Cryptography and Network Security

Public Key Cryptography and RSA

Outline

- ✓ Principi della criptografia a chiave pubblica
- ✓ Algoritmo RSA, implementazione e sicurezza
- ✓ Attacchi all'algoritmo RSA

Criptografia a chiave privata

- La criptografia tradizionale con chiave privata (singola/segreta) utilizza una sola chiave
- La chiave è condivisa da sender e receiver
- Se questa chiave viene conosciuta da un opponent tutte le comunicazioni sono compromesse
- La criptografia è simmetrica (le due parti si trovano nella stessa condizione)
- Sistema non scalabile perché il numero delle chiavi cresce con il quadrato del numero n degli utilizzatori $[n(n\text{-}1)/2]$
- Il sender non è protetto dal fatto che il receiver costruisca costruisca un messaggio falso e pretenda che sia stato inviato dal sender

Criptografia a chiave publica

- Avanzamento più significativo nella lunga storia della criptografia
- Usa due chiavi: pubblica e privata
- Sistema asimmetrico giacché le due estremità non si trovano nella stessa condizione
- Fondata sull'applicazione intelligente di concetti propri della teoria dei numeri
- È un complemento e non una sostituzione della criptografia a chiave privata

Criptografia a chiave pubblica

- Detta criptografia a chiave pubblica/a due chiavi/asimmetrica
- In ogni modo comporta l'uso di due chiavi:
  - Una chiave pubblica, che può essere conosciuta da tutti che può essere usata per criptare messaggi e per verifica di firme
  - Una chiave privata, nota soltanto al recipient, che può essere usata per decriptare messaggi e per firmare dei documenti
- È un sistema asimmetrico perché colui che cripta i messaggi o firma un messaggio usa una chiave diversa da quella usata da colui che riceve il messaggio

Criptografia a chiave pubblica

Sistema con sei elementi

Encryption di un messaggio

Perchè?

Due importanti problemi da risolvere

Distribuzione delle chiavi

Come avere delle comunicazioni sicure senza dovere ottenere le chiavi da un KDC fiduciale

Firma digitale

Come verificare che un messaggio provenga effettivamente da chi se ne dichiara autore

Invenzione da parte di W. Diffie & M. Hellman nel 1976

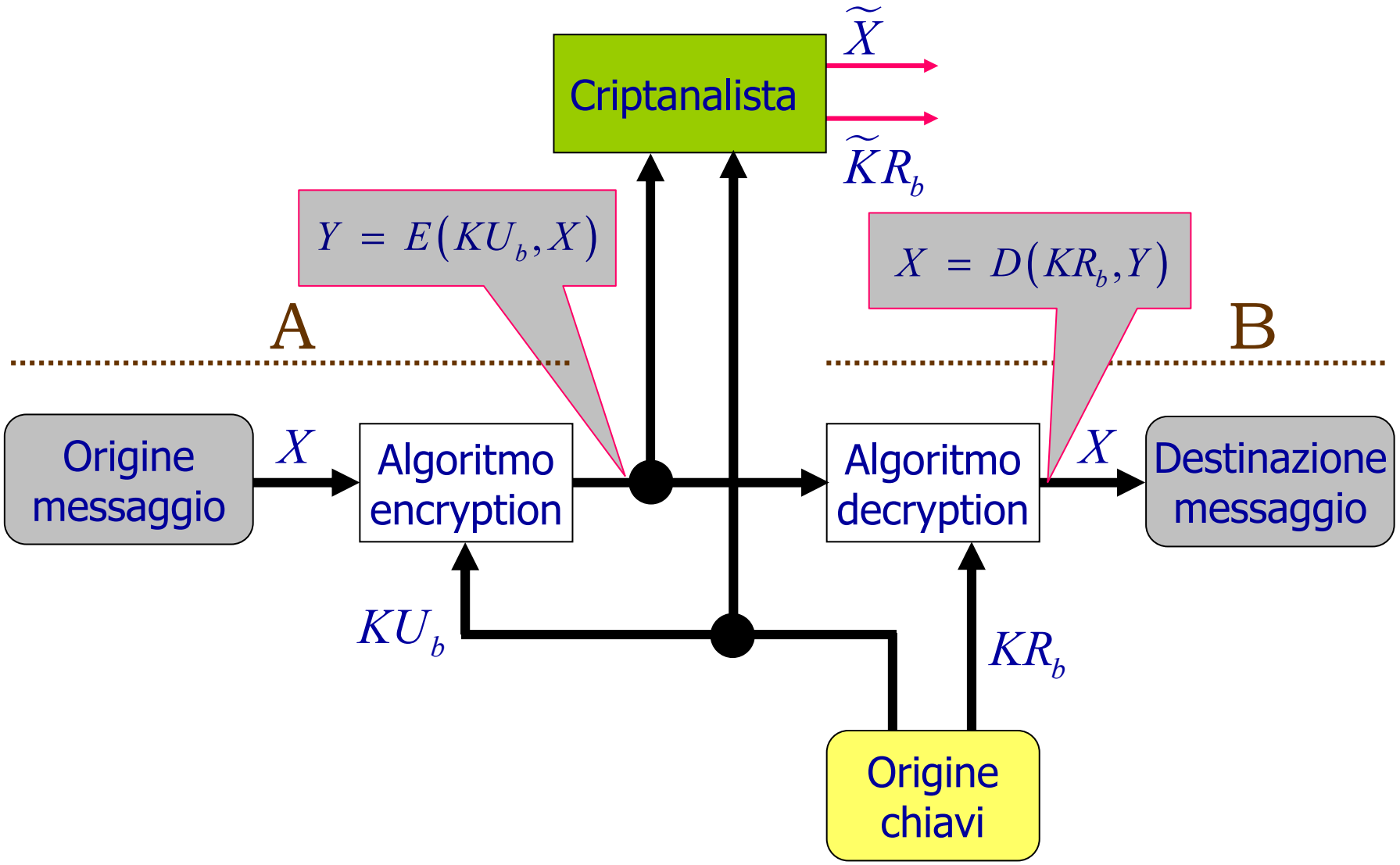
Idea nota già da prima in ambiente militare

Caratteristiche

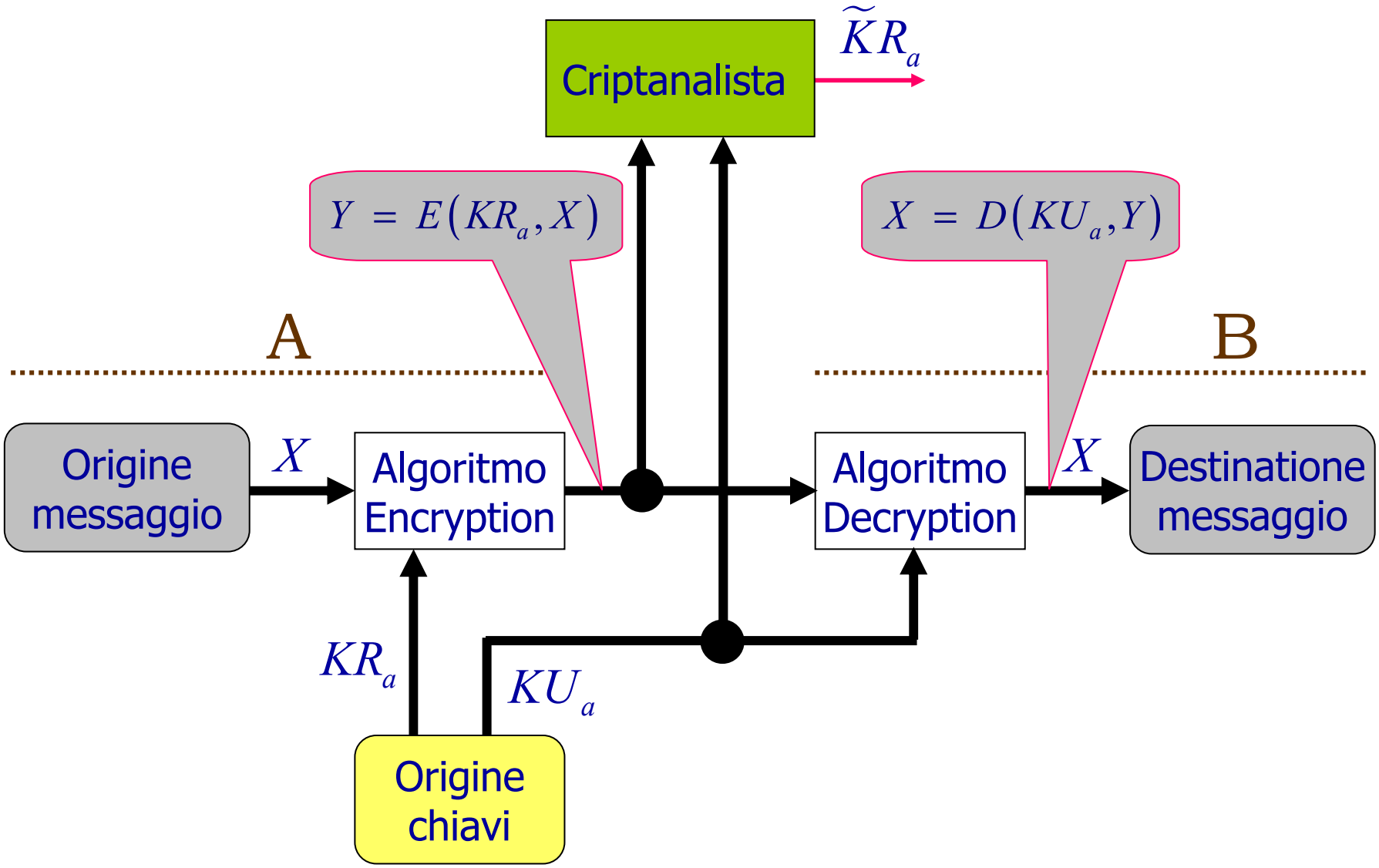
Gli algoritmi a chiave pubblica utilizzano due chiavi
e hanno le seguenti caratteristiche:

- È computazionalmente impossibile trovare la chiave di decriptazione dalla conoscenza dell'algoritmo e della chiave di criptazione
- È computazionalmente facile criptare/decriptare un messaggio se si conosce la chiave relativa
- In diversi schemi (come RSA) le due chiavi possono essere scambiate tra loro

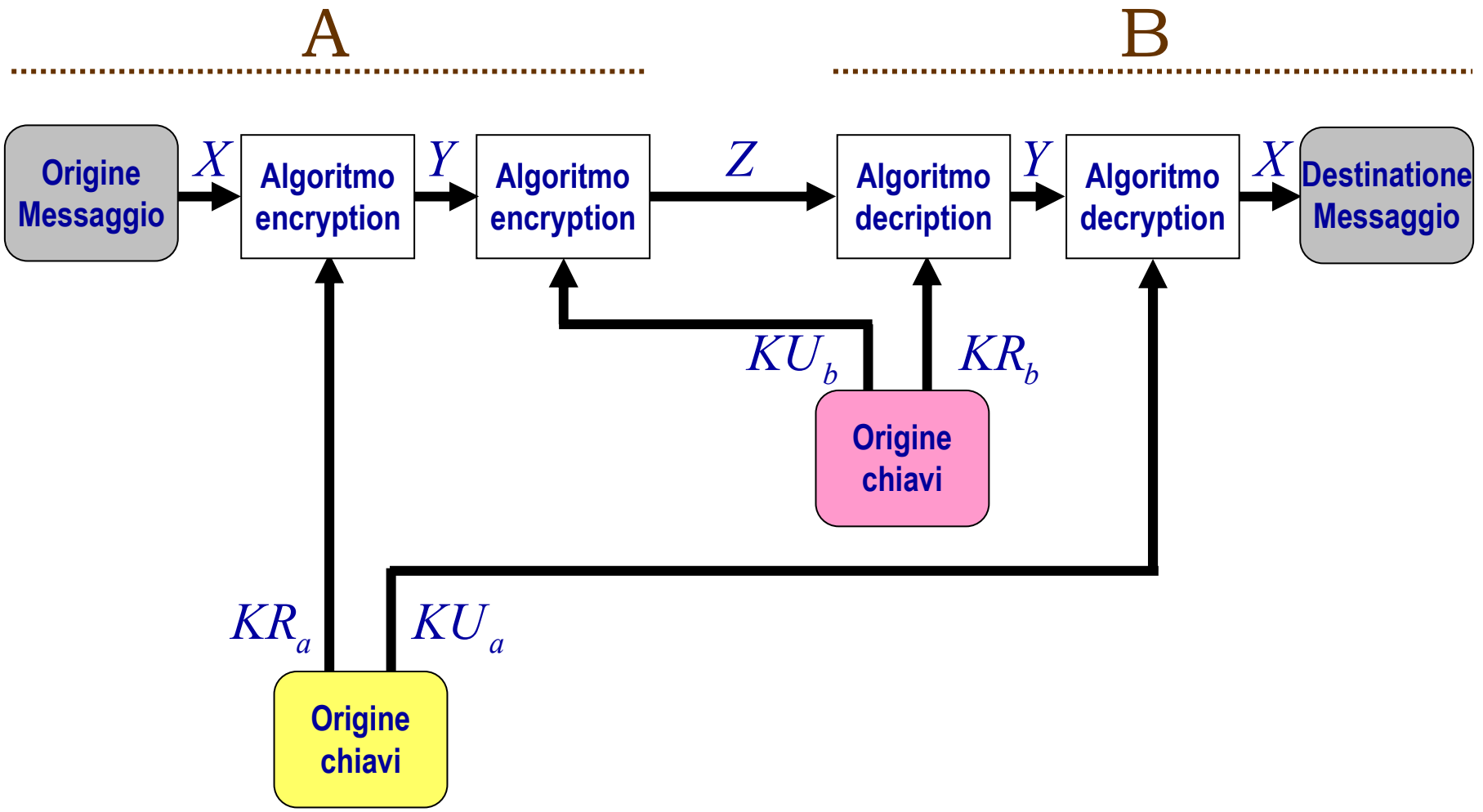
Segretezza con la PKC



Autenticazione con PKC



Criptosistemi a chiave pubblica



Applicazioni

Possibile classificarle in 3 categorie

- Encryption/decryption (segretezza)
- Firma digitale (autenticazione)
- Scambio delle chiavi (per le chiavi di sessione)

Non tutti gli algoritmi adatti per tutti gli usi; alcuni sono specifici

Algoritmi principali

Algoritmo	Encryption Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Sicurezza schemi a chiave pubblica

- Come negli schemi a chiave privata è sempre possibile in via teorica l'attacco con ricerca esaustiva a forza bruta
- Possibilità pratica legata però alla lunghezza delle chiavi che qui sono troppo lunghe (512 - 1024 – 2048 bit)
- Sicurezza basata su differenza tra difficoltà di un pesante problema di criptanalisi e facilità di encrypt/decrypt
- La possibilità della criptanalisi esaustiva è nota ma è resa praticamente troppo pesante
- Bisogno di usare numeri grandissimi
- Quindi lentezza nei confronti degli schemi a chiave privata

RSA

- Rivest, Shamir e Adleman del MIT nel 1977
- Sistema a chiave pubblica più noto e usato
- Brevetto scaduto nel 2000
- Sicurezza legata alla difficoltà di fattorizzare grandi numeri
- Basato sulla esponenziazione modulo un primo, di numeri interi in un campo di Galois
- Utilizza grandi numeri interi (tipicamente > 512 bit)
- Chiavi funzioni di una coppia di grandi numeri primi
- Cifrario a blocco

RSA Key Setup

Un utente genera una coppia privata/pubblica di chiavi

1. Selezione a caso di due grandi primi p, q (segreti)
2. Calcolo del modulo di sistema $n = p \cdot q$ (pubblico)

   Questo n avrà un toziente $\phi(n) = (p-1)(q-1)$
3. Selezione a caso dell'esponente pubblico e

   con $GCD(e, \phi(n)) = 1; \ 1 < e < \phi(n)$
4. Determinazione dell'esponente privato d tale che:

   $e \cdot d \equiv 1 \bmod \phi(n)$ e $0 \leq d \leq n$

- Pubblicazione della chiave pubblica di encryption $KU\{e,n\}$
- Conservazione della chiave privata di decryption $KR\{d,n\}$

Uso di RSA

Per criptare un messaggio M il sender:

Ottiene la chiave pubblica del recipient $KU\{e,n\}$

Calcola $C = M^e \bmod n$ ove $0 \le M < n$

Per decriptare un ciphertext C il recipient:

Usa la sua chiave privata $KR\{d,n\}$

Calcola $M = C^d \bmod n$

RSA (2)

$$C = M^e \pmod{n}$$

$$M = C^d \pmod{n} = \left(M^e \bmod n\right)^d \bmod n = M^{ed} \pmod{n}$$

Sender conosce n ed e $\longrightarrow$ $KU = \{n, e\}$

Receiver conosce n e d $\longrightarrow$ $KR = \{n, d\}$

Problema:

Come determinare un insieme $\{n, e, d\}$ tale che

$$M^{ed} \pmod{n} = M \quad \forall\, M < n$$

RSA (3)

Teorema di Eulero

$$\forall\left((a,n)\ : GCD(a,n)=1\right)\ \ a^{\phi(n)} \equiv 1\ \ (\mathrm{mod}\, n)$$

Un corollario stabilisce che, dati:

- Una coppia di numeri primi $p,\ q$
- Due interi $m,\ n$ tali che $n = pq$ e $0 < m < n$
- Un intero arbitrario k

allora:

$$m^{k\phi(n)+1} = m^{k(p-1)(q-1)+1} \equiv m\,(\mathrm{mod}\, n)$$

Perché RSA funziona

Teorema di Eulero

$$\forall\left((a,n)\ : GCD(a,n)=1\right)\ \ a^{\phi(n)} \equiv 1\ (\mathrm{mod}\, n)$$

In RSA si ha: $n = p\cdot q$ con p,q primi

quindi $\phi(n) = (p-1)(q-1)$

d è scelto per essere inverso moltiplicativo di $e\ \mathrm{mod}\,\phi(n)$

Quindi $\exists K : e\cdot d = 1+K\cdot\phi(n)$

Perché RSA funziona

Da tutto ciò $C^d = M^{ed} = M^{1+K\cdot\phi(n)} = M\cdot\left(M^{\phi(n)}\right)^K \bmod n$

In base al lemma del teorema di Eulero si ha:

$$M^{\phi(n)} \equiv 1 \bmod n$$

Allora $C^d = M \bmod n$

Esempio di RSA

1. Si scelgono p = 47 e q = 71
2. Si calcola $n = pq$ = 3337
3. Si calcola $\phi(n)$ = $(p\text{-}1)(q\text{-}1)$ = 46·70 = 3220
4. Si sceglie un e = 79 [tale che GCD(e, $\phi(n)$)]
5. Si determina d = $79^{-1} \bmod 3220$= 1019
6. Si pubblica la chiave pubblica KU = {3337, 79}
7. Si conserva la chiave privata KR = {3337, 1019}

Esempio di RSA (cont)

Un esempio di encryption/decryption è:

Messaggio M = 6882326879666683

Viene spezzato in blocchi

m1 = 688 m2 = 232 m3 = 687 m4 = 966 m5 = 668 m6 = 003

Primo blocco criptato come 688^{79} mod 3337 = 1570

Procedendo similmente si ha:

C = 1570 2756 2091 2276 2423 158

Aspetti computazionali

Due problemi: Encryption/Decryption, Keys generation

Encryption/Decryption comporta esponenziazioni

$$C = M^e \bmod n$$

Si è detto

$$(a \times b) \bmod n = \left[(a \bmod n) \times (b \bmod n) \right] \bmod n$$

Da questa -> possibilità di calcolo di a^b

In genere

$$b = \sum_{b_i \neq 0} 2^i$$

Esponenziazione

Allora $$a^b = a^{\left(\sum_{b_i \neq 0} 2^i\right)} = \prod_{b_i \neq 0} a^{2i}$$

e quindi

$$a^b \bmod n = \left[\prod_{b_i \neq 0} a^{2i}\right] \bmod n = \left[\prod_{b_i \neq 0} \left(a^{2i} \bmod n\right)\right] \bmod n$$

Algoritmo Square and Multiply

Esponenziazione

```
c  =  0;  f  =  1
for i = k  downto o
   do c = 2 x c
      f = (f x f) mod n
   if  bi  ==  1 then
      c = c + 1
      f = (f x a) mod n
return f
```

Encryption efficiente

Encryption utilizza l'esponenziazione ad una potenza e

Se e è piccola il processo risulta più veloce

Frequentemente si sceglie $e = 65537 = 2^{16} + 1$

Altre scelte frequenti sono $e = 3$ ed $e = 17$

Se però e troppo piccolo ($e = 3$) possibilità di attacchi

Usando il teorema cinese del resto e 3 messaggi con moduli diversi

Se e è fissato bisogna scegliere n in modo che

$$GCD\left(e, \phi(n)\right) = 1$$

Ossia non usare p e q che non siano coprimi ad e

Decryption efficiente

Decryption utilizza l'esponenziazione ad una potenza d

Se d non grande -> insicurezza

Possibile usare il teorema cinese del resto (CRT) per calcoli mod p e mod q separati e quindi combinarli

Risulta approssimativamente 4 volte più veloce

Solo il possessore della chiave privata che conosce p e q può usare questa tecnica

Generazione delle chiavi RSA

Gli utilizzatori di RSA devono:

- Determinare a caso due primi p e q
- Selezionare e o d e calcolare l'altro esponente

I primi p e q non devono essere facilmente ricavabili dal modulo n

- Ciò vuol dire che devono essere grandi
- Tipicamente si tenta e usano test probabilistici

Sicurezza di RSA

Quattro possibili tipi di attacco:

- Forza bruta (praticamente impossibile)
- Attacco matematico (arduo per la difficoltà di calcolare $\phi(n)$ fattorizzando modulo n
- Timing attacks (basati sul tempo di decryption)
- Chosen ciphertext attacks (sfruttando proprietà dell'algoritmo)

Attacco matematico a RSA

Tre approcci possibili:

Ricordiamo che è nota la chiave pubblica KU$\{e, n\}$

- Fattorizzare $n = pq$; questo permette di calcolare $\phi(n)$ e quindi d
- Determinare direttamente $\phi(n)$ e quindi trovare d

$$e \cdot d \equiv 1 \bmod \phi(n) \quad \text{e} \quad 0 \leq d \leq n$$

- Trovare d direttamente

Sono ritenuti tutti e 3 equivalenti alla fattorizzazione

Il problema è però meno difficile di quello che sembra

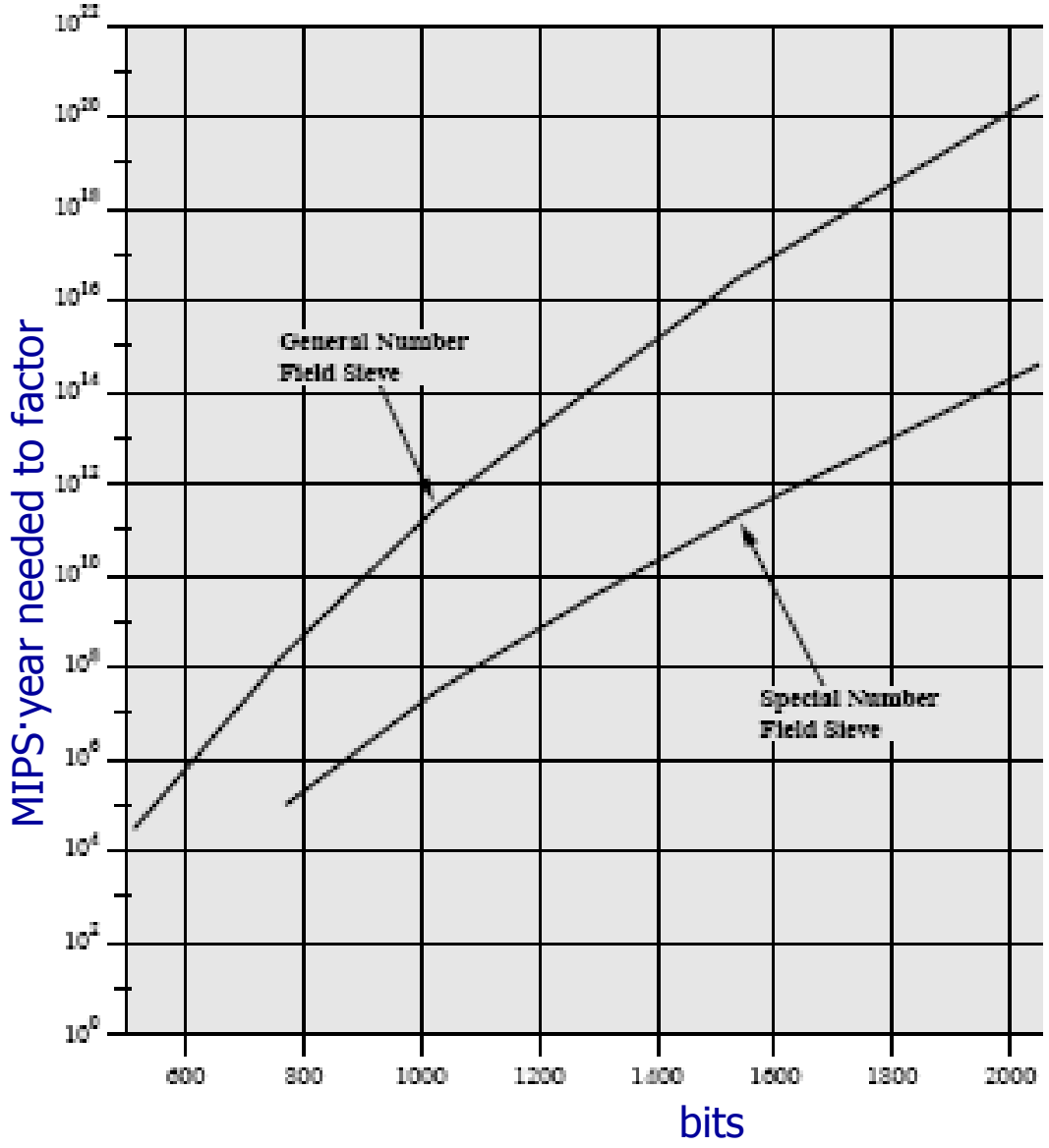
Vi sono stati miglioramenti nel tempo dovuti alla potenza dei computer e al miglioramento degli algoritmi usati

Problema della fattorizzazione

L'approccio della fattorizzazione è stato il più tentato

Number of Decimal Digits	Approximate Number of bits	Date Achieved	MPIS·Year	Algorithm
100	332	April 1991	7	Quadratic sieve
110	365	April 1992	75	Quadratic sieve
120	398	June 1993	830	Quadratic sieve
129	428	April 1994	5000	Quadratic sieve
130	431	April 1996	1000	Generalized number field sieve
140	465	February 1999	2000	Generalized number field sieve
155	512	August 1999	8000	Generalized number field sieve
160	530	April 2003	-	Lattice sieve
174	576	December 2003	-	Lattice sieve
200	663	May 2005	-	Lattice sieve

Problema della fattorizzazione



Timing Attacks

Similare ad uno scassinatore che osserva quanto tempo si impiega per girare il quadrante di una cassaforte

Applicabile pure ai sistemi criptografici

Un criptanalista può calcolare una chiave privata notando il tempo necessario per decriptare i messaggi.

L'esponente è calcolato bit per bit partendo dal LSB

Il pericolo è eliminato facilmente introducendo un tempo di esponenziazione costante, un ritardo casuale, mascherando il processo con una moltiplicazione per un numero casuale prima della esponenziazione