$$y = x^e \equiv 4^3 \equiv 31 \pmod{33}$$

$$\begin{aligned} 4^3 &= 64 \\ &= 33 \times 1 + 31 \\ &= 31 \pmod{33} \end{aligned}$$

Bob

1. choose $p = 3$ and $q = 11$
2. $n = p \cdot q = 33$
3. $\Phi(n) = (3 - 1)(11 - 1) = 20$
4. choose $e = 3$
5. $d \equiv e^{-1} \equiv 7 \pmod{20}$



$$y^d = 31^7 \equiv 4 = x \pmod{33}$$

$$\begin{aligned} 31^7 &= 27.512.614.111 \\ &= 833.715.579 \times 33 + 4 \\ &= 4 \pmod{33} \end{aligned}$$

Altro esempio

Se vogliamo cifrare un testo associamo ad ogni lettera un numero

A \rightarrow 00, B \rightarrow 01, ..., Z \rightarrow 25

Lo spazio è codificato con 26

Tutti i valori sono minori di 33

Plaintext (P)		Ciphertext (C)		After decryption		
Symbolic	Numeric	P^3	$P^3 \pmod{33}$	C^7	$C^7 \pmod{33}$	Symbolic
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E
Sender's computation				Receiver's computation		

Perché RSA funziona?

Se decifriamo un messaggio cifrato eseguiamo le seguenti operazioni

$$d_{k_{pr}}(y) = d_{k_{pr}}(e_{k_{pub}}(x)) \equiv (x^e)^d \equiv x^{de} \equiv x \pmod{n}. \quad \text{Perché } x^{de} \equiv x \pmod{n}?$$

Se $d \cdot e \equiv 1 \pmod{\phi(n)}$ allora $de = 1 + t \cdot \phi(n)$ per un qualche $t > 0$, quindi

$$d_{k_{pr}}(y) \equiv x^{de} \equiv x^{1+t \cdot \phi(n)} \equiv x^{t \cdot \phi(n)} \cdot x^1 \equiv (x^{\phi(n)})^t \cdot x \pmod{n}.$$

Se $1 = \text{mcd}(x, n)$, applichiamo il teorema di Eulero, ottenendo

$$d_{k_{pr}}(y) \equiv (x^{\phi(n)})^t \cdot x \equiv 1 \cdot x \equiv x \pmod{n}.$$

Theorem 6.3.3 Euler's Theorem

Let a and m be integers with $\text{gcd}(a, m) = 1$, then:

$$a^{\phi(m)} \equiv 1 \pmod{m}.$$

Se $1 \neq \text{mcd}(x, n)$, allora o $x = r \cdot p$ con $r < q$ oppure $x = s \cdot q$ con $s < p$

Senza perdere di generalità, assumiamo $x = r \cdot p$ con $r < q$, quindi

$$1 = \text{mcd}(x, q)$$

Perché RSA funziona?

Il teorema di Eulero vale nella seguente forma per qualsiasi intero t

$$1 \equiv 1^t \equiv (x^{\Phi(q)})^t \pmod{q},$$

Consideriamo nuovamente $(x^{\Phi(n)})^t$

$$(x^{\Phi(n)})^t \equiv (x^{(q-1)(p-1)})^t \equiv ((x^{\Phi(q)})^t)^{p-1} \equiv 1^{(p-1)} = 1 \pmod{q}. \quad \text{Per cui}$$

$$(x^{\Phi(n)})^t = 1 + u \cdot q,$$

Consideriamo nuovamente $x \cdot (x^{\Phi(n)})^t$

$$\begin{aligned} x \cdot (x^{\Phi(n)})^t &= x + x \cdot u \cdot q \\ &= x + (r \cdot p) \cdot u \cdot q \\ &= x + r \cdot u \cdot (p \cdot q) \\ &= x + r \cdot u \cdot n \end{aligned}$$

$$x \cdot (x^{\Phi(n)})^t \equiv x \pmod{n}.$$

Quindi

$$d_{k_{pr}}(y) = (x^{\Phi(n)})^t \cdot x \equiv x \pmod{n}$$

anche quando $1 \neq \text{mcd}(x, n)$

Operazioni in \mathbb{Z}_n visto come anello

$(\mathbb{Z}_n, +)$ gruppo e (\mathbb{Z}_n, \cdot) monoide (gruppo moltiplicativo senza inverso)

Efficienza operazioni

- RSA esegue le seguenti operazioni
 - Generazione primi p e q
 - Generazione di e , calcolo di d
 - Elevazione a potenza modulare
 - Per cifrare e decifrare
 - Metodo naive inefficiente, meglio usare la tecnica square-and-multiply
- FARE ATTENZIONE A
COME SI GENERANO
I PARAMETRI**

Elevamento a potenza

Metodo naive per calcolare $x^y \bmod n$

```
NaiveModularPower(x, y, n)
  r=1
  for i = 1 to y do
    r = (r · x) mod n
  return a
```

- Se l'esponente è composto da 512 bit, esso può essere un numero grande fino a 2^{512}
- Sono necessarie $O(2^{512})$ operazioni
- Il numero di atomi dell'universo visibile è stimato in 2^{300}



Square-and-multiply

- Se $y = y_t y_{t-1} \dots y_2 y_1 y_0$ in binario, allora il seguente algoritmo è più efficiente del precedente

```
ModularPower(x, y, n)
  r=1
  for i = t downto 0 do
    r = r2 mod n
    if  $y_i == 1$  then
      r = (r · x) mod n
  return r
```

Se $t+1 = 512$,
il numero di operazioni è
proporzionale a $O(768)$

Ogni operazione è eseguita
su numeri MOLTO grandi

- Numero di elevamenti al quadrato: $t+1$
- Numero medio di moltiplicazioni: $(t+1)/2$

Velocizzare le operazioni RSA

- L'elevamento a potenza è un'operazione onerosa
- Anche usando la tecnica square-and-multiply, RSA potrebbe essere *lento* su dispositivi limitati (e.g., smart-card)
- Si usano altri trucchi
 - Esponente e piccolo
 - Teorema Cinese dei Resti (Chinese Remainder Theorem - CRT)
 - Elevamento a potenza con pre-computazione

Esponente e piccolo

- Non inficia la sicurezza di RSA
- Velocizza il calcolo dell'elevamento a potenza
- Tipici valori utilizzati

Public Key	e as binary string	#MUL + #SQ
$2^1 + 1 = 3$	$(11)_2$	$1 + 1 = 2$
$2^4 + 1 = 17$	$(1\ 0001)_2$	$4 + 1 = 5$
$2^{16} + 1$	$(1\ 0000\ 0000\ 0000\ 0001)_2$	$16 + 1 = 17$

Settando $r=x$ invece di $r=1$ si riduce di uno il numero di squaring

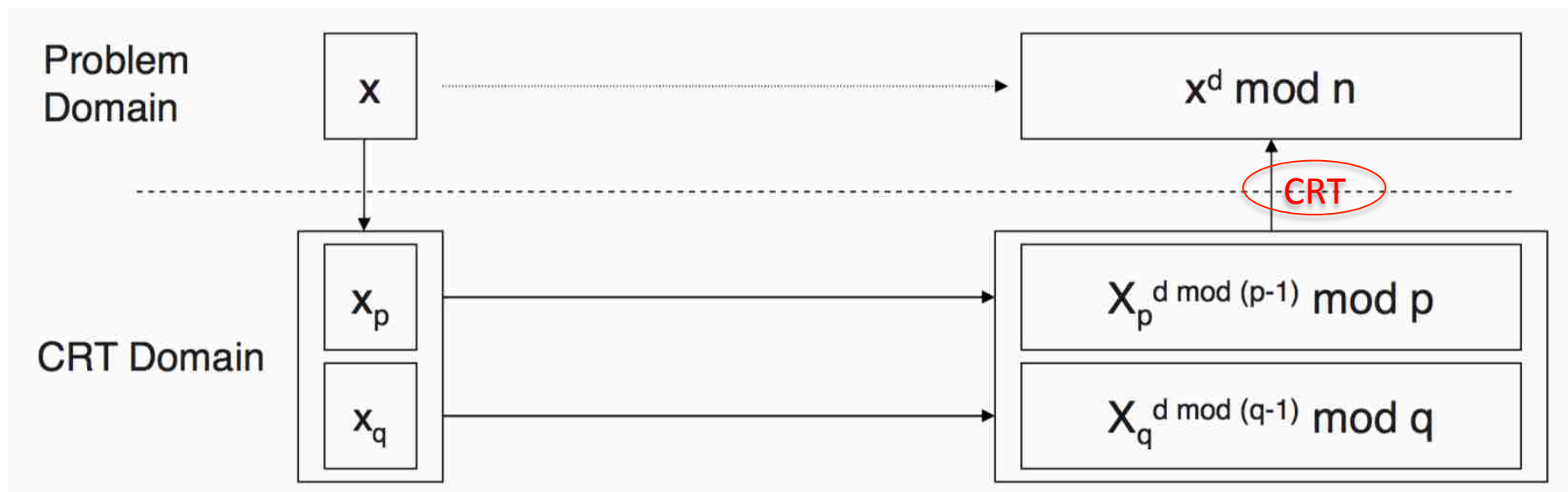
- Questo trucco è utilizzato comunemente
 - SSL/TLS

CRT e l'esponente d

- Se si sceglie d piccolo si possono avere problemi di sicurezza
 - d dovrebbe avere almeno $0,3 \cdot t$ bit se t è la lunghezza del modulo n
- L'utilizzo Teorema Cinese dei Resti (CRT) velocizza il calcolo di $x^d \bmod n$
- Il calcolo di $x^{d \bmod \phi(n)} \bmod n$ è ridotto al calcolo di $x^{d \bmod p-1} \bmod p$ e di $x^{d \bmod q-1} \bmod q$
 - p e q sono *piccoli* in confronto ad n

Principio alla base

- Trasformazione degli operandi nel dominio CRT
- Calcolo operazioni nel dominio CRT
- Tornare al dominio di partenza (trasformazione inversa)



CRT – trasformazione

- È necessario conoscere p e q
- Il valore x è rappresentato nel dominio CRT calcolando (x_p, x_q)
 - $x_p \equiv x \pmod{p}$ e $x_q \equiv x \pmod{q}$
- Il valore d è rappresentato nel dominio CRT calcolando (d_p, d_q)
 - $d_p \equiv d \pmod{p-1}$ e $d_q \equiv d \pmod{q-1}$

CRT – elevamento a potenza

- Sono necessarie due elevamenti a potenza nel dominio CRT

$$y_p \equiv x_p^{d_p} \pmod{p} \text{ e } y_q \equiv x_q^{d_q} \pmod{q}$$

- Usiamo square-and-multiply
 - #sq= $0,5 \cdot t$
 - #mult = $0,25 \cdot t$
 - #op complessive = $2(0,5 \cdot t + 0,25 \cdot t) = 1,5 \cdot t$

operandi di $0,5 \cdot t$ bit – elevamento a potenza 4 volte più veloce

CRT – ritorno al dominio iniziale

- Si calcolano i coefficienti c_p e c_q come segue

$$c_p \equiv q^{-1} \bmod p, \quad c_q \equiv p^{-1} \bmod q$$

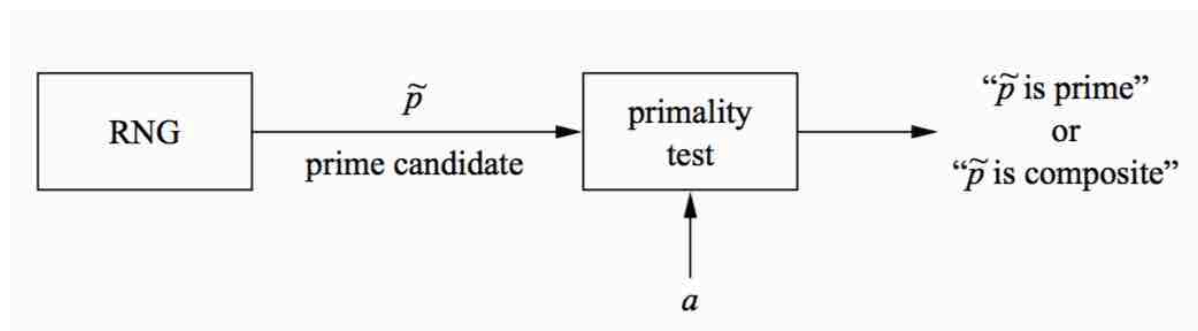
- Da (y_p, y_q) si passa a y come segue

$$y \equiv [q c_p] y_p + [p c_q] y_q \bmod n$$

- $q \cdot c_p$ e $p \cdot c_q$ possono essere pre-computati

Generazione di primi

- La generazione delle chiavi RSA richiede la generazione di due primi molto grandi
 - In genere sono grandi la metà del modulo
- Si genera un intero e si verifica se esso sia primo
 - Test primalità



Se scegliamo \tilde{p} dispari

$$P(\tilde{p} \text{ is prime}) \approx \frac{2}{\ln(\tilde{p})}$$

Test Primalità

Fermat Primality Test

Input: prime candidate \tilde{p} and security parameter s

Output: statement “ \tilde{p} is composite” or “ \tilde{p} is likely prime”

Algorithm:

```
1  FOR  $i = 1$  TO  $s$ 
1.1  choose random  $a \in \{2, 3, \dots, \tilde{p} - 2\}$ 
1.2  IF  $a^{\tilde{p}-1} \not\equiv 1$ 
1.3    RETURN (“ $\tilde{p}$  is composite”)
2  RETURN (“ $\tilde{p}$  is likely prime”)
```

Probabilità di errore $1/2^s$

Complessità $O(s \log^3 p)$

Miller–Rabin Primality Test

Input: prime candidate \tilde{p} with $\tilde{p} - 1 = 2^u r$ and security parameter s

Output: statement “ \tilde{p} is composite” or “ \tilde{p} is likely prime”

Algorithm:

```
1  FOR  $i = 1$  TO  $s$ 
    choose random  $a \in \{2, 3, \dots, \tilde{p} - 2\}$ 
1.2   $z \equiv a^r \pmod{\tilde{p}}$ 
1.3  IF  $z \not\equiv 1$  and  $z \not\equiv \tilde{p} - 1$ 
1.4    FOR  $j = 1$  TO  $u - 1$ 
         $z \equiv z^2 \pmod{\tilde{p}}$ 
        IF  $z = 1$ 
            RETURN (“ $\tilde{p}$  is composite”)
1.5  IF  $z \neq \tilde{p} - 1$ 
        RETURN (“ $\tilde{p}$  is composite”)
2  RETURN (“ $\tilde{p}$  is likely prime”)
```

Probabilità di errore $1/4^s$

Complessità $O(s \log^3 p)$

Densità numeri primi

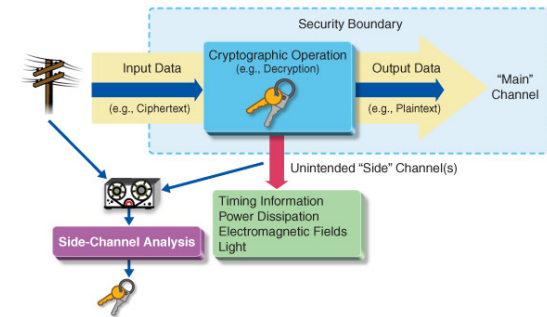
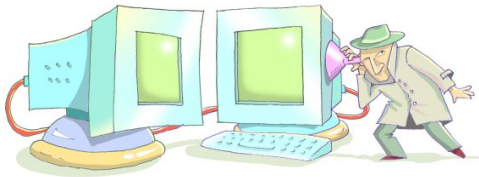
- $\pi(x)$ = numero dei primi minori o uguali a x
- Vale il seguente limite

$$\lim_{x \rightarrow \infty} \frac{\pi(x)}{x / \log(x)} = 1, \quad \text{quindi} \quad \pi(x) \sim \frac{x}{\log x}.$$

- Se scegliamo un numero a caso tra in $[2, x]$ la probabilità che esso sia primo è circa $1/\log x$
 - Escludendo i numeri pari la probabilità è $2/\log x$

log è il logaritmo in base e (Nepero)

Dimensione in bit	Numero medio tentativi
512	177
1024	355
2048	710
3072	1065
4096	1420



Attacchi ad RSA



Attacchi ad RSA

- Se d non è molto grande (ad esempio $d < 2^{16}$), allora tramite un attacco di forza bruta possiamo ricavare d a partire da e
- Se $d < N^{0,292}$, esiste un attacco che ci permette di calcolare d a partire da e
 - L'attacco è reso inefficace se si sceglie $e > N^{1,875}$
 - Si addiziona ad e un multiplo di $\phi(N)$ per ottenere un valore maggiore di $N^{1,875}$

Dan Boneh e Glenn Durfee

[Cryptanalysis of RSA with Private Key \$d\$ Less than \$N^{0.292}\$](#)

EUROCRYPT'99, LNCS 1592, pp. 1–11, 1999

Attacchi ad RSA

- Se un avversario riuscisse a fattorizzare di n saprebbe calcolare d
 - Fattorizza $n=p \cdot q$
 - Calcola $\phi(n)=(p-1) \cdot (q-1)$
 - Calcola $d = e^{-1} \bmod (p-1) \cdot (q-1)$
- Se un avversario riuscisse a calcolare $\varphi(n)$ saprebbe calcolare d

$$\varphi(n) = (p-1)(q-1) = n - (p+q) + 1 \implies \begin{cases} p + q &= n - \varphi(n) + 1 \\ pq &= n \end{cases}$$

$$x^2 - (p+q)x + pq = 0$$

L'avversario deve risolvere un'equazione di secondo grado

Attacchi RSA

- Se un avversario riuscisse a calcolare d , allora potrebbe fattorizzare n con probabilità pari ad almeno $\frac{1}{2}$. Ripetendo m volte la procedura la probabilità di successo sale a $1 - 1/2^m$
- Fattorizzare $n \rightarrow$ Calcolare d
- Calcolare $d \rightarrow$ Fattorizzare n

Calcolare d è equivalente a fattorizzare n

Sicurezza cifratura RSA

- Dato $c = m^e \bmod n$, se un avversario riuscisse a fattorizzare di n saprebbe calcolare m
- È vero anche il contrario?
- Dato $c = m^e \bmod n$, se un avversario riuscisse a calcolare m saprebbe fattorizzare n ?
- **Problema aperto!!**

Fattorizzazione

- Trovare i fattori primi di un numero n grande è un problema computazionalmente difficile
- L'algoritmo banale ha una complessità esponenziale $O(n^{1/2})$ nel numero di bit necessari per rappresentare n
 - $n \approx 2^{1024}$, allora $n^{1/2} \approx 2^{512}$
- Esistono algoritmi migliori per la fattorizzazione
 - Metodo di Dixon, Quadratic Sieve
 - General/Special **Field Number Sieve**

Complessità Fattorizzazione

- Per rappresentarla si usa la notazione L , simile alla notazione asintotica O -grande
- $L_n[\alpha, c]$ è definita come

$$L_n[\alpha, c] = e^{(c+o(1))(\ln n)^\alpha (\ln \ln n)^{1-\alpha}}$$

dove c è una costante positiva mentre $0 \leq \alpha \leq 1$

- $L_n[\alpha, c]$ è esponenziale in $\ln n$
 - numero di bit necessari per rappresentare n

$o(1)$ indica una qualsiasi funzione di n che tende a zero quando n tende ad infinito

Complessità Crivelli

Metodo naive $L_n[1, 1]$


- Metodo di Dixon

$$L_n[1/2, 2 \cdot 2^{1/2}]$$

- Quadratic Sieve

Usato per interi compresi tra 2^{200} e 2^{300}


$$L_n[1/2, 1]$$


$$\left(\frac{64}{9}\right)^{1/3}$$

- General Number Field Sieve

Usato per interi maggiori di 2^{300}

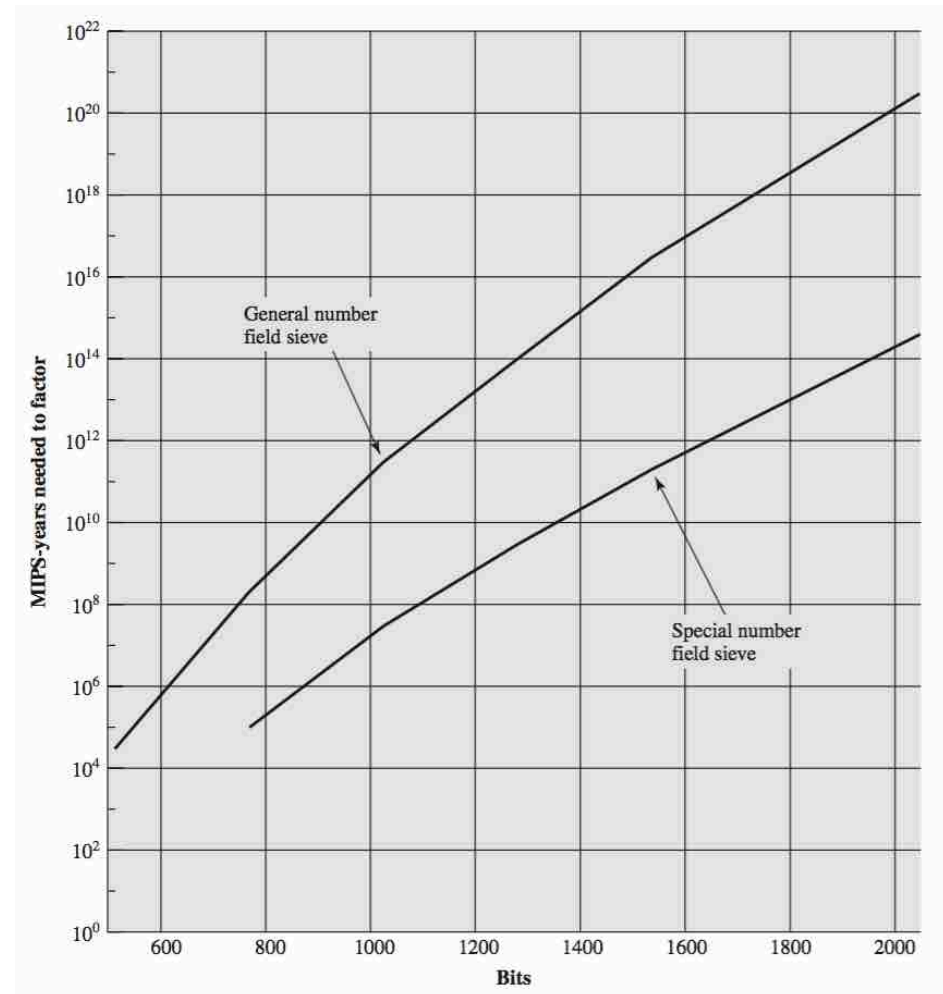
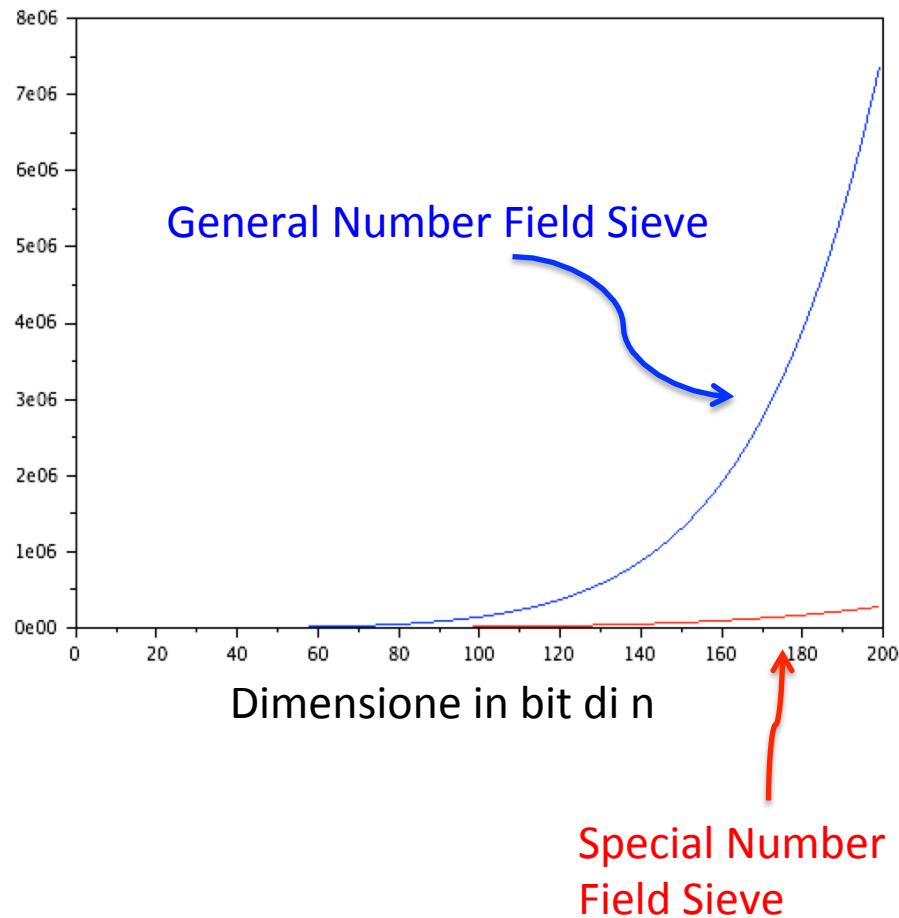
$$L_n[1/3, 1,923]$$


$$\left(\frac{32}{9}\right)^{1/3}$$

- Special Number Field Sieve

$$L_n[1/3, 1,526]$$

- Efficiente per numeri della forma $r^t \pm s$, con r ed s piccoli



Intel Core i7 3770K	106.924 MIPS at 3,9 GHz
Intel Core i7 3630QM	113.093 MIPS at 3,2 GHz
Intel Core i7 4770K	133.740 MIPS at 3,9 GHz
Intel Core i7 5960X	238.310 MIPS at 3,0 GHz
Raspberry Pi 2	4.744 MIPS at 1,0 GHz
Intel Core i7 6950X	317.900 MIPS at 3,0 GHz

MIPS: un milione di istruzioni per secondo
MIPS-year: istruzioni per anno $\approx 3.15 \times 10^{13}$


Intel Core i7 6950X $\approx 3,18 \times 10^5$ MIPS-year
Servono circa $3,14 \times 10^5$ processori per fattorizzare un numero (non in *forma speciale*) di 1000 bit in un anno

RSA Factoring Challenge

- Sfida lanciata nel 1991 da RSA Laboratories per incoraggiare la ricerca nella teoria dei numeri computazionale
- La sfida consisteva in un elenco di numeri (generati come prodotto di due primi) a cui era associato un premio in dollari in caso di fattorizzazione
- La sfida è stata annullata nel 2007, ma alcuni numeri non sono stati ancora fattorizzati

Progressi nella fattorizzazione RSA

Numero RSA	Numero cifre decimali	Numero cifre binarie	Data	Premio offerto
RSA-100	100	332	aprile 1991	\$ 1.000
RSA-110	110	365	aprile 1992	\$ 4.429
RSA-120	120	398	giugno 1993	\$ 5.898
RSA-129	129	428	aprile 1994	\$ 100
RSA-130	130	431	aprile 1996	\$ 14.527
RSA-140	140	465	febbraio 1999	\$ 17.226
RSA-155	155	512	agosto 1999	\$ 9.383
RSA-160	160	530	aprile 2003	\$ 8.787
RSA-576	174	576	dicembre 2003	\$ 10.000
RSA-200	200	663	maggio 2005	\$ 20.000
RSA-640	193	640	novembre 2005	\$ 20.000
RSA-220	220	729	maggio 2016	\$ 30.000
RSA-232	232	768	dicembre 2009	\$ 30.000



Premio offerto
nel 1977 da
Martin Gardner su
Scientific American

Maggiori dettagli su https://en.wikipedia.org/wiki/RSA_Factoring_Challenge

Short message and small e

- Se la chiave pubblica è piccola (spesso $e=3$) ed il messaggio da cifrare è corto, allora si può recuperare il messaggio senza conoscere d
 - In uno schema ibrido si cifra una chiave di pochi byte
- Assumiamo $e=3$ e $c=m^3 < N$
- Calcoliamo m come $c^{1/3}$
 - Il calcolo è fatto sugli interi

Common Modulus Attack

- Stesso modulo per chiavi diverse
 - Alice: (e_1, n) – Bob: (e_2, n)
- Se lo stesso messaggio m è inviato ad Alice e Bob, allora si riesce a recuperarlo senza conoscere la chiave segreta nel caso $1 = \text{mcd}(e_1, e_2)$
 - Succede perché con grande probabilità e_1 ed e_2 sono numeri primi
- Supponiamo che $c_1 = m^{e_1} \bmod n$ e $c_2 = m^{e_2} \bmod n$
- Si usa Euclide esteso per calcolare x ed y tali che $1 = xe_1 + ye_2$
- Si calcola m come $c_1^x c_2^y = c_1^x c_2^y = m^{xe_1} m^{ye_2} = m^{xe_1 + ye_2} = m$

Common Exponent Attack

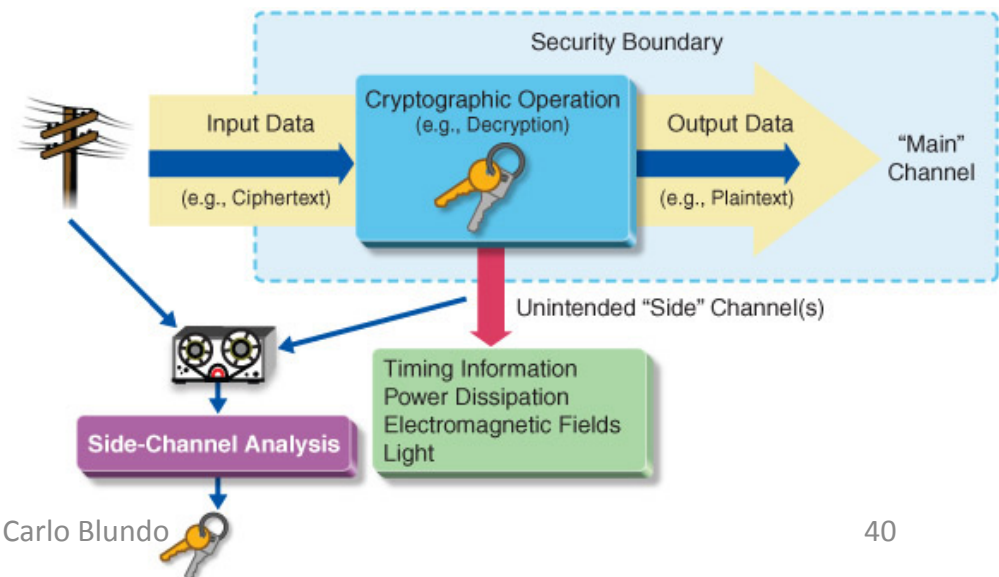
- Stesso esponente per chiavi diverse
 - Alice: $(3, n_1)$ – Bob: $(3, n_2)$ – Carol $(3, n_3)$
- Assumiamo $1 = \text{mcd}(n_i, n_j)$ per $i \neq j$
 - Altrimenti fattorizziamo n_i e n_j $m < \min\{n_1, n_2, n_3\}$
- Cifriamo lo stesso messaggio m con le tre chiavi
 - $c_1 = m^3 \bmod n_1$
 - $c_2 = m^3 \bmod n_2$
 - $c_3 = m^3 \bmod n_3$
- Settiamo $n = n_1 \cdot n_2 \cdot n_3$
- Per il Teorema Cinese dei Resti esiste un unico $c < n$ per cui
$$c = m^3 \bmod n$$
 - Dato che $m^3 < n$, si può calcolare m come $c^{1/3}$

L'attacco vale per qualsiasi e , ma dobbiamo avere e cifrature dello stesso messaggio

Side Channel Attack

- Attacchi all'implementazione che fruttano dispersioni fisiche dell'implementazione
 - Tempo impiegato ad eseguire un'operazione
 - Consumo di corrente (Power Attack)
 - Emissioni elettromagnetiche
 - Emissioni sonore

Non valgono solo ad RSA, ma si possono applicare a qualsiasi primitiva crittografica



Timing attack

- Per RSA, ricava i bit della chiave privata uno alla volta, analizzando il tempo richiesto per l'esponenziazione modulare (decifratura)
- L'attaccante tenta di compromettere un crittosistema analizzando il tempo necessario per eseguire le primitive crittografiche
- Attacco introdotto da Kocher nel 1996, contiene le idee di base della DPA

Paul C. Kocher

[Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems](#)

Advances in Cryptology — CRYPTO '96, LNCS Vol. 1109 pp. 104-113, 1996

Power Attack

- Introdotto nel 1998 da Kocher, Jaffe e Jun
- Ricava d analizzando la potenza consumata da una durante la decifratura
- Si sfrutta una debolezza del metodo square-and-multiply
 - Se un esponente è 0 si calcola solo uno quadrato (S)
 - Se un esponente è 1 si calcola un quadrato + una moltiplicazione (SM)

P. Kocher, J. Jaffe, B. Jun,

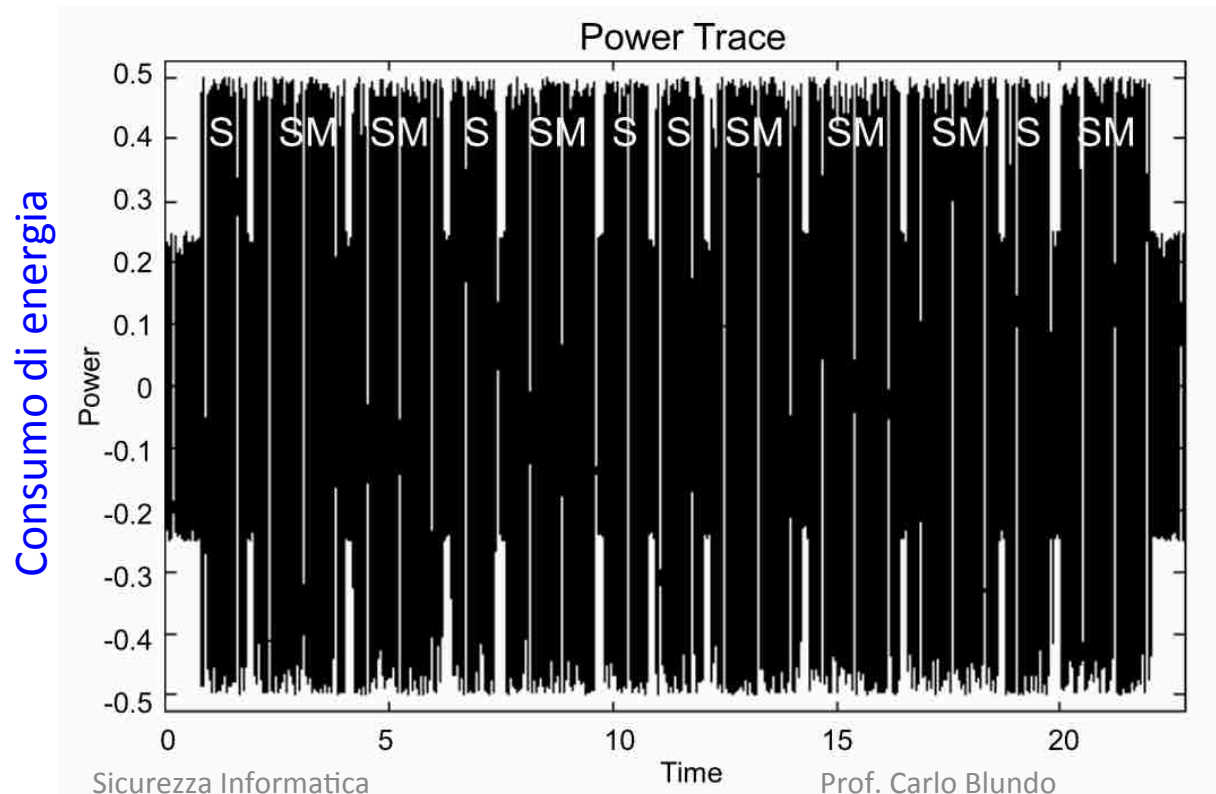
[Differential Power Analysis](#), Technical Report, 1998.

Successivamente Advances in Cryptology - Crypto '99, LNCS Vol. 1666, 1999.

Power Attack

- Se $d=1693$ $(011010011101)_2$, allora la traccia di square-and-multiply è la seguente

operations: *S SM SM S SM S S SM SM SM S SM*
private key: 0 1 1 0 1 0 0 1 1 1 0 1



La tecnica è chiamata
Simple Power Analysis

Contromisura

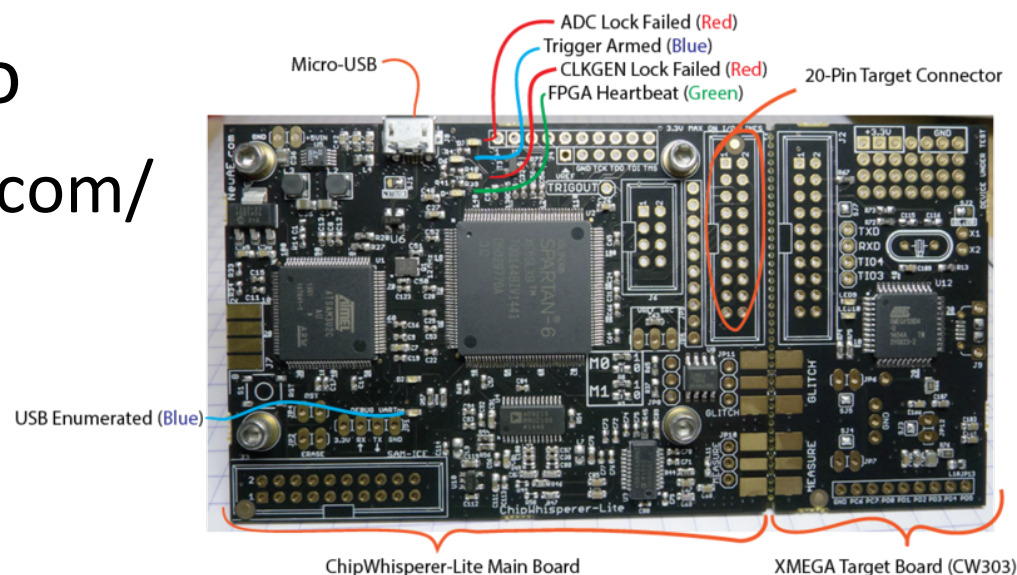
Fare in modo che le operazioni per 0 e per 1 consumino la stessa energia (moltiplicazione dummy)
Operazione meno efficiente

Power Attack

- Esistono altre tecniche più evolute
 - Differential power analysis (DPA)
 - Analisi statistiche per eliminare rumore presente
 - High-Order Differential Power Analysis (HO-DPA)
 - Si combinano segnali collezionati da sorgenti multiple con segnali collezionati usando tecniche di misura differenti
- Alcune contromisure sono protette da brevetto
- L'attacco è efficiente anche contro DES, AES e altre primitive crittografiche

Soluzioni Open-Source

- Esiste un insieme di strumenti di sviluppo che permette side-channel power analysis abbinato ad hardware per catturare i segnali
 - CW1173 ChipWhisperer-Lite
 - CW1200 ChipWhisperer-Pro
- Per chi è interessato
 - <https://wiki.newae.com/>



Fault-injection attack

- Utilizzare delle condizioni ambientali anormali per generare malfunzionamenti nel sistema. Si induce il dispositivo a commettere errori
 - Sbalzo di corrente
 - Errore indotto da radiazioni (e.g., luce ultravioletta)
 - Sfalsamento del clock
- Il sistema fornirà informazioni aggiuntive sulla sua computazione
 - Folklore: smart card inserita in un forno a microonde

Acoustic Cryptanalysis

La computazione fa rumore



- Si possono distinguere i tasti premuti su una tastiera dal tempo per premerli e dal rumore che fanno (utile per individuare una password)
 - Si usa una rete neurale per apprendere quale tasto sia stato premuto

Dmitri Asonov, Rakesh Agrawal

[Keyboard Acoustic Emanations](#)

IEEE Symposium on Security and Privacy, pp. 3-11, 2004

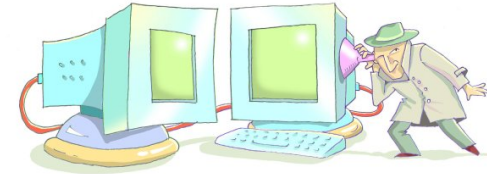
Li Zhuang, Feng Zhou, J. D. Tygar

[Keyboard acoustic emanations revisited](#)

ACM Transactions on Information and System Security, 2009.

Acoustic Cryptanalysis

La computazione fa rumore



- Si usa il suono generato dal computer (dalla CPU) durante la decifratura di alcuni testi cifrati (*Chosen CipherText Attack*)
 - L'attacco può essere condotto usando un telefonino posto vicino al computer oppure usando un microfono più sensibile posto anche a quattro metri di distanza

Adi Shamir and Eran Tromer

[Acoustic cryptanalysis, On nosy people and noisy machines](#)

Eurocrypt 2004 rump session, 2004

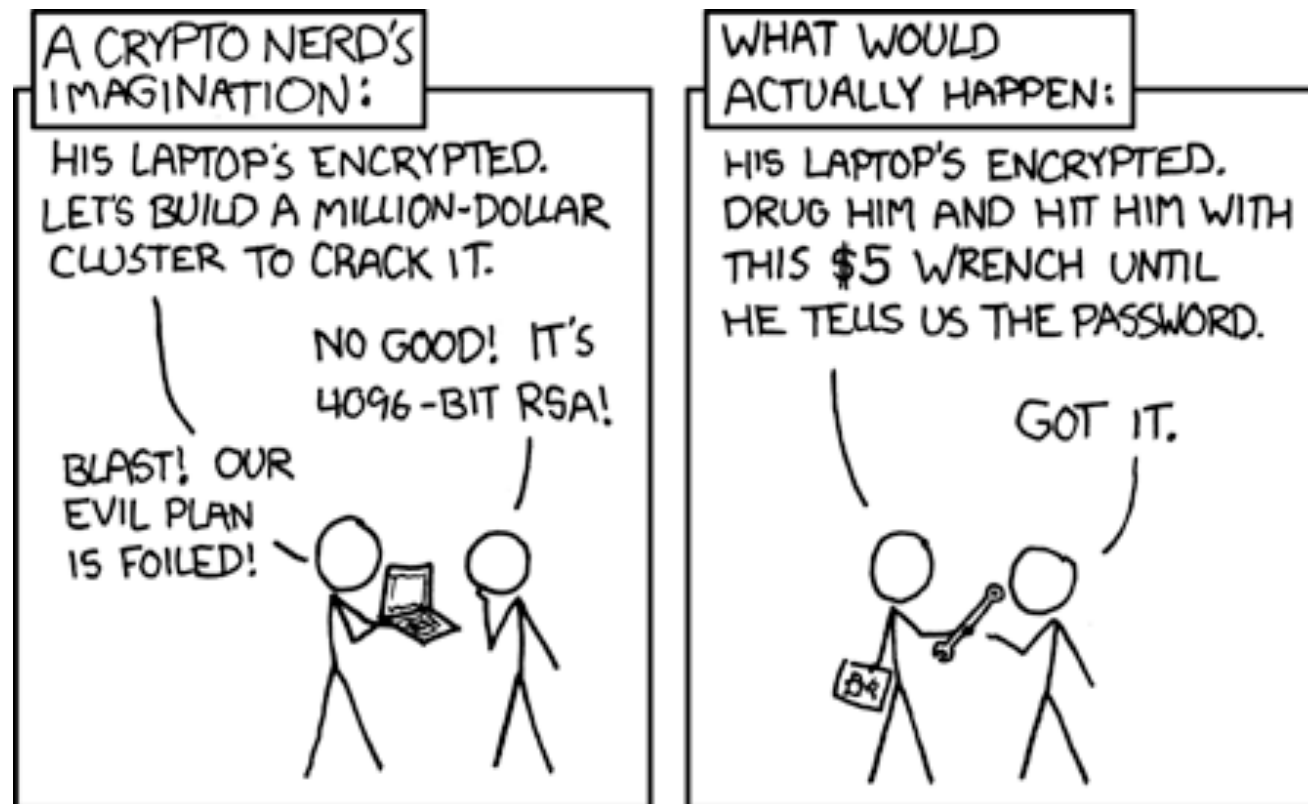
Daniel Genkin, Adi Shamir, and Eran Tromer

[Acoustic Cryptanalysis](#)

Journal of Cryptology, pp 1–52, 2016.

<http://cs.tau.ac.il/~tromer/acoustic/>

Rubber-hose cryptanalysis



Textbook RSA

- Lo schema RSA che abbiamo visto nelle slide precedenti è chiamato **Textbook RSA**
 - In pratica RSA non viene utilizzato come descritto
- Si può provare facilmente che Textbook RSA non è sicuro secondo i modelli di sicurezza che vedremo in seguito
 - Dobbiamo modificare il formato dei messaggi
 - Dobbiamo aggiungere randomness alla cifratura
- Lo schema modificato è sicuro anche contro gli attacchi presentati nelle slide precedenti

Modelli di Sicurezza – scenario

- I modelli di sicurezza si descrivono attraverso un esperimento (gioco) tra un avversario (**Adversary**) ed uno sfidante (**Challenger**)
- Spesso si fa riferimento ad un oracolo
 - Lo si può considerare come una funzione che ad un input associa un output
 - Non si sa come funziona l'oracolo
 - Oracolo di cifratura: cifra conoscendo la chiave pubblica
 - Ad identici input corrispondono identici output

Avversario PPTA

- Si assume che l'avversario possa essere rappresentato da un *Probabilistic Polynomial Time Algorithm* (PPTA)
- Un algoritmo polinomiale A è un algoritmo per cui esiste un polinomio $p(\cdot)$ tale che il tempo di esecuzione di A con input $x \in \{0,1\}^*$ è al più $p(|x|)$
- Un algoritmo probabilistico ha anche la capacità di generare bit casuali da utilizzare durante la sua computazione. A può generare solo un numero polinomiale di bit casuali

Funzione negligibile

- Una funzione $\mu(k) : \mathbb{N} \rightarrow \mathbb{R}$ è negligibile in k se per ogni polinomio positivo $\text{poly}(\cdot)$ esiste un intero $N_{\text{poly}} > 0$ tale che per ogni $k > N_{\text{poly}}$ si ha

$$\mu(k) < \frac{1}{\text{poly}(k)}$$

il parametro k rappresenta il parametro di sicurezza

Tipi di Modelli di Sicurezza

- IND-CPA
 - INDistinguishability under CPA
- IND-CCA₁
 - INDistinguishability under Chosen Ciphertext Attack 1
 - Non Adaptive CCA
- IND-CCA₂
 - INDistinguishability under Chosen Ciphertext Attack 2
 - Adaptive CCA

Possibili definizioni di sicurezza

- Dati pk e $Enc_{pk}(m)$, non è fattibile calcolare m
 - Troppo debole
 - L'avversario potrebbe ottenere qualche informazione sul messaggio m
- Dati pk e $Enc_{pk}(m)$ non è fattibile calcolare qualsiasi informazione su di m
 - **OK**
 - La definiamo formalmente tramite un esperimento

Esperimento IND-CPA

per un crittosistema a chiave pubblica $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$

CPA indistinguishability experiment $\text{PubK}_{\mathcal{A}, \Pi}^{\text{cpa}}(n)$

1. $\text{Gen}(1^n)$ is run to obtain keys (pk, sk) .
2. Adversary \mathcal{A} is given pk , and outputs a pair of messages m_0, m_1 of the same length. (These messages must be in the plaintext space associated with pk .)
3. A random bit $b \leftarrow \{0, 1\}$ is chosen, and then a ciphertext $c \leftarrow \text{Enc}_{pk}(m_b)$ is computed and given to \mathcal{A} . We call c the challenge ciphertext.
4. \mathcal{A} outputs a bit b' .
5. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.

Challenger

\mathcal{A} prova a determinare quale messaggio sia stato cifrato

L'avversario può cifrare un numero polinomiale di messaggi a sua scelta prima e dopo il passo 3

Sicurezza IND-CPA

- Uno schema $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ a chiave pubblica ha cifrature indistinguibili sotto un attacco con scelta dei testi in chiaro (CPA security) se per ogni avversario \mathcal{A} PPTA esiste una funzione trascurabile **negl** tale che

$$\Pr[\text{PubK}_{\mathcal{A}, \Pi}^{\text{cpa}}(n) = 1] \leq \frac{1}{2} + \text{negl}(n).$$

L'avversario non è avvantaggiato dalla conoscenza di cifrature di messaggi a sua scelta

Permettiamo che l'avversario possa determinare il messaggio cifrato con una probabilità maggiore di $1/2$ per un valore trascurabile

Il valore della funzione **negl** possiamo limitarlo scegliendo opportunamente il parametro di sicurezza

Textbook RSA è IND-CPA?

- No!!! Facile da provare
- L'**avversario** sceglie i messaggi m_0 ed m_1
- Il **challenger** ne cifra uno dei due m_b ed invia la cifratura c all'avversario
- L'**avversario**, conoscendo la chiave pubblica, calcola $c' = \text{Enc}_{pk}(m_0)$
 - se $c' = c$, il challenger ha cifrato m_0
 - se $c' \neq c$, il challenger ha cifrato m_1

Randomness necessaria

- Qualsiasi schema a chiave pubblica deterministico non è IND-CPA sicuro
- Molti schemi utilizzati in pratica furono progettati usando schemi a chiave pubblica deterministici
- Il lavoro fondamentale di Micali e Goldwasser è stato un punto di svolta nella ricerca in crittografia

Vincitori del Turing Award 2012

Silvio Micali and Shafi Goldwasser

[Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information.](#)
STOC 1982

[Probabilistic Encryption.](#) Journal of Computer and System Science, 1984

Scenario (plausibile?)

- In un'azienda i messaggi inviati ad dipendenti sono cifrati con la chiave pubblica della compagnia
- I dipendenti per decifrare i messaggi usano una macchina speciale
 - Memorizzano il messaggio su una penna USB e la inseriscono nella macchina
 - La macchina si accorge se lo stesso messaggio cifrato è stato già sottoposto a decifratura
- Un avversario vuole decifrare un messaggio non indirizzato a lui che è già stato decifrato dalla macchina
 - Se la macchina usa RSA, l'avversario può decifrare

Malleabilità

- Uno schema di cifratura è malleabile se è possibile per un avversario trasformare un testo cifrato c in un altro testo cifrato c' che una volta decifrato fornisce un testo in chiaro relato a quello di c .
- In altre parole, dato una cifratura di un testo in chiaro m è possibile generare un altro testo cifrato che una volta decifrato fornisce $f(m)$ per una funzione f nota senza necessariamente conoscere m .

Malleabilità di RSA

- L'attaccante è in grado di manipolare il testo in chiaro (usando il corrispondente testo cifrato) in maniera predicibile
 - Ad esempio, raddoppiamo il valore *contenuto* nel messaggio cifrato
 - Se $c_1 = m_1^e \bmod n$ e $c_2 = m_2^e \bmod n$, allora
$$c_1 c_2 = m_1^e \cdot m_2^e \bmod n = (m_1 \cdot m_2)^e \bmod n$$
- Per evitare tale problema si usa RSA OAEP oppure PKCS#1

Attacco (CCA) all'azienda

Obiettivo: decifrare $c = m^e \bmod n$

Sceglie a caso x

Calcola $x^e \bmod n$

Invia $c' = c \cdot x^e \bmod n$



$$c' = c \cdot x^e \bmod n$$

$$(c')^d \bmod n$$



c' non è stato già decifrato

L'avversario può calcolare m

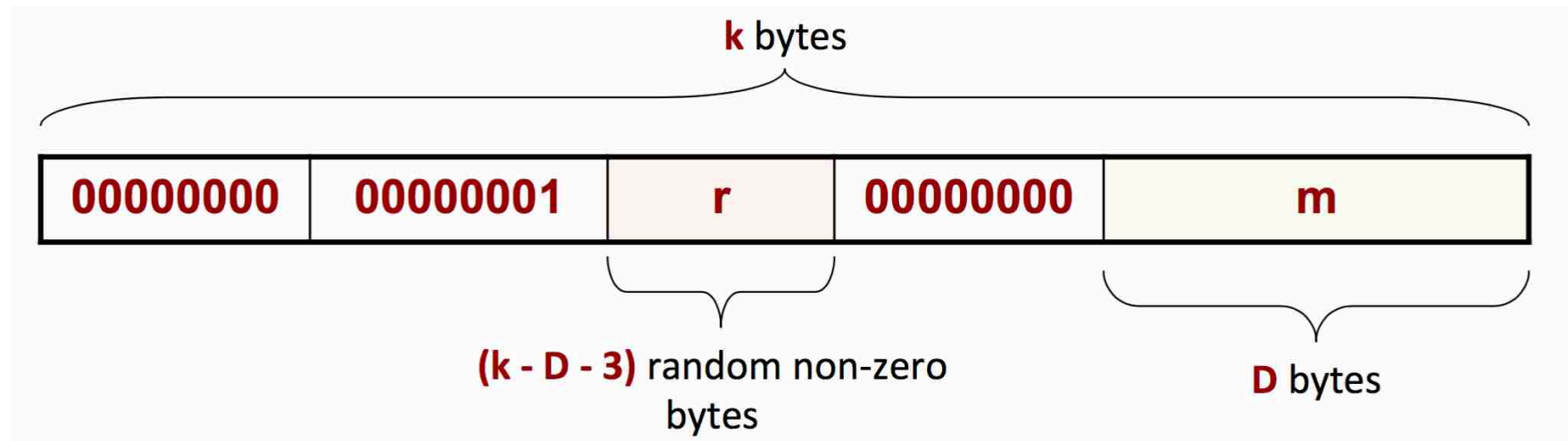
$$\begin{aligned} x^{-1} \cdot (c')^d \bmod n &= x^{-1} \cdot (c \cdot x^e)^d \bmod n \\ &= x^{-1} \cdot c^d \cdot x^{e \cdot d} \bmod n \\ &= x^{-1} \cdot (m^e)^d \cdot x \bmod n \\ &= (m^e)^d \bmod n \\ &= m \end{aligned}$$

IND-CCA (1 e 2)

- Cambiamo l'esperimento
- L'avversario può decifrare testi cifrati a scelta
- Non ha la chiave privata, ma ha accesso ad un oracolo di decifratura
- Può chiedere un numero polinomiale di decifrate e poi tentare di dedurre il messaggio contenuto in un testo cifrato **c** fornito dal challenger
- Se dopo aver ricevuto **c** può continuare a chiedere decifrate, siamo nel modello 2

Paranoia?

- Bleichenbacher ha mostrato un attacco di tipo chosen ciphertext (CCA) contro lo standard PKCS#1 v1.5



Daniel Bleichenbacher

[Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1](#)

Advances in Cryptology — CRYPTO '98, Vol.1462 of LNCS, pp 1-12, 1998.

Esperimento IND-CCA₂

per un crittosistema a chiave pubblica $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$

CCA₂ indistinguishability experiment $\text{PubK}_{A, \Pi}^{\text{cca}_2}(n)$:

1. $\text{Gen}(1^n)$ is run to obtain keys (pk, sk) .
2. The adversary \mathcal{A} is given pk and access to a decryption oracle $\text{Dec}_{sk}(\cdot)$. It outputs a pair of messages m_0, m_1 of the same length. (These messages must be in the plaintext space associated with pk .)
3. A random bit $b \leftarrow \{0, 1\}$ is chosen, and then a ciphertext $c \leftarrow \text{Enc}_{pk}(m_b)$ is computed and given to \mathcal{A} .
4. \mathcal{A} continues to interact with the decryption oracle, but may not request a decryption of c itself. Finally, \mathcal{A} outputs a bit b' .
5. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.

Challenge
Phase

L'avversario può cifrare un numero polinomiale di testi in chiaro

Non c'è nell'esperimento IND-CCA₁

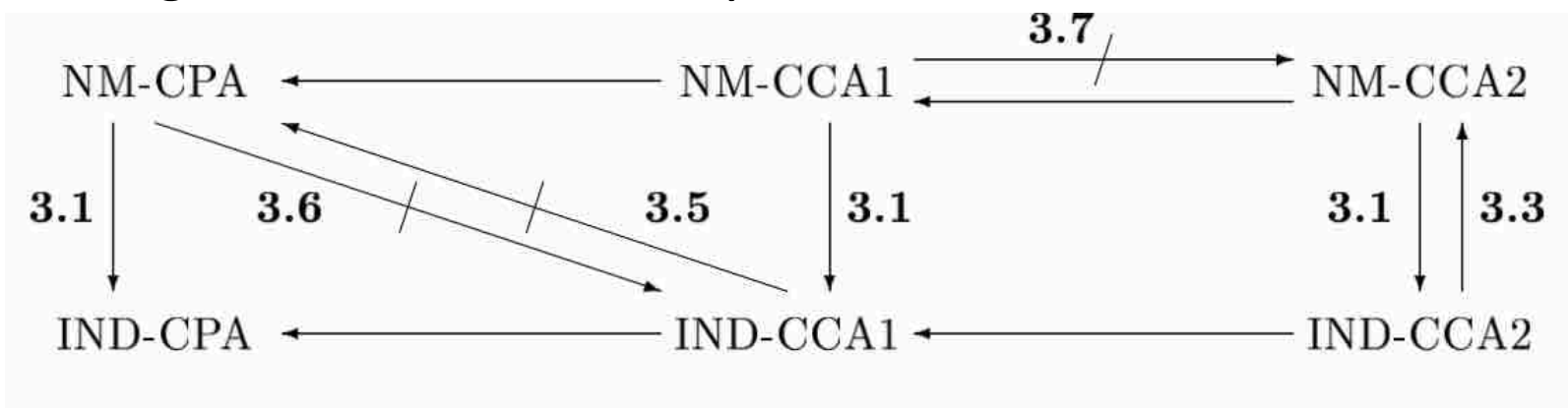
Sicurezza IND-CCA₂

- Uno schema $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ a chiave pubblica ha cifrature indistinguibili sotto un attacco con scelta dei testi cifrati (CCA security) se per ogni avversario \mathcal{A} PPTA esiste una funzione trascurabile **negl** tale che

$$\Pr[\text{PubK}_{\mathcal{A}, \Pi}^{\text{cca}_2}(n) = 1] \leq \frac{1}{2} + \text{negl}(n)$$

Relazioni tra le definizioni

NM significa non malleability



La freccia barrata indica che l'implicazione non è soddisfatta

3.x teorema dove è stata provata l'implicazione

M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway

[Relations among Notions of Security for Public-Key Encryption Schemes](#)

CRYPTO '98, LNCS vol. 1462, pp. 26-45, 1998

Textbook RSA è IND-CCA₁?

- Avendo provato che RSA non è IND-CPA, secondo le relazioni della slide precedente RSA non può essere né IND-CCA₁ né IND-CCA₂
- In ogni caso, l'avversario conosce la chiave pubblica, può sempre cifrare uno dei due messaggi m_0 e m_1 e confrontare il risultato con la sfida c ricevuta dal Challenger Oracle

Soluzioni

- Il messaggio da cifrare deve essere modificato opportunamente tramite padding
- Si potrebbe usare PKCS#1
 - La versione 1.5 è stata rotta da Bleichenbacher
 - Adesso c'è la versione 2.2 del 27 ottobre 2012
- Meglio usare OAEP RSA
 - Optimal Asymmetric Encryption Padding
 - PKCS#1 v2 e RFC 2437

M. Bellare e P. Rogaway.

[Optimal Asymmetric Encryption - How to encrypt with RSA](#)
EUROCRYPT '94, LNCS Vol. 950, 1995.

PKCS#1

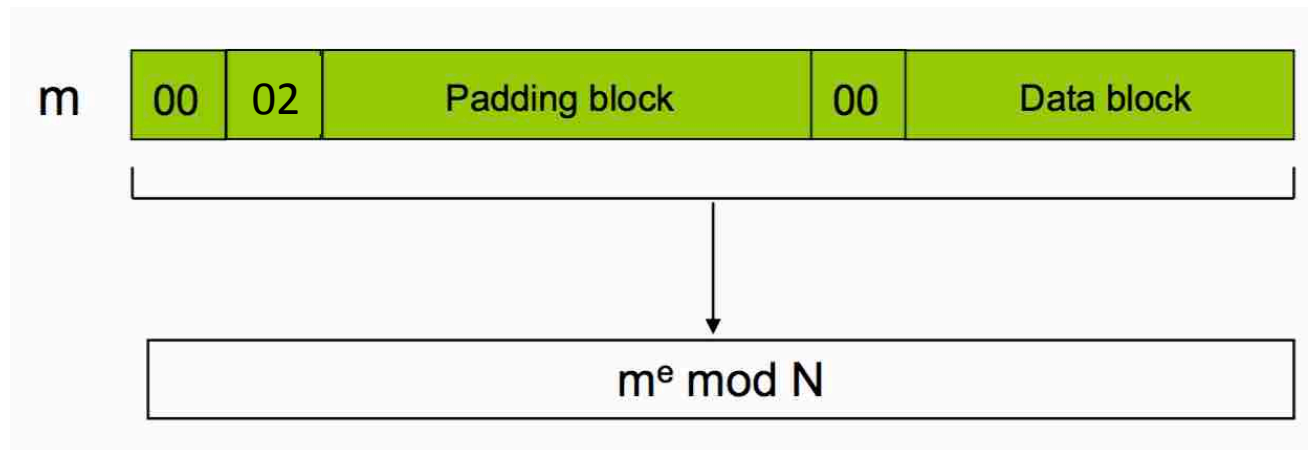
- PKCS#1 definisce (tra le altre cose) come usare RSA per cifrare
 - Version 1.5: novembre 1993, anche RFC 2313
 - Version 2.0: ottobre 1998, anche RFC 2427
 - Version 2.1: febbraio 2003, anche RFC 3447
 - Version 2.2: ottobre 2012
- PKCS#1 è usato, ad esempio, nel Protocollo di Handshake di SSL/TLS

PKCS#1v1.5 (nov. 1993)

- Assumiamo che il modulo n di una chiave RSA (e, N) sia grande k byte $2^{8(k-1)} \leq N \leq 2^{8k}$
- Il messaggio da cifrare è modificato aggiungendo alcuni byte di padding con un formato prestabilito
 - Non possiamo cifrare un messaggio di k byte
- Lo standard prevede tre formati
 - 00 e 01 usati per messaggi da firmare digitalmente
 - 02 usato per messaggi da cifrare

00, 01 e 02 espressi in esadecimale

Formato PKCS#1v1.5



00 e 02 espressi in esadecimale

- 00 iniziale, byte che garantisce che $m < N$
- 02 byte indica che il formato è per cifrare
- **Padding block** stringa casuale di almeno 8 byte (diversi da 00)
- 00 byte che indica la fine del **Padding block**
- **Data block** messaggio da cifrare più piccolo di $k-11$ byte
- m messaggio che sarà cifrato con PKCS#1v1.5

Esempio con RSA-256

- N è di 32 byte (256 bit)
- Vogliamo cifrare la chiave AES di 128 bit
 - La rappresentiamo in esadecimale
4E636AF98E40F3ADCFCCB698F4E80B9F
- Il messaggio da cifrare in formato PKCS#1v1.5 potrebbe essere il seguente (dipende dal padding block che è casuale)

0002257F48FD1F1793B7E5E02306F2D3228F5C95ADF5F31566729F132AA12009
E3FC9B2B475CD6944EF191E3F59545E671E474B555799FE3756099F044964038
B16B2148E9A2F9C6F44BB5C52E3C6C8061CF694145FAFDB24402AD1819EACEDF
4A36C6E4D2CD8FC1D62E5A1268F496**004E636AF98E40F3ADCFCCB698F4E80B9F**

Attacco di Bleichenbacher

- Supponiamo di avere accesso ad un oracolo che con input c restituisce in output **true** se $x = c^d \bmod N$ è formattato correttamente con PKCS#1v1.5, **false** altrimenti
 - x inizia con 00 02 oppure no
- Se otteniamo **true**, allora vale la seguente disuguaglianza $2 \cdot 2^{8(k-2)} \leq x \bmod N < 3 \cdot 2^{8(k-2)}$
 - k è il numero di byte del modulo N

Idea dell'attacco

- Supponiamo che $c^* = m^e \bmod N$ è il target ciphertext che cifra un messaggio m ignoto
- Invocheremo l'oracolo *molte volte* su input scegli opportunamente della forma $s^e c^* \bmod N$
 - s è scelto in maniera opportuna
- Ogni risposta true da parte dell'oracolo fornirà la disuguaglianza $2 \cdot 2^{8(k-2)} \leq s \cdot m \bmod N < 3 \cdot 2^{8(k-2)}$
- Analizzando le risposte l'attaccante recupera m

L'attacco ha successo?

- In genere sono necessari circa 2^{20} messaggi cifrati scelti dall'attaccante
 - Il numero dipende da molti dettagli che dipendono dall'implementazione
- Questo in teoria, ma in pratica esiste l'oracolo?
 - Può esistere a causa di messaggi di errore che restituiscono applicazioni che usano PKCS#1v1.5
 - Ad esempio, SSLv3.0 usava un formato leggermente differente ma che iniziava sempre con 00 02
 - Nel 1998 circa 2 su 3 server SSL testati rispondevano con messaggi di errore utili per l'attaccante

Conclusioni e contromisure

- PKCS1#v1.5 non è CCA sicuro
- **Non fornire informazioni tramite messaggi di errore**
 - Allevia il problema, ma non lo risolve. Comunque, PKCS1#v1.5 resta insicuro contro CCA
- **Usare uno schema di cifratura IND-CCA sicuro**
 - Ad esempio RSA-OAEP (PKCS#1v2)

OAEP

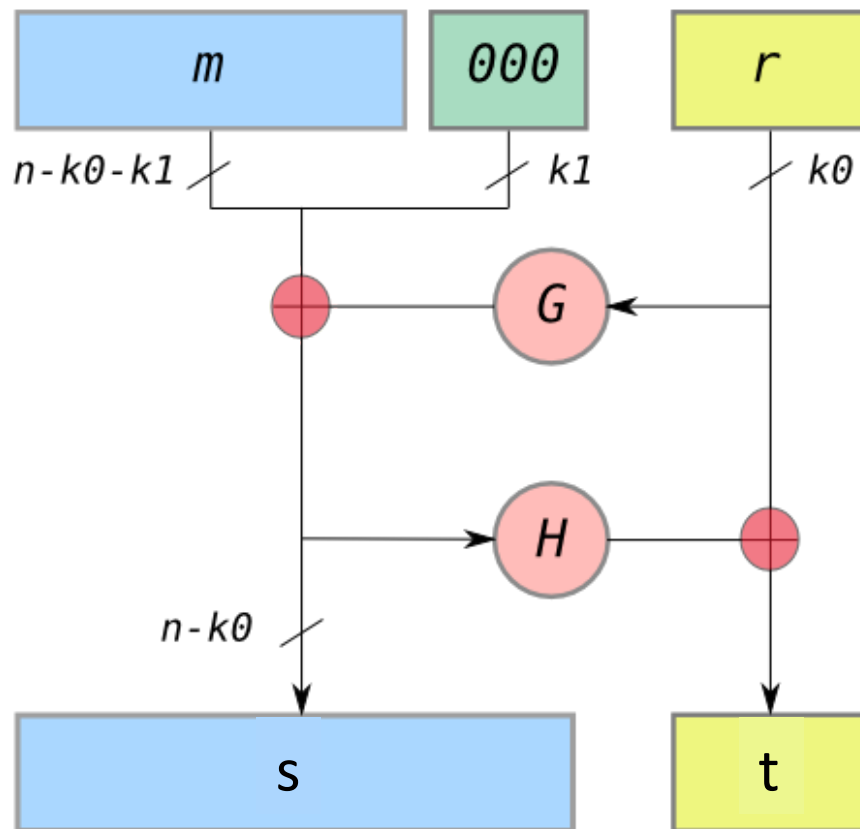
Bellare&Rogaway

- Nel 1995 Bellare e Rogaway introdussero una nuova nozione di cifratura **plaintext awareness**
- Uno schema di cifratura è **plaintext aware** se è impossibile per un attaccante produrre un valido testo cifrato senza conoscere il corrispondente testo in chiaro
- RSA non è **plaintext aware** perché è malleabile

$$\text{Enc}_{pk}^{H,G}(x) = f(x0^{k_1} \oplus G(r) \parallel rH(x0^{k_1} \oplus G(r)))$$

f trapdoor permutation, H e g funzioni hash

OAEP



H e G sono funzioni hash
 \oplus è lo xor

Padding

$$s = (m || 0 \dots 0) \oplus G(r)$$

$$t = H(s) \oplus r$$

Cifratura

$$c = \text{Enc}_{pk}(s || t)$$

Decifratura

$$(s || t) = d = \text{Dec}_{sk}(c)$$

Padding

$$r = H(s) \oplus t$$

$$m || 0 \dots 0 = s \oplus G(r)$$

Problemi?

- Cosa succede se $x = (s \parallel t)$ rappresenta un intero **più grande** del modulo N usato in RSA?
- Soluzione in BR94
 - Si sceglie della nuova randomness r fino a quando risulta che x rappresenta un intero minore di N
- RSA-OAEP descritta in PKCS#1v2 utilizza una soluzione leggermente differente
 - Si modifica la codifica OAEP in modo che l'output della codifica sia un byte più piccolo del numero necessario per rappresentare N
 - Setta il primo byte della codifica a 00

EME – Encoding Method for Encryption

EME-OAEP (PKCS# v2)

L: etichetta associata al messaggio, è opzionale (può essere la stringa vuota)

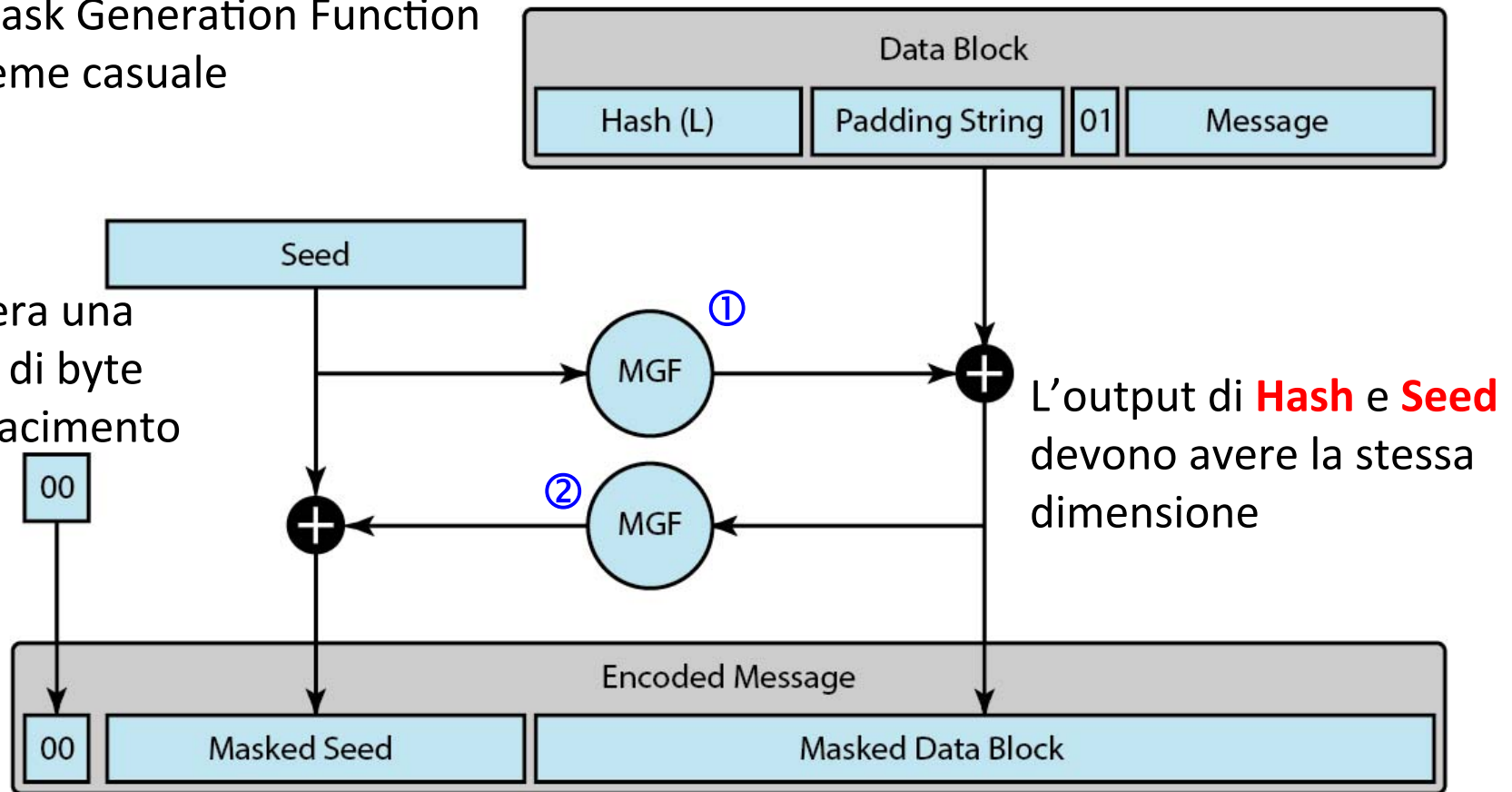
Padding String: padding di byte uguali a 0 (0x00), può essere nulla

Hash: funzione hash

MGF: Mask Generation Function

Seed: Seme casuale

MGF genera una maschera di byte lunga a piacimento



EME-OAEP (PKCS# v2)

- Assumiamo che il testo cifrato è lungo k byte (octet) e che l'output di Hash sia lungo $hLen$ byte (octet)
- La lunghezza $mLen$ del messaggio da cifrare potrà essere al più $k - 2 \cdot hLen - 2$ byte (octet)
- La lunghezza della Padding String è
 $k - mLen - 2 \cdot hLen - 2$ byte (octet)
- Output di MGF
 - Al punto ① slide precedente: $k - hLen - 1$ byte (octet)
 - Al punto ② slide precedente: $hLen$ byte (octet)

La Mask Generation Function

funzione deterministica

```
MGF(mgfSeed, maskLen)
```

```
Mask = ""
```

```
// hLen lunghezza output funzione Hash
```

```
C = maskLen/hLen //approssimato all'intero superiore
```

```
for(i=0 ; i<C; i++)
```

```
Mask = Mask || Hash(mgfSeed || i)
```

```
return I primi maskLen byte (octet) di Mask
```

Conclusioni

- Gli attacchi hanno successo contro schemi di cui non è stata provata la sicurezza
 - PKCS#1v1.5
- Lo standard PKCS#1 è stato revisionato per includere RSA-OAEP che è CCA sicuro
 - Ci sono attacchi contro piccole modifiche dello schema sicuro
 - Gli attacchi potrebbero dipendere dall'implementazione
 - Fare attenzione ai timing attack
 - TLS 1.2 (2008) ancora usa PKCS#1v1.5

TLS 1.3 è ancora in forma di bozza

SSL/TLS

- SSL (Secure Sockets Layer)
- TLS (Transport Layer Security)
 - sono protocolli che permettono la cifratura dei dati e l'autenticazione tra applicazioni e server in scenari dove i dati sono inviati attraverso una rete insicura
- SSL e TLS sono spesso utilizzati in congiunzione oppure intercambiabilmente. In realtà SSL è il predecessore di TLS
 - SSL 3.0 è servito come base di TLS 1.0 (a cui, a volte) si fa riferimento come SSL 3.1

TLS 1.2, RFC 5246 (2008)

- Estratto da RFC 5246

The RSAES-OAEP encryption scheme defined in [PKCS1] is more secure against the Bleichenbacher attack. However, for maximal compatibility with earlier versions of TLS, **this specification uses the RSAES-PKCS1-v1_5 scheme**. No variants of the Bleichenbacher attack are known to exist provided that the above recommendations are followed.

above recommendations

In any case, a TLS **server MUST NOT generate an alert** if processing an RSA-encrypted premaster secret message fails, or the version number is not as expected. Instead, it **MUST** continue the handshake with a randomly generated premaster secret. It may be useful to log the real cause of failure for troubleshooting purposes; however, care must be taken to avoid leaking the information to an attacker (through, e.g., timing, log files, or other channels.)

Il server non deve fare da oracolo

I parametri RSA

Scelta dei parametri

- È necessario scegliere i parametri RSA in maniera oculata in modo da evitare gli attacchi esaminati nelle slide precedenti
 - Quanto deve essere grande n ?
 - Che *forma* non devono assumere i parametri p e q ?
 - Come scegliere la chiave pubblica e e quella privata d ? (solo per textbook RSA)

Key Size

Gennaio 2016

- Quanto deve essere grande n ?
- Secondo NIST SP 800-57-P1-R4
 - IFC: Integer Factorization Cryptography
 - k è la dimensione in bit della chiave
 - I parametri con bit security 192 e 256 non sono inclusi negli attuali standard NIST per motivi di interoperabilità ed efficienza

Security Strength	IFC (e.g., RSA)
≤ 80	$k = 1024$
112	$k = 2048$
128	$k = 3072$
192	$k = 7680$
256	$k = 15360$

Con le conoscenze attuali, per fattorizzare un numero di 3072 bit ci vuole un tempo proporzionale a 2^{128}

Equivalenza tra Key Size

- Come si stabilisce l'equivalenza (in termini di sicurezza) tra le lunghezze delle chiavi per cifrari simmetrici e a chiave pubblica?
- Ci sono vari metodi, uno di quelli utilizzati consiste nel risolvere l'equazione
$$2^k = \text{complessità GNFS}(N) \quad (L_N[1/3, 1,923])$$
 - k security strength
 - $N = O(2^n)$ fattorizzare numeri di n bit

Calcoliamo k

- Poniamo $L(N) = L_N[1/3, C]$. Nell'espressione

$$L(N) = Ae^{(C+o(1))(\ln N)^{1/3}(\ln \ln N)^{2/3}}$$

si assume $o(1)=0$, si considera $C=(64/9)^{1/3}$ e a partire da una stima di $L(n_{512})$ si calcola A

- $L(n_{512})$ è la complessità della fattorizzazione di un modulo RSA di 512 bit (Indicato con RSA-512)
 - Dall'esperienza si deduce che la resistenza RSA-512 di è circa 4 o 6 bit in meno di DES-56. Per una stima conservativa si assume pari a 50
- La security strength $s(n)$ di un modulo RSA N di n bit è

$$s(n) = \left(\frac{64}{9}\right)^{1/3} \log_2(e) (n \ln 2)^{1/3} (\ln(n \ln 2))^{2/3} - 14.$$

Fonti indipendenti

- ECRYPT II,
Yearly Report on Algorithms and Keysizes
(2011-2012), settembre 2012
 - Inglobato in Algorithms, key size and parameters report – 2014 dell'ENISA (European Union Agency for Network and Information Security)
- BSI TR-02102-1, Kryptographische Verfahren:
Empfehlungen und Schlüssellängen, febbraio
2016
 - Bundesamt für Sicherheit in der Informationstechnik
 - Ufficio Federale Tedesco per la Sicurezza Informatica

Indicazioni ECRYPT II

Table 7.3: Effective Key-size of Commonly used RSA/DLOG Keys.

RSA/DLOG Key	Security (bits)
512	50
768	62
1024	73
1536	89
2048	103




Table 7.2: Key-size Equivalence.

Security (bits)	RSA	DLOG		EC
		field size	subfield	
48	480	480	96	96
56	640	640	112	112
64	816	816	128	128
80	1248	1248	160	160
112	2432	2432	224	224
128	3248	3248	256	256
160	5312	5312	320	320
192	7936	7936	384	384
256	15424	15424	512	512

Date	Symmetric	Factoring Modulus	Discrete Key	Logarithm Group	Elliptic Curve	Hash	
2015	128	2048	224	2048	224	SHA-224 SHA-256 SHA-512/256	SHA-384 SHA-512 SHA-512/224
2016	128	2048	256	2048	256	SHA-256 SHA-384	SHA-512 SHA-512/256
2017 - 2021	128	3072 (*)	256	3072 (*)	256	SHA-256 SHA-384	SHA-512 SHA-512/256

BSI

<https://www.keylength.com/>

Method	Date	Symmetric	Factoring Modulus	Discrete Key	Logarithm Group	Elliptic Curve	Hash
[1] Lenstra / Verheul 	2016	83	1664 1312	146	1664	155	165
[2] Lenstra Updated 	2016	79	1273 1392	158	1273	158	158
[3] ECRYPT II	2016 - 2020	96	1776	192	1776	192	192
[4] NIST	2011 - 2030	112	2048	224	2048	224	224
[5] ANSSI	2014 - 2020	100	2048	200	2048	200	200
[6] NSA	-	-	-	-	-	-	-
[7] RFC3766 	-	-	-	-	-	-	-
[8] BSI	2016	128	2048	256	2048	256	256

Link utile

- <https://www.keylength.com/>
- Sito web che riassume le indicazioni, di note organizzazioni (NIST, ANSI, BSI, ...), sulla dimensione delle chiavi di primitive crittografiche
- Permette di confrontare le varie indicazioni

Come scegliere p e q

- $|p| \approx |q|$
 - Per evitare l'attacco di Lenstra (Elliptic Curve Method)
- $p-q$ deve essere grande
 - Per evitare l'attacco *trial division* (si parte da $n^{1/2}$)
- $p-1$ deve avere almeno un fattore primo grande
 - Per evitare l'attacco p di Pollard
- $p+1$ deve avere almeno un fattore primo grande
 - Per evitare l'attacco di Williams

p e q devono essere *strong prime*

- Un numero p è strong prime se
- p è un numero primo grande
- $p-1$ ha almeno un fattore primo grande
 - $p = a_1 p_1 + 1$, dove p_1 è un primo grande
- p_1-1 ha almeno un fattore primo grande
 - $p_1 = a_2 p_2 + 1$, dove p_2 è un primo grande
- $p+1$ ha almeno un fattore primo grande
 - $p = a_3 p_3 - 1$, dove p_2 è un primo grande

Prestazioni di RSA

- Implementazioni in hardware di RSA sono circa 1000 volte più lente di DES e AES, implementazioni in software sono circa 100 volte più lente (modulo di almeno 512 bit)

Modulus [bit]	Dec [ms]	Enc ($e = 2^{16} + 1$) [ms]
1024	0.299	0.019
2048	2.021	0.062
3072	25.1	12.2
7680	113.3	14.1
15360	720.2	20.4

Prestazioni di OpenSSL su un
processore AMD
Opteron™ Processor-8378, 2.4Ghz

È stato usato solo un core

Riferimenti

Christof Paar and Jan Pelzl
Understanding Cryptography
Capitolo 7 **The RSA Cryptosystem**

William Stallings
Cryptography and Network Security: Principles and Practice (6th Ed)
Capitolo 9, Paragrafo **The Security of RSA**

Jonathan Katz and Yehuda Lindell
Introduction to Modern Cryptography (1st edition)
Capitolo 10, Paragrafi 10.1, 10.2 (tranne 10.2.2) e 10.6