Crittografia a chiave pubblica

One-Time Pad (1917)

Non può essere decifrato senza conoscere la chiave

Assolutamente sicuro, ma...

- richiede una nuova chiave segreta per ogni messaggio
- perfettamente casuale
- e lunga come il messaggio da scambiare!

come si genera e come si scambia la chiave???

Estremamente attraente per chi richieda una sicurezza assoluta e sia disposto a pagarne i costi

AES

Advanced Encryption Standard (AES)

- standard per le comunicazioni riservate ma “non classificate”
- pubblicamente noto e realizzabile in hardware su computer di ogni tipo
- Chiavi brevi (qualche decina di caratteri, 128 o 256 bit)

e le chiavi ?

Ma come si può scambiare una chiave segreta con facilità e sicurezza?

La chiave serve per comunicare in sicurezza , ma deve essere stabilita comunicando "in sicurezza" senza poter ancora usare il cifrario...

Il problema dello scambio della chiave

N utenti vogliono comunicare tra loro su un canale condiviso

Charlie può vedere i messaggi scambiati da Alice e Bob, ma non li può decifrare

Soluzione naïve: ogni utente memorizza N-1 chiavi diverse,condivise con ciascun altro utente (# quadratico di chiavi, da generare e scambiare)

Soluzione alternativa

- ricorrere ad un trusted 3rd party (TTP):
- ogni utente si deve ricordare una sola chiave per comunicare con TTP
- TTP gestisce la creazione delle chiavi condivise tra le coppie di utenti

TTP: esempio

Alice vuole comunicare con Bob usando un TTP, che conosce la chiave k_A di Alice e la chiave k_B di Bob

- Alice avvisa TTP: "Voglio comunicare con Bob"
- TTP genera casualmente una chiave k_{AB}
- TTP manda ad Alice $c_A = C(k_{AB}, k_A)$ and $c_B = C(k_{AB}, k_B)$
- Alice invia c_B a Bob
- Alice e Bob iniziano a comunicare usando la chiave k_{AB}

TTP: problemi

Questo schema presenta due problemi principali

- TTP deve essere sempre online
- TTP conosce tutte le chiavi: è un unico punto di errore per l'intero sistema

Ci sono situazione in cui questo approccio ha senso:

- all'interno di una università/ambiente ristretto

New Directions in Cryptography

Invited Paper

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

***Abstract*—Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.**

I. Introduction

WE STAND TODAY on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create a need for new types of cryptographic systems which minimize the necessity of secure key distribution channels and supply the equivalent of a written signature. At the same time, theoretical developments in information theory and computer science show promise of providing provably secure cryptosystems, changing this ancient art into a science.

The development of computer controlled communication networks promises effortless and inexpensive contact between people or computers on opposite sides of the world, replacing most mail and many excursions with telecommunications. For many applications these contacts must be made secure against both eavesdropping and the injection of illegitimate messages. At present, however, the solution of security problems lags well behind other areas of communications technology. Contemporary cryptography is unable to meet the requirements, in that its use would impose such severe inconveniences on the system users, as to eliminate many of the benefits of teleprocessing.

Manuscript received June 3, 1976. This work was partially supported by the National Science Foundation under NSF Grant ENG 10173. Portions of this work were presented at the IEEE Information Theory Workshop, Lenox - MA, June 23-25, 1975 and the IEEE International Symposium on Information Theory in Ronneby, Sweden, June 21-24, 1976.

W. Diffie is with the Department of Electrical Engineering, Stanford University, Stanford, CA, and the Stanford Artificial Intelligence Laboratory, Stanford, CA 94305.

M. E. Hellman is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305.

The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel. In order to use cryptography to insure privacy, however, it is currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such as private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks.

Section III proposes two approaches to transmitting keying information over public (*i.e.*, insecure) channels without compromising the security of the system. In a *public key cryptosystem* enciphering and deciphering are governed by distinct keys, E and D, such that computing D from E is computationally infeasible (e.g., requiring 10^{100} instructions). The enciphering key E can thus be publicly disclosed without compromising the deciphering key D. Each user of the network can, therefore, place his enciphering key in a public directory. This enables any user of the system to send a message to any other user enciphered in such a way that only the intended receiver is able to decipher it. As such, a public key cryptosystem is a multiple access cipher. A private conversation can therefore be held between any two individuals regardless of whether they have ever communicated before. Each one sends messages to the other enciphered in the receiver's public enciphering key and deciphers the messages he receives using his own secret deciphering key.

We propose some techniques for developing public key cryptosystems, but the problem is still largely open.

Public key distribution systems offer a different approach to eliminating the need for a secure key distribution channel. In such a system, two users who wish to exchange a key communicate back and forth until they arrive at a key in common. A third party eavesdropping on this exchange must find it computationally infeasible to compute the key from the information overheard. A possible solution to the public key distribution problem is given in Section III, and Merkle [1] has a partial solution of a different form.

A second problem, amenable to cryptographic solution, which stands in the way of replacing contemporary busi-

Distribuzione delle chiavi

Nel 1976 viene proposta alla comunità scientifica un algoritmo per generare e scambiare una chiave segreta su un canale insicuro

Merkle Hellman Diffie

senza la necessità che le due parti si siano scambiate informazioni o incontrate in precedenza

questo algoritmo, detto protocollo DH, è ancora largamente usato nei protocolli crittografici su Internet

Crittografia a chiave pubblica

Nel 1976 Diffie e Hellman. propongono alla comunità scientifica anche la definizione di crittografia a chiave pubblica (ma senza averne un'implementazione pratica)

Turing Award 2015

Cifrari simmetrici

Nei cifrari simmetrici, la chiave di cifratura

- è uguale a quella di decifrazione (o l'una può essere facilmente calcolata dall'altra)
- ed è nota solo ai due partner che la scelgono di comune accordo e la mantengono segreta

Cifrari a chiave pubblica (asimmetrici)

Obiettivo: permettere a tutti di inviare messaggi cifrati ma abilitare solo il ricevente (BOB) a decifrarli

Le operazioni di cifratura e decifrazione sono pubbliche e utilizzano due chiavi **diverse**:

k_{pub} per cifrare: è **pubblica**, nota a tutti;

k_{priv} per decifrare: è **privata**, nota solo a BOB

Esiste una coppia <k_{pub}, k_{priv}> per ogni utente del sistema, scelta da questi nella sua veste di possibile destinatario

Cifrari a chiave pubblica

La cifratura di un messaggio m da inviare a BOB è eseguita da qualunque mittente come

$$c = C(m, k_{pub})$$

chiave k_{pub} e funzione di cifratura sono note a tutti

La decifrazione è eseguita da BOB come

$$m = D(c, k_{priv})$$

la funzione di decifrazione è nota a tutti, ma k_{priv} non è disponibile agli altri

Cifrari asimmetrici

- Ruoli completamente diversi svolti da mittente e destinatario di un messaggio

  che hanno invece ruoli intercambiabili nei cifrari simmetrici dove condividono la stessa informazione (la chiave segreta)

- Si aggiunge alla crittografia simmetrica, senza sostituirla
- È basata su proprietà di teoria dei numeri e algebra modulare, e non sull'uso di permutazioni e sostituzioni

Cifrari a chiave pubblica

1) Correttezza del procedimento di cifratura e decifrazione

BOB deve interpretare qualunque messaggio

Quindi per ogni possibile messaggio *m*

$$D\,(C\,(m,k_{pub}),\,k_{priv}) = m$$

Cifrari a chiave pubblica

2) Efficienza e sicurezza del sistema

a) la coppia di chiavi è **facile** da generare, e deve risultare praticamente impossibile che due utenti scelgano la stessa chiave (generazione casuale delle chiavi)

b) dati m e k_{pub} è **facile** per ALICE calcolare il crittogramma $c = C\,(m, k_{pub})$ (adottabilità del sistema)

c) dati c e k_{priv} è **facile** per BOB calcolare il messaggio originale $m = D\,(c, k_{priv})$ (adottabilità del sistema)

d) pur conoscendo il crittogramma c, la chiave pubblica, e le funzioni di cifratura e decifrazione, è **difficile** per il crittoanalista risalire al messaggio m (sicurezza del cifrario)

Crittografia a chiave pubblica

La funzione di cifratura deve essere una funzione one-way trap-door

→ calcolare $c = C(m, k_{pub})$ è computazionalmente facile

→ decifrare c è computazionalmente difficile

→ a meno che non si conosca un meccanismo segreto, rappresentato da k_{priv} (trap-door)

facile da calcolare …

e difficile da invertire …

a meno che non si conosca la chiave privata!

RSA (1977)

Adleman Shamir Rivest

propongono un sistema a chiave pubblica basato su una funzione "facile" da calcolare e "difficile" da invertire

(Turing Award 2002)

GCHQ: Government Communications Head Quarter

Ellis, Cock e Williamson 1970 – 75

Rapporti TOP SECRET al GCHQ sulla trasmissione cifrata non preceduta dall'accordo su una chiave, con un metodo matematico basato sui numeri primi

Il cifrario RSA

RSA si basa sulla moltiplicazione di due numeri primi p, q

Calcolare n = p x q è facile

- richiede tempo quadratico nella lunghezza della loro rappresentazione

Invertire la funzione, cioè ricostruire p e q a partire da n (univocamente possibile solo se p e q sono primi) richiede tempo esponenziale

- anche se non vi è dimostrazione del fatto che il problema sia NP hard
- l'esistenza di un algoritmo polinomiale è assai improbabile ma non assolutamente da escludere

Trap door: se si conosce uno dei fattori, ricostruire l'altro è facile

RSA utilizza l'algebra modulare

Il cifrario di ElGamal (1985)

Sistema a chiave pubblica basato sul problema del logaritmo discreto

Il cifrario di ElGamal utilizza l'algebra modulare

Prerequisiti

Algebra modulare

Usata in molti algoritmi crittografici per

➔ ridurre lo spazio dei numeri su cui si opera, e quindi aumentare la velocità di calcolo

➔ rendere *difficili* problemi computazionali che sono semplici (o anche banali) nell'algebra non modulare

Algebra modulare

Nell'aritmetica modulare le funzioni tendono a comportarsi in modo 'imprevedibile'

La funzione 2^x nell'aritmetica ordinaria è una funzione monotona crescente

x	1	2	3	4	5	6	7	8	9	10	11	12
2^x	2	4	8	16	32	64	128	256	512	1024	2048	4096

La funzione 2^x mod 13 diventa

x	1	2	3	4	5	6	7	8	9	10	11	12
2^x	2	4	8	3	6	12	11	9	5	10	7	1

Algebra modulare

- Dati due interi a, b ≥ 0 e n > 0, a è congruo a b modulo n

$$a \equiv b \bmod n$$

se e solo se esiste k intero tale che

$$a = b + k\,n$$

(a mod n = b mod n)

- Nelle relazioni di congruenza la notazione mod n si riferisce all'intera relazione, nelle relazioni di uguaglianza la stessa notazione si riferisce solo al membro dove appare:
  - 5 ≡ 8 mod 3,
  - 5 ≠ 8 mod 3 e 2 = 8 mod 3

Proprietà

$$(a + b) \bmod m = (a \bmod m + b \bmod m) \bmod m$$

$$(a - b) \bmod m = (a \bmod m - b \bmod m) \bmod m$$

$$(a \times b) \bmod m = (a \bmod m \times b \bmod m) \bmod m$$

$$a^{r \times s} \bmod m = (a^r \bmod m)^s \bmod m \quad (r \text{ e } s \text{ interi positivi})$$

ESEMPIO

$(12456 \times 3678) \bmod 10 = 45813168 \bmod 10 = 8$

$(12456 \times 3678) \bmod 10 = (6 \times 8) \bmod 10 = 8$

Algebra modulare

- n ∈ N, intero positivo

  $Z_n = \{0, 1, 2, \ldots , n - 1\}$

  $Z_n^* \subseteq Z_n$

  è l'insieme degli elementi di Z_n co-primi con n.

- Se n è primo, $Z_n^* = \{1, 2, \ldots , n - 1\}$
- Se n non è primo, calcolare Z_n^* è computazionalmente difficile

  richiede tempo proporzionale al **valore** di n (per confrontare n con gli elementi di Z_n) quindi esponenziale nella lunghezza della sua rappresentazione.

Funzione di Eulero

La funzione di Eulero $\phi(n)$ è definita come il numero di interi minori di *n* e co-primi con esso

$$\phi(n) = |Z_n^*|$$

***n* primo** $\Longrightarrow \phi(n) = n - 1$

TEOREMA

n composto $\Rightarrow$ $\phi(n) = n\,(1-1/p_1) \ldots (1-1/p_k)$

$p_1, \ldots, p_k$ fattori primi di *n*, presi senza molteplicità

TEOREMA

n prodotto di due primi (semiprimo)

$n = p\,q \Rightarrow \phi(n) = (p-1)\,(q-1)$

Teorema di Eulero

Teorema di Eulero

Per n > 1 e per ogni a **primo** con n,

$$a^{\phi(n)} \equiv 1 \mod n$$

Piccolo Teorema di Fermat

Per n primo e per ogni $a \in Z_n^*$

$$a^{n-1} \equiv 1 \mod n$$

Calcolo dell'inverso in modulo

Per qualunque a primo con n,

$a \times a^{\Phi(n)-1} \equiv 1 \mod n$ (teorema di Eulero)

$a \times a^{-1} \equiv 1 \mod n$ (definizione di inverso)

quindi

$$a^{-1} = a^{\Phi(n)-1} \mod n$$

L'inverso a^{-1} di a modulo n si può dunque calcolare per esponenziazione di a, ma occorre conoscere Φ(n), cioè fattorizzare n

... problema "difficile"

Esistenza e unicità dell'inverso sono garantiti se e solo se a e n sono co-primi, i.e., mcd(a, n) = 1

Equazione $a\,x \equiv b \bmod n$

Teorema

L'equazione $a\,x \equiv b \bmod n$ ammette soluzione se e solo se $\mathrm{mcd}(a, n)$ divide b.

In questo caso si hanno esattamente $\mathrm{mcd}(a, n)$ soluzioni distinte

Equazione $a\,x \equiv b \bmod n$

Corollario

L'equazione $a\,x \equiv b \bmod n$ ammette **un'unica soluzione**

$\Leftrightarrow$

a e n sono co-primi, i.e., $\mathrm{mcd}(a, n) = 1$

$\Leftrightarrow$

esiste l'inverso a^{-1} di a

Equazione $a\,x \equiv 1 \bmod n$

Ammette esattamente una soluzione (l'inverso di a) se e solo se a e n sono primi tra loro

L'inverso si può calcolare come

$$a^{-1} = a^{\Phi(n)-1} \bmod n$$

ma occorre conoscere $\Phi(n)$, cioè fattorizzare n

... problema “difficile”

Algoritmo di Euclide Esteso

L’algoritmo di Euclide per il calcolo del mcd si può estendere per risolvere l'equazione in due incognite

$$a\,x + b\,y = \mathrm{mcd}\,(a, b)$$

```
Function Extended_Euclid(a,b)
        if (b = 0) then return <a, 1, 0>
        else
          < d’, x’, y’> = Extended_Euclid(b, a mod b);
          < d, x, y> = <d', y', x' - ⌊a/b⌋ y' >
          return <d, x, y>
```

La funzione Extended_Euclid restituisce una delle triple di valori $<\mathrm{mcd}(a,b), x, y >$ con x, y tali che $ax + by = \mathrm{mcd}(a,b)$. Quindi $d = \mathrm{mcd}\,(a, b)$

Algoritmo di Euclide Esteso

L'algoritmo di Euclide per il calcolo del mcd si può estendere per risolvere l'equazione in due incognite

$$a\,x + b\,y = mcd\,(a, b)$$

```
Function Extended_Euclid(a,b)
        if (b = 0) then return <a, 1, 0>
        else
          < d', x', y'> = Extended_Euclid(b, a mod b);
          < d, x, y> = <d', y', x' - ⌊a/b⌋ y' >
          return <d, x, y>
```

Complessità logaritmica nel valore di a e b, quindi **polinomiale** nella dimensione dell'input

Calcolo dell'inverso $x = a^{-1} \bmod b$

mcd(a, b) = 1

$$a\,x \equiv 1 \bmod b$$

$$\Leftrightarrow$$

$a\,x = b\,z + 1$ per un opportuno valore di *z*

$$\Leftrightarrow$$

$a\,x + b\,y = mcd(a, b)$, dove $y = -z$ e $mcd(a, b) = 1$

x è il valore dell'inverso, e si può calcolare eseguendo Extended_Euclid *(a, b)* e restituendo il secondo valore della tripla.

Esempio

$x = 5^{-1} \bmod 132 \quad \rightarrow \quad 5x + 132\,y = 1$

$\langle d, x, y \rangle = \langle d', y', x' - \lfloor a/b \rfloor y' \rangle$

E_E (5, 132) $\rightarrow$ <1, **53**, ...> //

E_E(132, 5) $\rightarrow$ <1, -2, 1 + 2*26> = <1, -2, 53> //

E_E(5, 2) $\rightarrow$ <1, 1, 0-1*2> = <1, 1, -2> //

E_E(2, 1) $\rightarrow$ <1, 0, 1-0*2> = <1, 0, 1> //

E_E(1, 0) $\rightarrow$ <1, 1, 0> //

Teorema cinese del resto

Siano $n_1, ..., n_k$ interi a due a due coprimi ($MCD(n_i, n_j) = 1$ quando $i \neq j$). Allora, comunque si scelgano degli interi $a_1, ..., a_k$, **esiste un'unica soluzione** intera x del sistema di congruenze

$x \equiv a_1 \pmod{n_1}$

$x \equiv a_2 \pmod{n_2}$

...

$x \equiv a_k \pmod{n_k}$

modulo il prodotto **n** = $n_1 ... n_k$.

Mostra come, sotto opportune condizioni, un sistema di congruenze possa essere sostituito da un'unica congruenza: conoscendo i resti della divisione di un intero x per gli interi $n_1, ..., n_k$, si può determinare univocamente il resto della divisione di x per il prodotto degli interi $n_1, ..., n_k$ (a condizione che questi siano coprimi tra loro)

Teorema cinese del resto

Si può trovare una soluzione x come segue:

- Per ogni i gli interi n_i e $N_i = n/n_i$ sono coprimi
- utilizzando l'algoritmo di Euclide esteso si possono trovare due interi r e s tali che

  $r\, n_i + s\, N_i = 1$
- Ponendo $e_i = s\, N_i$, si ottiene

  $e_i \equiv 1 \pmod{n_i}$ infatti $r\, n_i + e = 1$

  $e_i \equiv 0 \pmod{n_j}$ per $j \neq i$, infatti $n_j \mid N_i$ e $e_i = s\, N_i$

Una soluzione del sistema di congruenze è quindi $x = \sum_{i=1}^{k} a_i\, e_i$

Il teorema è molto usato per eseguire velocemente computazioni con interi molto grandi, in quanto permette di sostituire una computazione con diverse computazioni simili, definite su interi più piccoli

Generatori

$a \in Z_n^*$ è un generatore di Z_n^* se la funzione

$a^k \bmod n \qquad 1 \le k \le \phi(n)$

genera **tutti e soli** gli elementi di $\mathbf{Z_n^*}$

Produce come risultati tutti gli elementi di Z^*_n, ma in un ordine difficile da prevedere

ESEMPIO: $g = 2$ è un generatore di $Z^*_{13} = \{1,2,\ldots, 12\}$

x	1	2	3	4	5	6	7	8	9	10	11	12
2^x	2	4	8	3	6	12	11	9	5	10	7	1

Teorema di Eulero $[a^{\Phi(n)} \bmod n = 1]$

$\Longrightarrow \quad 1 \in Z_n^*$ è generato per $k = \Phi(n)$

Per ogni generatore, $a^k \not\equiv 1 \bmod n \qquad 1 \le k < \Phi(n)$

Generatori

Non per tutti i valori di n, Z_n^* ha generatori

➔ Z_8^* non ammette generatori

TEOREMA

Se n è un numero primo, Z^*_n ha almeno un generatore

Per n primo, non tutti gli elementi di Z_n^* sono suoi generatori (1 non è mai un generatore, e altri elementi possono non esserlo)

Per n primo, i generatori di Z_n^* sono in totale $\Phi(n-1)$

Esempi

3 genera Z_7^*

$Z_7^* = \{1, 2, 3, 4, 5, 6\}$

$\Phi(7) = 6$

$3^k \bmod 7 = 3, 2, 6, 4, 5, 1 \qquad 1 \leq k \leq 6$

2 non genera Z_7^*

$2^3 \bmod 7 = 2^6 \bmod 7 = 1$

Problemi sui generatori rilevanti in Crittografia

Determinare un generatore di Z_n^*, n numero primo

- si possono provare per a tutti gli interi in [2, n-1], fino a trovare il generatore

  ... tempo esponenziale nella dimensione di n ...

- il problema si ritiene computazionalmente difficile
- risolto in pratica con algoritmi randomizzati (con alta probabilità di successo)

Problemi sui generatori rilevanti in Crittografia

Calcolo del logaritmo discreto

- risolvere nell'incognita x l'equazione

  $a^x \equiv b \bmod n$ n primo

- l'equazione ammette una soluzione per ogni valore di b se e solo se a è un generatore di Z_n^*. Tuttavia
  - non è noto a priori in che ordine sono generati gli elementi di Z_n^*
  - quindi non è noto per quale valore di x si genera b mod n
  - un esame diretto della successione richiede tempo esponenziale nella dimensione di n
  - non è noto un algoritmo polinomiale di soluzione

Algebra modulare

Nell'aritmetica modulare le funzioni tendono a comportarsi in modo 'imprevedibile'

La funzione 2^x nell'aritmetica ordinaria è una funzione monotona crescente

x	1	2	3	4	5	6	7	8	9	10	11	12
2^x	2	4	8	16	32	64	128	256	512	1024	2048	4096

La funzione 2^x mod 13 diventa

x	1	2	3	4	5	6	7	8	9	10	11	12
2^x	2	4	8	3	6	12	11	9	5	10	7	1

Algebra modulare

Algebra NON modulare

$2^x = 512$, x = ?

per errore, supponiamo x = 8
$2^8 = 256$ → è stato scelto un numero troppo basso
proviamo x = 9 ok!

Algebra modulare

$2^x \bmod 13 = 5$, x = ?

per errore, supponiamo x = 6

$2^6 \bmod 13 = 12$

???

occorre provare tutti i valori di x
la risposta giusta è x = 9

Funzioni *one-way trap-door*

- Esistono funzioni matematiche che sembrano possedere i requisiti richiesti (proprietà di teoria dei numeri e di algebra modulare)
- Il loro calcolo risulta incondizionatamente semplice, e la loro inversione è semplice se si dispone di un'informazione aggiuntiva sui dati (cioè di una chiave privata)
- Senza questa informazione, l'inversione richiede la soluzione di un problema NP hard, o comunque di un problema noto per cui non si conosce un algoritmo polinomiale

Fattorizzazione

- Calcolare n = p x q è facile
  - richiede tempo quadratico nella lunghezza della loro rappresentazione
- Invertire la funzione, cioè ricostruire p e q a partire da n (univocamente possibile solo se p e q sono primi) richiede tempo esponenziale
  - per quanto noto fino a oggi, anche se non vi è dimostrazione del fatto che il problema sia NP hard
  - l'esistenza di un algoritmo polinomiale è assai improbabile ma non assolutamente da escludere
- TRAP DOOR: se si conosce uno dei fattori (la chiave segreta) ricostruire l'altro è facile

Calcolo della radice in modulo

- Calcolare $y = x^z \bmod s$, con x, z, s interi,

  richiede tempo polinomiale
  - procedendo per successive esponenziazioni, si eseguono $\Theta(\log_2 z)$ moltiplicazioni

- Se s non è primo, invertire la funzione e calcolare

$$x = y^{1/z} \bmod s$$

  richiede tempo esponenziale per quanto noto

$y = x^z \bmod s$

- Se x è primo con s e si conosce

$$v = z^{-1} \bmod \Phi(s) \qquad \text{(chiave segreta)}$$

  si determina facilmente x calcolando

$$x = y^v \bmod s$$

- Infatti

$$y^v \bmod s = x^{zv} \bmod s$$

$$= x^{1 + k\Phi(s)} \bmod s = x\, x^{k\Phi(s)} \bmod s$$

$$= x \bmod s \qquad \text{(Teorema di Eulero)}$$

Calcolo del logaritmo discreto

- Calcolare la potenza

  $$y = x^z \bmod s$$

  è facile

- Invertire rispetto a z, cioè trovare z t.c.

  $$y = x^z \bmod s$$

  dati x, y, s è computazionalmente difficile

  - Gli algoritmi noti hanno la stessa complessità della fattorizzazione
  - È possibile introdurre una trap-door

Crittografia a chiave pubblica

VANTAGGI

- Se gli utenti di un sistema sono n, il numero complessivo di chiavi (pubbliche e private) è $2n$ anziché $n(n-1)/2$
- Non è richiesto alcuno scambio segreto di chiavi

SVANTAGGI

- Questi sistemi sono molto più lenti di quelli basati sui cifrari simmetrici
- Sono esposti ad attacchi di tipo chosen plain-text

Attacchi chosen plain-text

Un crittoanalista può

- scegliere un numero qualsiasi di messaggi in chiaro

  $m_1 , \dots , m_h$

- cifrarli con la funzione pubblica C e la chiave pubblica k_{pub} del destinatario, ottenendo i crittogrammi

  $c_1 , \dots , c_h$

- quindi può confrontare qualsiasi messaggio cifrato c* che viaggia verso il destinatario con i crittogrammi in suo possesso

Attacchi chosen plain-text

- se c* coincide con uno di essi il messaggio è automaticamente decifrato

- se invece è diverso da tutti i c_i il crittoanalista ha comunque acquisito una informazione importante: il messaggio è diverso da quelli scelti

- L'attacco è particolarmente pericoloso se il crittoanalista sospetta che DEST debba ricevere un messaggio particolare ed è in attesa di vedere quando questo accada

Cifrari ibridi

- Si usa un cifrario a chiave segreta (AES) per le comunicazioni di massa
- e un cifrario a chiave pubblica per scambiare le chiavi segrete relative al primo, senza incontri fisici tra gli utenti
- La trasmissione dei messaggi lunghi avviene ad alta velocità, mentre è lento lo scambio delle chiavi segrete
  - le chiavi sono composte al massimo da qualche decina di byte
  - attacco chosen plain-text è risolto se l'informazione cifrata con la chiave pubblica (chiave segreta dell'AES) è scelta in modo da risultare imprevedibile al crittoanalista
- La chiave pubblica deve essere estratta da un certificato digitale valido, per evitare attacchi *man-in-the-middle*

Realizzazioni

Merkle

- La difficoltà di inversione della funzione di cifratura era basata sulla risoluzione del problema dello Zaino (NP hard)
- Il cifrario è stato violato per altra via, successivi cifrari basati sullo stesso problema sono tuttora inviolati

Rivest, Shamir e Adleman → RSA (1978)

- Fonda la sua sicurezza sulla difficoltà di fattorizzare grandi numeri interi (problema che non è comunque NP-hard)
- È ad oggi inviolato, ed è stato il primo cifrario asimmetrico di largo impiego

ElGamal

- Fonda la sua sicurezza sulla difficoltà di calcolare il logaritmo discreto (→ ECC)