Crittografia a chiave privata

n = 2 round

Cifrari a Blocchi — Block Ciphers [S-P-BLOCKS]

Cifrari a cascata — Stream Ciphers $d = 2^m - 1$

m = registri

(A) $\xrightarrow{\{P\}_{K_{AB}}}$ (B) — chiavi simmetriche K_{AB}

K_{AB} K_{AB}

Crittografia a chiave pubblica

serve per distribuire le chiavi K_{AB}

- Meccanismi

$n = p \cdot q$ primalità, fattorizzazione
RSA, Shamir 3 passi, Needham-Schroeder PK

$b \equiv a^x \pmod n$ log discreto $x = L_a(b)$
D-H, ElGamal, DSA, ...

$x^2 \equiv a \pmod n$ radice quadrata $x \equiv \sqrt{a}$
Rabin, ...

- nei campi finiti $\mathbb{Z}_p^*$, $GF(p^n)$
- nei gruppi abeliani delle curve Ellittiche
- nei reticoli n-dimensionali (Lattices) gruppi abeliani

- Altri meccanismi — Knapsack - Merkle-Hellmann
  — McEliece - Codici a correzione di errori
  — Quantum Cryptography

Problema {dello zaino / del raccoglimento}

- Knapsack problem

$$\sum_{i=0}^{k-1} a_i v_i = V \qquad (a_i \in \mathbb{Z}_2;\ \{v_i\})$$

data $\{v_i\}$ e V, trovare $\{a_i\}$

e cioè trova il subset I degli indici $1 \le i \le k-1$ tale che la somma dei valori v_i di questi indici ($a_i = 1$) sia V

Crittografia di McEliece

Codici a correzione di errore

Se ho una stringa binaria di 1024 bit che contiene 50 errori, allora ci sono

$$\binom{1024}{50} \approx 3 \cdot 10^{85}$$

possibili localizzazioni di questi errori. Una ricerca esaustiva diventa infattibile in tempi ragionevoli. Se si dispone di un efficiente algoritmo di decodifica sconosciuto agli altri, allora si possono correggere gli errori localizzandoli ed ottenendo la stringa corretta.

Quantum key Distribution (sistemi ottici)

i qubit sono unità vettoriali (polarizzazione dei fotoni) e costituiscono l'architettura di codifica dei bit Quantici. L'intercettazione provoca la modifica dello stato delle particelle e quindi l'immediata rivelazione dell'intrusione da parte degli interlocutori. (Perfect Security?) ☺.
Visti gli errori di trasmissione introdotti dall'intercettatore.

cifrario a blocchi

1977 Rivest-Shamir-Adleman

modulo $n = p \times q$ $\quad p$ e q primi grandi

l'esponente e deve essere scelto tale che

$$mcd(e, \varphi(n)) = 1 \quad \text{cioè} \quad e \in \mathbb{Z}^*_{\varphi(n)}$$

allora esiste $e^{-1} = d \in \mathbb{Z}^*_{\varphi(n)} \qquad d, e \in \mathbb{Z}^*_{\varphi(n)}$

$\varphi(n) = (p-1)(q-1)$ è pari $\qquad (P \cdot P = P)$

$\varphi[\varphi(n)] - 1$

risulta $d \equiv e \pmod{\varphi(n)}$

Si sceglie Plaintext $P \in \mathbb{Z}_n \qquad d \cdot e \equiv 1 \pmod{\varphi(n)}$

$$\begin{cases} C = P^e \pmod n \\ P = C^d \pmod n \end{cases} \qquad P, C \in \mathbb{Z}_n$$

la aritmetica è tale che se

$$d \cdot e \equiv 1 \pmod{\varphi(n)}$$

allora

$$(P^e)^d \equiv (P^d)^e \equiv P \pmod n$$

infatti risulta che

$$d \cdot e = t\,\varphi(n) + 1$$

ammettiamo che $P \in \mathbb{Z}_n^*$ e cioè $\text{mcd}(P, n) = 1$

allora $$P^{\varphi(n)} \equiv 1 \pmod{n}$$

per cui $$(P^e)^d \equiv P^{t\varphi(n)+1} \equiv \left\{\left[P^{\varphi(n)}\right]^t \cdot P\right\} \equiv$$

$$\equiv (1^t \cdot P) \equiv P \pmod{n}$$

Il cifrario è "commutativo"

$$(P^e)^q \equiv (P^d)^e \equiv P \pmod{n}$$

se $d \cdot e \equiv 1 \pmod{\varphi(n)}$.

Il protocollo a 3 passi di Shamir sfrutta la proprietà commutativa del cifrario RSA

M1. $A \to B$: $\quad RSA_{K_A}(P)$

M2. $B \to A$: $\quad RSA_{K_B}\left[RSA_{K_A}(P)\right] \equiv RSA_{K_A}\left[RSA_{K_B}(P)\right]$

M3. $A \to B$: $\quad RSA_{K_B}(P)$

Il messaggio P è una chiave simmetrica K_{AB} scambiata (ad es. chiave a 128 bit AES).
K_A è l'esponente "e_A" di Alice e K_B l'esponente "e_B" di Bob. Bob e Alice decifrano con i rispettivi "d_B" e "d_A" e cioè K_B^{-1} e K_A^{-1}, inversi di "e_B" e "e_A" $\pmod{\varphi(n)}$.

Ci sono due assunzioni (claim)

Trovare $\varphi(n)=(p-1)(q-1)$ oppure trovare l'esponente di decifratura d è computazionalmente difficile come fattorizzare $n=p\times q$.

claim 1: Noti $n=p\times q$ e $\varphi(n)$ è possibile ricavare p e q.

$$p,q=\frac{(n-\varphi(n)+1)\pm\sqrt{(n+\varphi(n)+1)^2-4n}}{2} \quad (*)$$

claim 2 Noti "d" e "e" si può fattorizzare probabilmente n.

si adotta il metodo dell'"esponente universale" per fattorizzare n, infatti si ha

per "a" tale che $mcd(a,n)=1$

allora: $a \equiv (a^{\varphi(n)})^{de-1} \equiv 1 \pmod n$

(*) si osserva che $n=p\times q$ $\varphi(n)=(p-1)(q-1)$

allora $(n-\varphi(n)+1)= pq-(p-1)(q-1)+1 = p+q$

Noti $(p\cdot q)$ e $(p+q)$, le radici di

$$X^2-(p+q)X+pq=(X-p)(X-q)$$

sono p e q.

Scelta di p e q (primi grandi)

numeri interi da 100-300 cifre decimali

la scelta migliore è due numeri primi di grandezza quasi uguali (ma non troppo) (ad es. entrambi attorno 200 cifre decimali, ad es: 170 e 230 cifre.)

Per contrastare l'attacco di fattorizzazione $n = p \times q$, molto è importante la scelta di $(p-1)$ e $(q-1)$. Sono numeri fattorizzati, ad es:

$$p-1 = \prod_{i=2}^{m} 2 \cdot p_i^{r_i} \quad \left(\text{esempio } \begin{matrix} p_1 = 2 \\ r_1 = 1 \end{matrix} \right)$$

è importante che $(p-1)$ non abbia fattori p_i piccoli (ad es. 10 cifre decimali): se fosse così p va scartato in quanto n diventa facilmente fattorizzabile col metodo del $(p-1)$.

Inoltre per quanto riguarda l'esponente "e" ci sono due esigenze contrastanti:

1- la velocità di calcolo di $P^e \pmod n$ richiede square & multiply e un numero "e" moderatamente grande

2- se "e" è piccolo è più facile da attaccare (attacchi al piccolo esponente)

In realtà contano tutti e due gli esponenti di cifratura "e" e di decifratura "d" legati da $d \cdot e \equiv 1 \pmod{\varphi(n)}$.

Potrei scegliere $e = 2^{16}+1 = 65537$ (*) (5 cifre decimali) numero primo per cui $\mathrm{mcd}(e, \varphi(n)) = 1$ e di conseguenza noto $n = p \times q$. È ufficiente che d sia grande, e quindi noto n si sceglie un d grande $\mathrm{mcd}(d, \varphi(n)) = 1$ e di conseguenza si calcola "e" con $d \cdot e \equiv 1 \pmod{\varphi(n)}$

- **Attacco al piccolo esponente** — p e q primi

Teorema. Sia $q < p < 2q$ e $n = p \cdot q$ e sono $1 \leq d, e < \varphi(n)$ tali che $d \cdot e \equiv 1 \mod \varphi(n)$.

Se $d < \frac{1}{3} n^{\frac{1}{4}}$, allora d può essere calcolato con un tempo polinomiale in $\log n$. (*)

(*) Per esempio SAM è un algoritmo con tempo di esecuzione polinomiale in $\log n$. La cifratura con $e = 2^{16}+1$ conta di 16 somme di quadrati successivi.

Altra attacco va fondato alla scelta della dimensione del "plaintext" P.

Supponiamo che P è una chiave DES a 56 bit e cioè un numero a circa $56/\log_2 10 \cong 56/3{,}33 \cong 17$ cifre decimali. Se n è a 200 cifre decimali allora risulterà $C = P^e \pmod n$ con e moderatamente grande, ma anche C grande anche lui a 200 cifre decimali. Tuttavia si può fare l'attacco dello

Short Plaintext Attack

Occorre due liste

1. $C\,x^{-e} \pmod n$ per $1 \le x \le 10^9$
2. $y^e \pmod n$ per $1 \le y \le 10^9$

e controllo le coincidenze tra i termini e trovo

$$C \equiv (xy)^e \pmod n$$

per cui

$$P \equiv xy \pmod n$$

L'attacco ha successo se P è il prodotto di due soli x e y minori di 10^9!

Per contrastare l'attacco al plaintext cifrato (testo in chiaro cifrato) bisogna fare così: usare un P^* derivato da P cifrato

$$P^* \equiv (x \text{ padding bits}, P, y \text{ padding bits})$$

in cui i bit x di preambolo e i bit y di coda vengono randomici e ripetuti.

Un metodo più sofisticato è quello detto OAEP = optimal asymmetric encryption padding.

FUNZIONI HASH

Funzioni hash - Attacco del compleanno
Oracolo casuale - "Random oracle model"

Dato un messaggio m di lunghezza arbitraria

$h_m = h(m)$ h_m codice hash / message digest / hash del messaggio

h_m è di lunghezza fissa.
Nella moderna crittografia h_m è al minimo 160 bit. (SHA-1), mentre molte funzioni hash più lunghe fino a 512 bit (SHA-512, Whirlpool)

1. Dato il messaggio m, $h(m)$ si calcola velocemente
2. Scelto un codice y è computazionalmente infattibile trovare m' con $h(m') = y$
   - h è una funzione unidirezionale (one-way) detta anche resistente alle pre-immagini, o pre-image resistant, nel senso che è difficile trovare una pre-immagine m' di $y = h(m')$.
   - Si osservi che se y è il codice di message digest, si è interessati a trovare alcuni messaggi m' tali che $h(m') = y$ e non necessariamente al messaggio originario m che lo ha generato.
   - È computazionalmente difficile trovare due messaggi m_1 e m_2 tali che $h(m_1) = h(m_2)$, 3a) → in tale caso $h(\cdot)$ è detta fortemente senza collisioni (strongly collision-free).

Si osserva che se il messaggio m è lungo "a" bit (ad esempio $a = 65,536$) e il codice hash h è lungo "b" bit (ad esempio $160 = b$), l'insieme dei possibili messaggi ha cardinalità $2^{65,536}$ mentre l'insieme degli hash ha dimensione 2^{160}, e quindi ci sono molti casi di messaggi m_1 e m_2 tali che $h(m_1) = h(m_2)$. Il requisito di strongly collision-free dice solo che è difficile trovare questi casi.
In pratica si può rilassare tale requisito richiedendo alla funzione $h(x)$ che sia debolmente senza collisioni Weakly collision-free. Dato a priori x è computazionalmente infattibile trovare

$$x' \neq x \text{ tale che } h(x') = h(x)$$

3b) Questa ultima proprietà è anche chiamata resistenza alla seconda pre-immagine second preimage resistance.

Le funzioni hash servono per formare digitalmente l'hash del messaggio con una forma lunga almeno quanto l'hash. In generale le funzioni hash servono per verificare l'integrità del messaggio

ESEMPI

Sia n intero > 0 prendo ad esempio

$$h(m) = m \bmod n \qquad h(m) \in \mathbb{Z}_n$$

un intero compreso tra 0 e $n-1$.
Non va bene per (2) e (3)

esempio

(1) p primo e α è un intero con $p \nmid \alpha$.

$$h(x) \equiv \alpha^x \pmod p$$

Perché non va bene come funzione hash?
Perché è facile costruire le collisioni

$$h(x) \equiv h(x + p - 1)$$

infatti $h(x) \equiv \alpha^x \alpha^{p-1} \bmod p \equiv \alpha^x$

onde x è resistente alla pre-immagine e coll non facilmente invertibile

(2) Sia $n = pq$ prodotto di primi

$$h(x) = x^2 \pmod n$$

1 - è pre-image resistant?
è un problema complesso di ricerca delle radici quadrate equivalente alla fattorizzazione
2 - perché non è collision-free?
perché risulta $h(x) \equiv h(n-x) \quad \forall x \pmod n$

In definitiva $h_m = h(m)$

$|h_m| = b$-bit $\quad |m| = a$ bit

hash code
2^b codici hash

blocco messaggio
2^a possibili blocchi

funzione h_m: $y = h(m)$ one-way $\quad y$ = hash, m = messaggio

(1) **preimage resistance**

dato y è difficile trovare m tale che

$$y = h(m)$$

Attacco forza bruta #tentativi medi $= 2^{a-1}$

(2) **second preimage resistance**
(weak collision resistance)

dato m è difficile trovare m' tale che

$$y = h(m) = h(m') \text{ per } \forall m' \neq m$$

Attacco di forza bruta #tentativi medi $= 2^{a-1}$

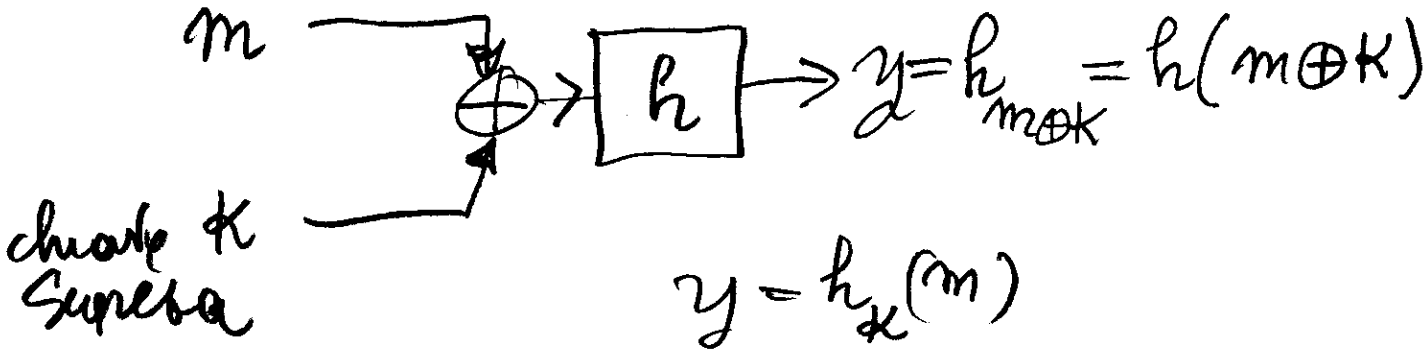
(3) **collision resistance**
(strong collision resistance)

è difficile trovare m e m' tali che

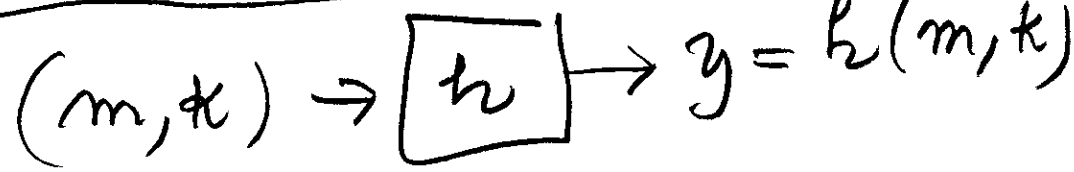
$$y = h(m) = h(m'), \text{ per } \forall m' \neq m$$

Attacco del compleanno #tentativi medi $\cong 2^{\frac{a}{2}}$

Posso mescolare un segreto nel codice hash (3ter)



Keyed-hash function



caso encatenato

- Per la funzione hash $y = h_m$

$y = h(m)$

vale la regola dell'"oracolo casuale" per cui

$$P\{h(m) = y\} = \frac{1}{2^b}$$

e nei tutti i codici hash se $h(m)$ è uguale di (1), (2) e (3) sono equiprobabili.

Funzione hash dei logaritmi discreti

Discrete Log Hash Function

p, primo grande tale che $(p-1) = 2 \cdot q$, con q primo abbastanza grande.
Scegliamo α e β radici primitive di $\mathbb{Z}_p$ tali che

$$(1)\quad \alpha^a \equiv \beta \pmod{p}$$

con a intero non noto: trovare a è computazionalmente difficile dovendo risolvere un problema di logaritmi discreti
La funzione hash h ha codominio così
Scrivo il messaggio m come

$$(2)\quad m = x_0 + x_1 q \quad \text{con} \quad 0 \le x_0, x_1 \le q-1$$

e la funzione $x_0, x_1 \in \mathbb{Z}_q$

$$(3)\quad h(m) \equiv \alpha^{x_0} \beta^{x_1} \pmod{p}$$

è strongly collision-free

Vale il claim: se conosciamo messaggi $m \neq m'$ con $h(m) = h(m')$, allora possiamo calcolare il logaritmo discreto $a = L_\alpha(\beta)$

La funzione (3) è resistente alla pre-immagine e collega interi mod q^2 con interi mod p:
il message digest è approssimativamente lungo la metà dei bit del messaggio. (Non si usa)

m mod q $\quad m = \lfloor \frac{m}{q} \rfloor q + r \quad x_0 = r \,;\, x_1 = \lfloor \frac{m}{q} \rfloor$

PROVA $\quad m = x_0 + x_1 q \qquad (x_0, x_1) \in \mathbb{Z}_q$

$m' = x_0' + x_1 q \qquad (x_0', x_1') \in \mathbb{Z}_q$

Supponiamo di aver trovato $m = m' \qquad (p-1) = 2q$

$$\alpha^{x_0} \beta^{x_1} \equiv \alpha^{x_0'} \beta^{x_1'} \pmod{p}$$

allora $\quad a(x_1 - x_1') - (x_0' - x_0)$

$$\alpha \equiv 1 \pmod{p}$$

$\alpha^k \equiv 1 \pmod{p}$ se e solo se $k \equiv 0 \pmod{p-1}$

$$(1) \quad a(x_1 - x_1') \equiv x_0' - x_0 \pmod{p-1}$$

i fattori di $(p-1)$ sono $1, 2, q, (p-1). \; (1, 2, q, 2q)$

Se $\underline{d} = mcd(x_1 - x', p-1)$ ci sono

$\underline{d}$ soluzioni della congruenza (1)

poiché $\quad 0 \le x_1, x_1' \le q-1$

$$-(q-1) \le x_1 - x_1' \le q-1$$

Se $(x_1 - x_1') \neq 0$ e quindi è un multiplo di $\underline{d}$

$d \neq q, 2q \; (= p-1)$

e quindi d o è 1 o è 2.

Ci sono quindi due soluzioni al massimo per la (1) provale tutte e due e trova quella che dà β.

Se $x_1 - x_1 = 0$ allora

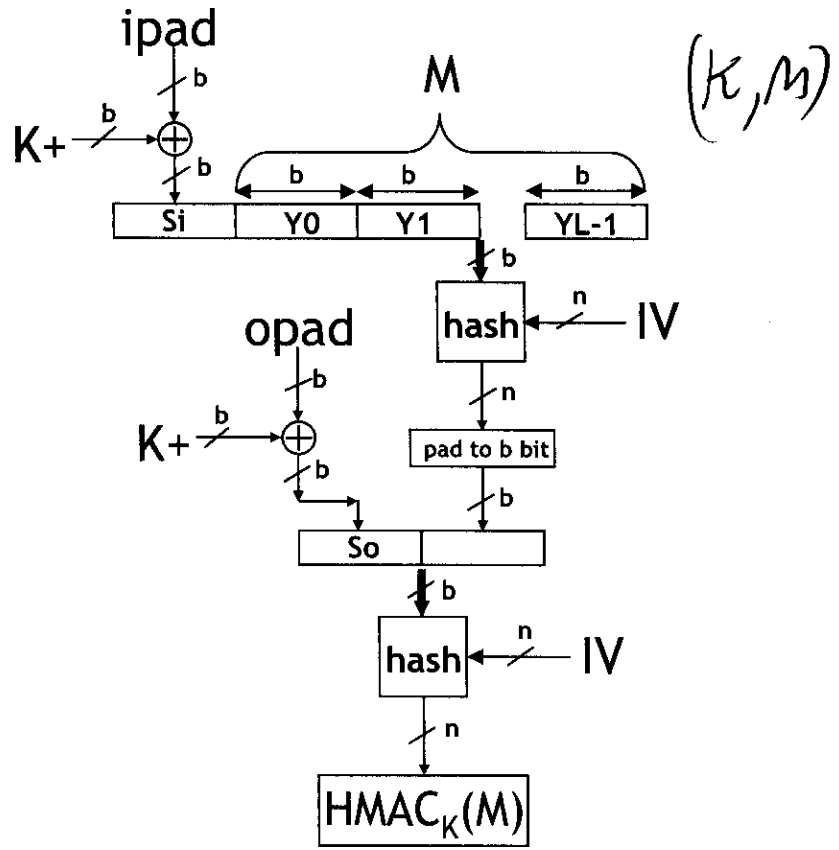
$$(1') \quad x_0' - x_0 \equiv 0 \pmod{p-1}$$

e quindi $x_0' \equiv x_0$ che proviene da $m = m'$, contrario alle nostre assunzioni

Hashed Message Authentication Code

- M, composto da L blocchi Yi ciascuno di b bit, es. b=512 bit, i=0, 1, 2, ..., L-1
- $HMAC_K(M)$, di n bit, es. n = 160 bit
- Hash + Secret key K (K lunga comunque)

  $HMAC_K(M) = h[So,h(Si,M)]$
- Ove
  - So= K+ ⊕ opad; Si= K+ ⊕ ipad
  - K+= K padded con 0s fino a b bit
  - ipad = 00110110 (36 hex) ripetuto b/8 volte
  - opad = 01011100 (5C hex) ripetuto b/8 volte



messaggio m di lunghezza L, diviso in blocchi di n-bit con $n \ll L$. Ciascun blocco è detto m_j

$$m = [m_1, m_2 \dots m_\ell] \qquad \ell = \left\lceil \frac{L}{n} \right\rceil$$

e il blocco m_j è padded per avere lunghezza n-bit.

$$1 \le j \le \ell \qquad m_j = [m_{j1}, m_{j2} \dots m_{jn}]$$

ove m_{ij} è un bit.

Se ruotiamo a sinistra il blocco di $(j-1)$ posizioni di bit per problema

$$1 \le j \le \ell \qquad m'_j = RL(m_j \leftarrow j-1)$$

Allineiamo i blocchi in colonne e XORiamo i bit delle colonne

allora da

$$M = \begin{bmatrix} m_{11} & m_{12} & & m_{1n} \\ m_{21} & m_{22} & & m_{2n} \\ & & & \\ m_{\ell 1} & m_{\ell 2} & & m_{\ell n} \end{bmatrix} \Rightarrow \begin{bmatrix} m_{11} & m_{12} & & m_{1n} \\ m_{22} & m_{23} & & m_{21} \\ m_{33} & m_{34} & & m_{32} \\ & & & \\ m_{\ell\ell} & m_{\ell,\ell+1} & & m_{\ell,\ell-1} \end{bmatrix}$$

$$h(m) = [c_1 c_2 \dots c_n] \quad \text{ove}$$

$$1 \le i \le n \qquad \begin{aligned} c_1 &= m_{11} \oplus m_{22} \oplus \dots \oplus m_{\ell\ell} \\ c_2 &= m_{12} \oplus m_{23} \oplus \dots \oplus m_{\ell,\ell+1} \\ &\vdots \\ c_n &= m_{1n} \oplus m_{21} \oplus \dots \oplus m_{\ell,\ell-1} \end{aligned}$$

HASH STANDARDS

Le funzioni SHA usano procedure iterative su n ROUND successivi. Si impiegano le tecniche precedenti, ma usano funzioni di compressione f che calcola il blocco corrente e il risultato del round precedente.

Si inizia con X_0 e poi $X_j = f(X_{j-1}, m_j)$

$1 \le j \le e$ $\quad X_e$ = hash code

Le funzioni di compressione $f_K(\text{Stringa}_1, \text{Stringa}_2 \dots \text{Stringa}_K)$ per esempio SHA-1 opera su stringhe di 32 bit e usa del tipo

$$f_3(B,C,D) = \begin{cases} (B \wedge C) \vee (\bar{B} \wedge D) \\ B \oplus C \oplus D \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) \end{cases}$$

dove $\wedge$ AND $\quad \oplus$ XOR

$\vee$ OR

il tutto modulo 2^{32}.

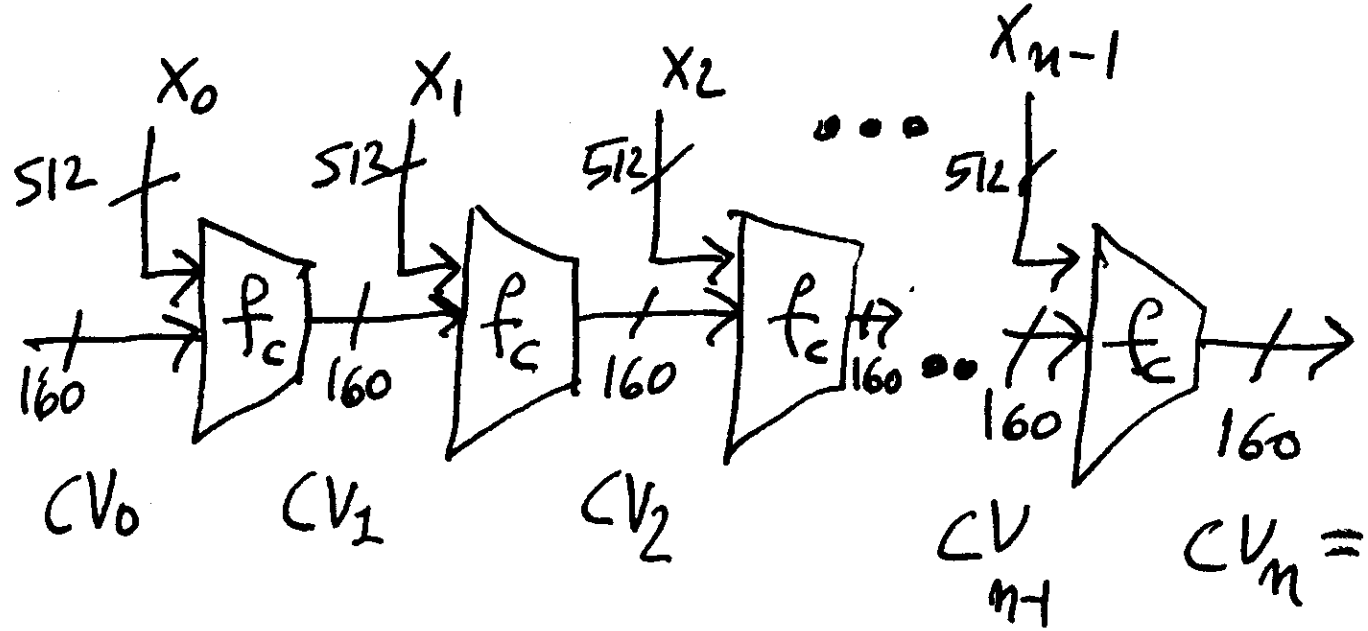
La funzione hash Whirlpool usa invece un algoritmo del tipo AES per fornire l'hash di un messaggio

Prima di SHA-1 e SHA-2 si usava anche DES e un codice hash più corto (MD4 e MD5).

	SHA-1	SHA-2		
		SHA-256	SHA-384	SHA-512
dimensione hash (bit)	160	256	384	512
dimensione blocco (bit)	512	512	1024	1024
dimensione messaggio (bit)	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$

Funzione di compressione f_c e hash iterativo

es. SHA-1



$$CV_i = f(X_{i-1}, CV_{i-1})$$

$$1 \le i \le (n-1)$$

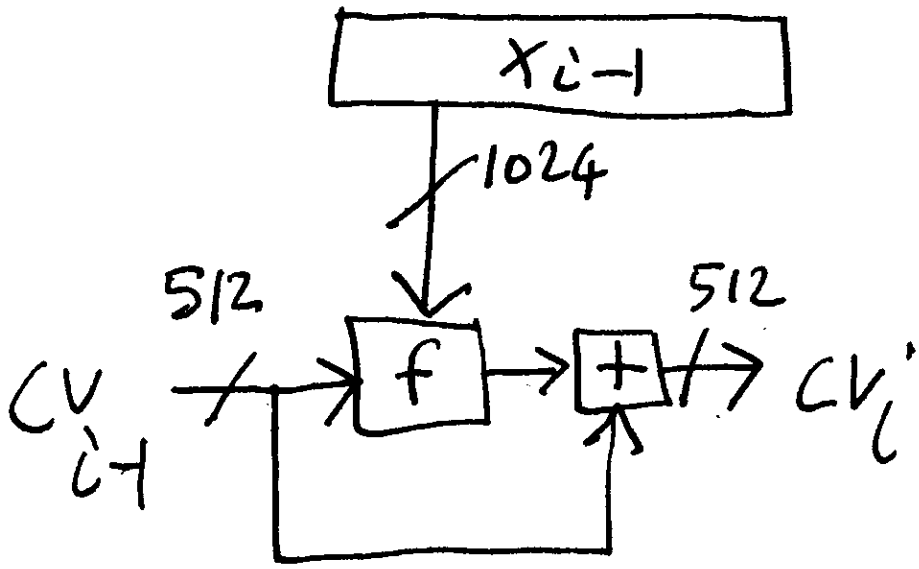
$CV_0 = IV$ vettore iniziale

messaggio

$$X \equiv (X_0, X_1, \dots X_{n-1})$$

$y = f_c(x)$ funzione di compressione

Caso SHA-512



+ mod 512

trtato

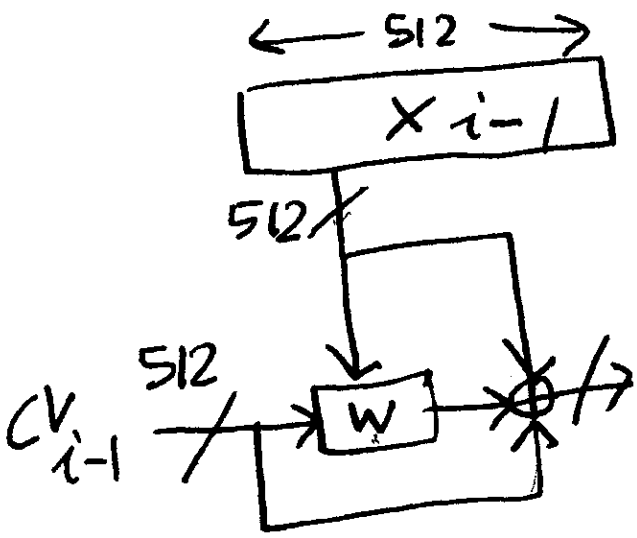
Whirlpool "idromassaggio"

hash

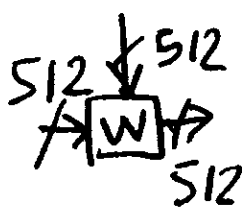
pad PADDING

$|x| \equiv 256 \mod 512$ 256 bit di meno

$b = 512 = a = 512$ $|x| \leq 2^{256} - 1$



single-stage hashing in Whirlpool



W BLOCK CIPHER

similare a AES (10 ROUND)

Attacco del compleanno

Se ci sono N oggetti, con N grande, e r persone e ciascuno sceglie un oggetto (con rimpiazzo: più persone fanno scegliere lo stesso oggetto), allora

$$P_{2\,\text{MATCH}} = P\{\text{che almeno due persone scelgono lo stesso oggetto}\} \gtrsim 1 - e^{-\frac{r^2}{2N}}$$

Se voglio $P_{2\,\text{MATCH}} \geq 0{,}5$: deve essere $e^{-\frac{r^2}{2N}} = e^{-\ln 2} = \frac{1}{2}$

allora $\frac{r^2}{2N} = \ln 2$ e $r \approx 1{,}177\sqrt{N} \approx \sqrt{N}$

$r \approx \sqrt{N}$

Se ci sono N possibilità, con una lista lunga $\sqrt{N}$ c'è una chance del 50% di avere almeno un match. Per aumentare le chance bisogna allungare la lista $2\sqrt{N}$; $3\sqrt{N}$; ... ecc..

Per esempio: se $N = 1000$ e $r = 40$

$$\frac{r^2}{2N} = 0{,}8 \text{ e } P_{2\,\text{MATCH}} \gtrsim 1 - e^{-0{,}8} \cong 0{,}55$$

Se h è un codice hash a n bit $N = 2^n$, crea una lista $h(x)$ per r scelte casuali di x con

$$r = \sqrt{N} = 2^{\frac{n}{2}}$$

almeno circa il 50% di probabilità di trovare due messaggi x_1 e x_2 con lo stesso valore $h(x_1) = h(x_2)$.

Se $$r = 10\sqrt{N} = 10 \cdot 2^{\frac{n}{2}}$$

la probabilità di trovare almeno una collisione è molto più elevata!

Similmente se prendiamo due liste per messaggi s e t e prendiamo $\sqrt{N}$ valori casuali di s e $\sqrt{N}$ valori casuali di t e calcoliamo

$$h(s) \quad e \quad h(t)$$

al 50% troviamo almeno una coppia (s^*, t^*) per cui si ha $h(s^*) = h(t^*)$

è il principio dell'attacco alle firme digitali se $n = 60$ ogni lista è lunga $2^{\frac{n}{2}} = 2^{30} \approx 10^9$ e in poche secondi si trova una collisione Per cui si impone $n \geq 128$

preferibilmente $n = 160; 192; 256; 512$.

ATTACCO DEL COMPLEANNO AI LOGARITMI DISCRETI

Va risolta la congruenza

$$\alpha^x \equiv \beta \pmod p$$

e cioè trovare

$$x = L_\alpha(\beta)$$

per p primo grande

Facciamo due liste di lunghezza circa $\sqrt{p}$

1. Prima lista contiene $\alpha^k \pmod p$ per $\sqrt{p}$ valori casuali di k
2. La seconda lista contiene $\beta \alpha^{-j} \pmod p$ per $\sqrt{p}$ valori casuali j

sorbiamo le due liste e c'è il 50% di probabilità di trovare un match (uguaglianza) tra alcuni elementi della prima lista e alcuni elementi della seconda lista

$$\alpha^{k} \equiv \beta\alpha^{-j} \quad : \quad \alpha^{k+j} \equiv \beta \pmod p$$

per cui $x \equiv k+j \pmod{p-1}$
è la soluzione.

ALGORITMO BABY STEP, GIANT STEP (BSGS)

Passo da banalmente, passo da giganti
è invece un algoritmo Deterministico per il calcolo dei logaritmi discreti la differenza dell'attacco del confronto che è probabilistico.
ha uso spazio di memoria anche lui proporzionale a $\sqrt{p}$, ma è anche più veloce nel calcolo degli esponenziali. se problema dei logaritmi discreti

$$\alpha^{x} \equiv \beta \pmod p$$

l'attaccante BSGS sceglie un intero N con $N^2 \geq p-1$
per esempio $N = \lfloor \sqrt{p-1} \rfloor + 1$ e fa due liste

1. $\alpha^{k} \pmod p$ per $0 \leq k < N$
2. $\beta\alpha^{-Nj} \pmod p$ per $0 \leq j \leq N$

trova un'eguaglianza (match) tra le due liste

$$\alpha^{k} \equiv \beta \alpha^{-Nj} \quad \text{per cui}$$

$$\alpha^{k+Nj} \equiv \beta$$

per cui $x = k + Nj$ e risolve il problema dei logaritmi discreti.

Perché deve essere un match? Poiché

$$0 \leq x < p-1 \leq N^2$$

---

scriviamo x in base N

$$x = x_0 + N x_1$$

ove $\begin{cases} x_1 = \left\lfloor \frac{x}{N} \right\rfloor \\ x_0 = x - N x_1 \end{cases}$ $\begin{cases} 0 \leq x_1 \leq N-1 \\ 0 \leq x_0 \leq N-1 \end{cases}$

$x \bmod N = r$ $\begin{cases} x = \left\lfloor \frac{x}{N} \right\rfloor N + r \\ x = x_1 N + x_0 \end{cases}$ $\begin{matrix} x_0 = r \\ x_1 = \left\lfloor \frac{x}{N} \right\rfloor \end{matrix}$

per cui $k = x_0$ e $j = x_1$

Si fa così: si calcola α^{k} per $k = 0, 1, 2, \ldots N$ con moltiplicazioni successive, poi si calcola $\beta \alpha^{-Nj}$ per $k = 0, 1, 2, \ldots$ e mi fermo quando trovo il match, confrontando di volta in volta con la prima

lista - Sfortunatamente se p ha 20 cifre decimali $N \cong \sqrt{p} \cong 10^{10}$ e richiede una memoria più molto grande - Non serve, così come il Birthday Attack, per p a 100 cifre decimali!

MULTICOLLISIONI

Se trovo più messaggi: $x_1, x_2, \ldots, x_k$, con lo stesso valore di hash trovo delle multicollisioni ($k>2$).

Se si ripete il calcolo del confronto

$$P_{k\,MATCH} = P\{\text{almeno } k \text{ collisioni}\} > 0{,}5$$

la probabilità di una k-collisione è del 50% se

$$r \cong N^{\frac{k-1}{k}}$$ k messaggi con lo stesso hash

se $k=2$ $$r \cong N^{\frac{1}{2}}$$ 2 messaggi con lo stesso hash

Allora per codici hash a n-bit, $N = 2^n$

$$r \cong 2^{n\frac{(k-1)}{k}}$$

è la lunghezza della lista per trovare con buona probabilità una k-collisione.

Attacco del Compleanno

Codici hash lunghi n bit; uno $N = 2^n$ oggetti equiprobabili

Probabilità di collisione $h_i = h_j$ per $m_i \neq m_j$

Numero di prove $= r$.

$1 \leq i, j \leq r$ per $i \neq j$

prove

Un numero dell'oggetto stesso.

Disposizioni (anche ripetizione) con e senza ripetizione di N oggetti a gruppi di r

N^r - disposizioni con ripetizione (si collisioni) (*)

$\frac{N!}{(N-r)!}$ - disposizioni senza ripetizione (no collisioni) (*)

(*) Se $\overbrace{h_i = h_j = h_k = \dots}^{k \text{ codici hash identici}}$; per $i \neq j \neq k \dots$

collisioni multiple = k-collisioni

2 codici $\rightarrow$ 2-collisioni = collisioni

k codici $\rightarrow$ k-collisioni

Allora

$$P\{\text{no collisioni}\} = \frac{\#\text{eventi senza collisioni}}{\#\text{eventi totali}} = \frac{\frac{N!}{(N-r)!}}{N^r} =$$

$$= \frac{(N-1)!}{N^{r-1}(N-r)!} = \frac{(N-r+1)\cdots(N-2)(N-1)}{N^{r-1}} =$$

$$= \prod_{i=1}^{r-1} \frac{N-i}{N} = \prod_{i=1}^{r-1} \left(1 - \frac{i}{N}\right) < \prod_{i=1}^{r-1} e^{-\frac{i}{N}} \approx e^{-\frac{r(r-1)}{2N}} \approx$$

($1 - x < e^{-x}$)

$$\approx e^{-\frac{r^2}{2N}}.$$

Quindi

$$P\{\text{almeno una collisione}\} = 1 - P\{\text{nessuna collisione}\} > 1 - e^{-\frac{r^2}{2N}} = x$$

se voglio almeno una collisione (due messaggi diversi con stesso codice hash) con probabilità $x = 0{,}5$ (50%) allora deve essere

infatti per $x = 0{,}5$ si ha: $e^{-\frac{r^2}{2N}} = 0{,}5$, cioè: $e^{\frac{r^2}{2N}} = 2$, allora $\frac{r^2}{2N} = \log_e 2$ e cioè $r = \sqrt{(2\ln 2)N}$

$$r \approx 1{,}177\sqrt{N} \approx \sqrt{N}$$

e cioè $r \approx 2^{\frac{n}{2}}$ cioè per forza bruta il numero delle prove è $\approx \sqrt{2^n}$.

Si ha poiché se voglio

$$P\{\text{almeno una } k\text{-collisione}\} > 0{,}5$$

allora $r \approx N^{\frac{k-1}{k}}$ e cioè

$$r \approx 2^{n\frac{k-1}{k}}$$

Se $k = 3$ $\quad r \approx 2^{\frac{2n}{3}} \approx \sqrt[3]{2^{2n}}$

$$C_A = A, K_A, \{h(A, K_A)\}_{K_{TA}^{-1}} \quad (1)$$

CERTIFICATO

Alice e Trusted Authority

scenario RSA $\quad m = 5 \times 13 = 65$

$$\varphi(m) = 4 \times 12 = 48 = 2^4 \cdot 3$$

$$\varphi(48) = 2^3 \cdot 2 = 2^4 = 16$$

$$\begin{cases} K_A = 5 \quad (5 \perp \varphi(m)) \\ K_A^{-1} = 5^{-1} = 5^{15} \bmod 48 = 29 \end{cases}$$

identità di A

$A \equiv 34 \bmod 65$

$$\begin{cases} K_{TA} = 11 \quad (11 \perp \varphi(m)) \\ K_{TA}^{-1} = 11^{-1} \equiv 11^{15} \equiv 35 \pmod{48} \end{cases}$$

si suppone che sia definita la funzione hash $z = h(x, y)$ STANDARD NOTO

così definita per numeri $x, y, z \in \mathbb{Z}_{65}$

(1) $z = (x \oplus \overline{y}) \wedge SL_2(x \vee y)$

Qual è il Certificato di Alice?

(2)

$$C_A = A, K_A, \{h(A, K_A)\}_{K_{TA}^{-1}}$$

$$= 34, 5, \{h(34,5)\}_{K_{TA}^{-1}}$$

ove $z = h(x,y)$ $\quad x = 34 \equiv 100010$

$y = 5 \equiv 000101$

$\bar{y} \equiv 111010$

$x \oplus \bar{y}$ $\quad 100010 \oplus$
111010
$011000 \rightarrow \alpha$

$x \vee y$ $\quad 100010 \vee$
000101
100111

$SL_2(100111) =$
$\equiv 011110 \rightarrow \beta$

$\alpha \wedge \beta$ $\quad 011000 \wedge$
011110
$h(x,y) = 011000 = 24$

$$C_A = (34, 5, 24^{35}) \bmod 65 \qquad 24^{35} \equiv 19$$

$$C_A \equiv 34, 5, 19 \pmod{65}$$

In maniera
chiunque prende 34 e 5, calcola lo standard hash(1)
e può verificare $h(34,5) = 19$ ok

Scambio delle chiavi di Diffie e Hellman

scelgo p primo tale che $p-1 = 2 \cdot q$ con q primo

scelgo poi $\alpha \in \mathbb{Z}_p^*$ radice primitiva di $\mathbb{Z}_p^*$

Alice sceglie x $\quad 1 \leq x \leq p-2$

Bob sceglie y $\quad 1 \leq y \leq p-2$

$x, y \in \mathbb{Z}_{p-1}$

$x, y \neq 0$

M1. $A \to B$: $\alpha^x \bmod p$

M2. $B \to A$: $\alpha^y \bmod p$

Bob calcola $K = (\alpha^x)^y \equiv \alpha^{xy} \pmod p$

Alice calcola $K = (\alpha^y)^x \equiv \alpha^{xy} \pmod p$

Problema D-H COMPUTAZIONALE

Dati $\alpha^x \bmod p$ e $\alpha^y \bmod p$ calcolare $\alpha^{xy} \bmod p$

Problema D-H DECISIONALE

Dati $\alpha^x \bmod p$ e $\alpha^y \bmod p$, e $c \not\equiv 0 \pmod p$

decidere se $c \equiv \alpha^{xy} \pmod p$

Diffie-Hellman D-H

$p = 107$ $\quad \frac{(p-1)}{2} = \frac{106}{2} = 53$ primo!

$(p-1) = 106 = 2 \times 53 = q_1 \times q_2$

$\alpha = 5$ primitivo $\mathbb{Z}^*_{107}$

$\alpha^{\frac{p-1}{q_1}} \not\equiv 1 \quad 5^2 \equiv 25 \pmod{107}$

$\alpha^{\frac{p-1}{q_2}} \not\equiv 1 \quad 5^{53} \equiv 106 \pmod{107}$

Alice sceglie $a_A = 23$

Bob sceglie $a_B = 3$

$a_A, a_B \in \mathbb{Z}_{p-1}; \neq 0 \quad q_A, q_B$

$(1 \leq a_A, a_B \leq p-2)$

Alice manda

M1. $A \to B$: $5^{23} \equiv 59 \pmod{107}$

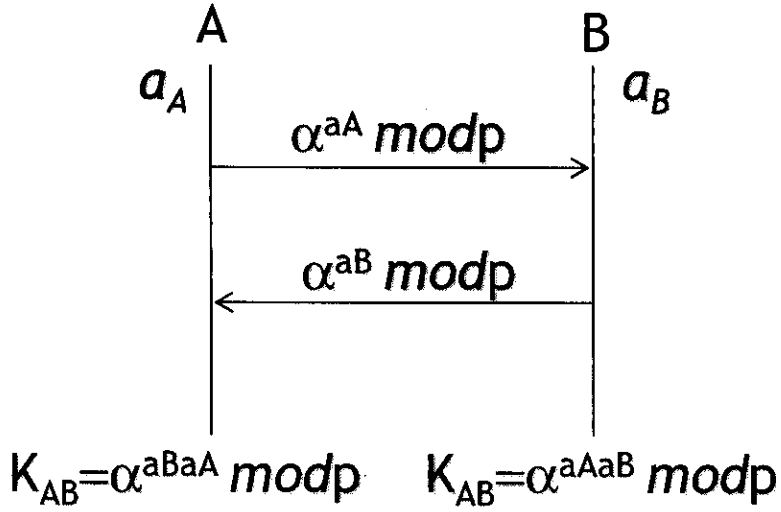
Bob calcola $59^3 \equiv 46 \equiv k_{AB} \pmod{107}$

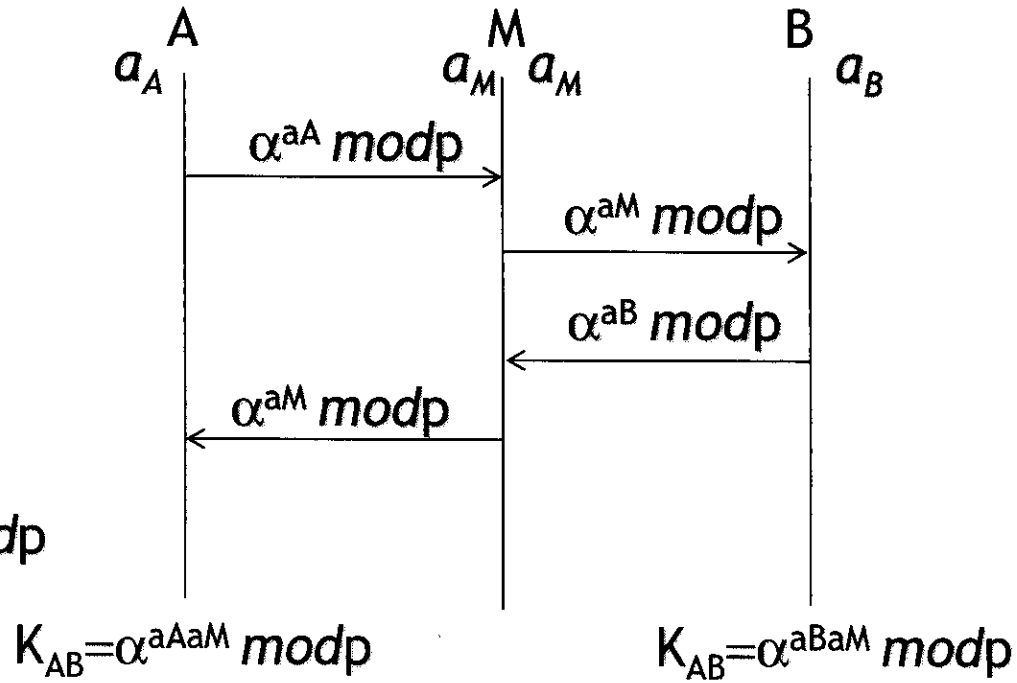
M2. $B \to A$: $5^3 \equiv 18 \pmod{107}$

Alice calcola $18^{23} \equiv 46 = k_{AB} \pmod{107}$

Diffie-Hellman

Dipartimento
Elettronica
Informazione





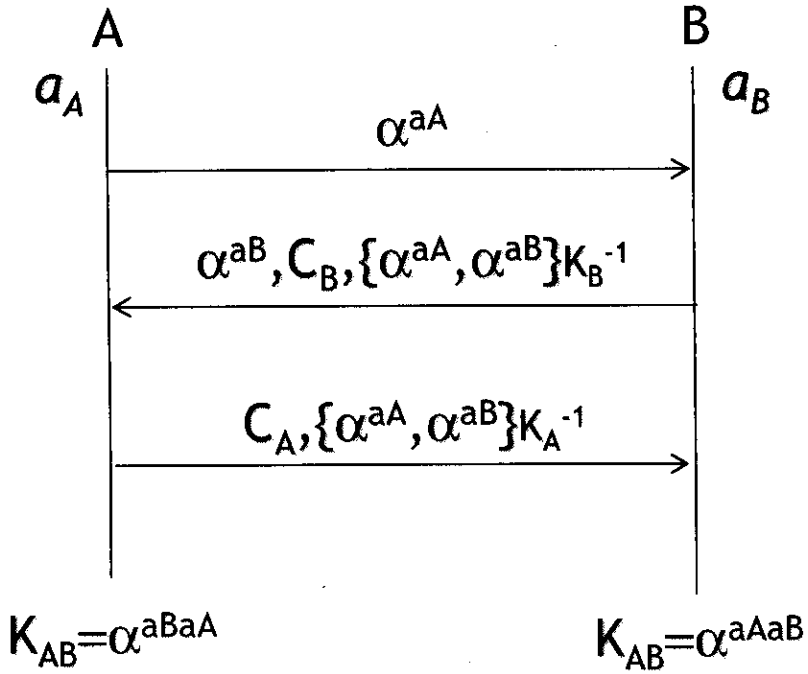
Dati: *p* primo grande,
(p-1)/2, ancora primo,
α elemento primitivo di Z_p^*, e
$0 < a_A, a_B \leq p-2$

MITM Attack

A e B parlano con Oscar, e credono di parlare tra loro!
FIX: certificati di identità

Station-to-Station Protocol

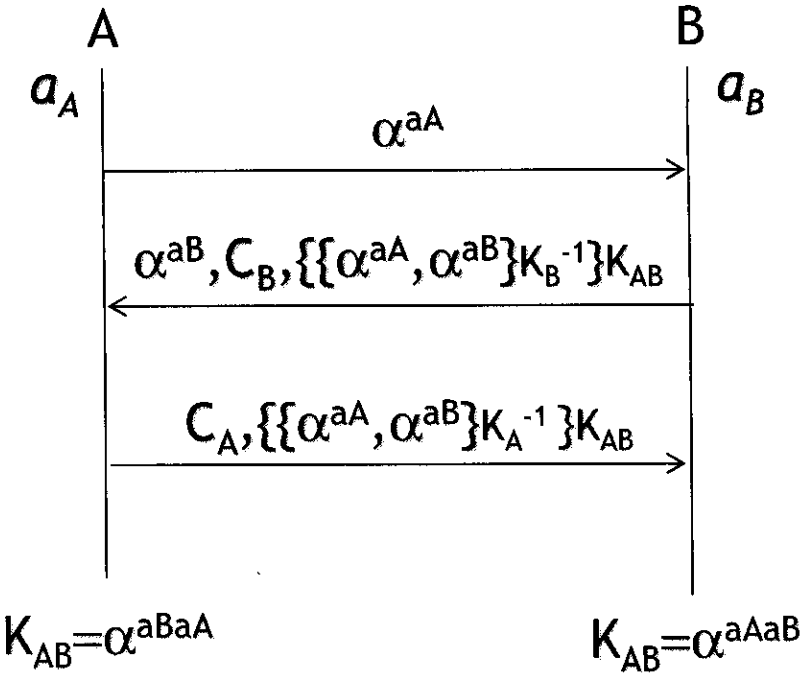
Dipartimento
Elettronica
Informazione



$C_B = B, K_B, \{h(B,K_B)\}K_{TA}^{-1}$

$C_A = A, K_A, \{h(A,K_A)\}K_{TA}^{-1}$

A e B mandano i loro certificati digitali e le firme delle porzioni della chiave



Con 'conferma' di possesso mutuo della chiave di sessione (doppia cifratura: da evitare!)

Crittosistema di Elgamal

p - primo ; α radice primitiva di $\mathbb{Z}_p$: $\alpha \in \mathbb{Z}_p$

$P \in \mathbb{Z}_p^*$; $1 \le P \le p-1$

intero $a \in \mathbb{Z}_{p-1}$: $a \neq 0$ $\quad 1 \le a \le p-2$

$\beta = \alpha^a \bmod p$: $\beta \in \mathbb{Z}_p^*$

CHIAVE PUBBLICA$_{BOB}$ $(p ; \alpha ; \beta)$

CHIAVE PRIVATA$_{BOB}$ (a)

$k \in \mathbb{Z}_{p-1}$: $k \neq 0$ $\quad 1 \le k \le p-2$

Alice sceglie k a caso e per una solo volta e spedisce a Bob la coppia (z, t)

$z, t \in \mathbb{Z}_p^*$

$$\begin{cases} z = \alpha^k \bmod p \\ t = \beta^k P \bmod p \end{cases} \qquad k \equiv \text{NONCE}_{\text{ALICE}}$$

Bob usa la sua chiave privata a e decifra

$$t z^{-a} \equiv P \pmod p$$

infatti

$$\beta^k P z^{-a} \equiv \alpha^{ak} P \alpha^{-ak} \equiv P \pmod p$$

essendo

$$\beta^k z^{-a} \equiv 1 \pmod p$$

esempio $p = 43 \quad \alpha = 3 \; ; \quad a = 10$

$\beta = 3^{10} \bmod 43$

$= 10 \bmod 43 \qquad k = 11$

$(p, \alpha, \beta) \equiv$

$\equiv (43, 3, 10)$

$15 = P \in \mathbb{Z}_p \qquad 1 \le k \le p-2$

Alice sceglie $k = 11$ e $P = 15$

e invia

$$\begin{cases} r = 3^{11} \bmod 43 = 30 \bmod 43 \\ t = (10^{11} \times 15) \bmod 43 = 10 \bmod 43 \end{cases}$$

$10^5 \bmod 43 = 25 \qquad (30, 10) \equiv (r, t)$

$25 \times 25 \times 10 \times 15 = 625 \times 150 = 93.750 \bmod 43 = 10$

$=$

Bob decifra

$$10 \cdot (30^{10})^{-1} \bmod 43$$

$30^{10} \bmod 43 = 15 \qquad 15^{-1} = 15^{41} \bmod 43 = 23$

$(10 \times 23) \bmod 43 = 15 \equiv P !$

Si osserva che k è un numero casuale e β^k è anche casuale. Quindi $t = \beta^k P$ è P moltiplicato per un numero casuale e quindi t non dà informazioni su P.

k è difficile da calcolare: lo faremo dopo.

È importante che k sia un NONCE usato una sola volta. Supponiamo che Alice cifra due messaggi P_1 e P_2 per Bob e usa lo stesso valore k, r sarà uguale e di due testi cifrati (r, t_1) e (r, t_2). Se Oscar trova P_1 egli può determinare P_2 con'

Poiché:

$$\frac{t_1}{P_1} \equiv \beta^k \equiv \frac{t_2}{P_2} \pmod p$$

Oscar conosce t_1 e t_2 e calcola

$$P_2 = \frac{t_2 P_1}{t_1} \pmod p -$$

Firme digitali

FIRMA RSA

Firma di un messaggio P

$n = p \times q$ Alice e_A ; d_A

firma $y = P^{d_A} \pmod n$

(P, y) (documento, firma del documento)

Bob conosce (e_A, n)

calcola $z \equiv y^{e_A} \pmod n$

se $z \equiv P$ Bob accetta la firma come valida
altrimenti è falsificata!

per l'attacco valgono le considerazioni nella sottostruttura di RSA.

- In questo caso Oscar sceglie a priori un codice $y = y_1$ e poi trova $P_1 \equiv y_1^{e_A} \pmod n$
Oscar può costruire (P_1, y_1) firmato da Alice è tuttavia molto probabile che P_1 non avrà senso e quindi Alice può facilmente notare che è falsificato!

FIRME CIECHE BLIND SIGNATURE

Il messaggio da firmare è P

1. Alice sceglie RSA mod n $(n = p \times q)$, esponente di cifratura "e" e decifratura "d" (n, e) pubblico (p, q, d) privato

2. Bob sceglie un intero casuale $k \pmod n$ con $\text{mcd}(k, n) = 1$ $\quad k \in \mathbb{Z}_n^* \quad k \perp n$ e calcola $\quad t = k^e P \pmod n$

e manda t a Alice

3. Alice forma t $\quad s \equiv t^d \pmod n$

e manda s a Bob

4. Bob calcola $\quad \frac{s}{k} \bmod n \equiv P^{ed} \pmod n$

il messaggio formato. Infatti

$$k^{ed} \equiv k \bmod n$$

$$\frac{s}{k} \equiv \frac{t^d}{k} \equiv \frac{k^{ed} P^d}{k} \equiv P^d \pmod n$$

k è casuale usato una sola volta

k^e è casuale anche

e quindi $\quad k^e P$ non dà informazioni su P

Alice non sa nulla sul contenuto del messaggio P che firma!

Firme cieche

Sono inattivi di attacchi dal fatto di basarsi sui protocolli impiegati

FIRMA ElGamal

più forte per lo stesso messaggio $P \in \mathbb{Z}_p$

- p primo grande; α primitivo di $\mathbb{Z}_p^* = \{1, 2, \dots p-1\}$
- a intero segreto: $1 \le a \le p-2$; $a \in \mathbb{Z}_{p-1}$ $a \neq 0$

$$\beta \equiv \alpha^a \bmod p \; : \; \beta \in \mathbb{Z}_p^*$$

- p, α, β Pubblici
- a segreto
- k numero segreto $k \in \mathbb{Z}_{p-1}^*$: $\mathrm{mcd}(k, p-1) = 1$

$$\text{Firma}_{ElG}(P, k) = (r, s)$$

$$\begin{cases} r \equiv \alpha^k \pmod p & : \; r \in \mathbb{Z}_p^* \\ s \equiv k^{-1}(P - ar) \pmod{p-1} & : \; s \in \mathbb{Z}_{p-1} \end{cases}$$

messaggio firmato (P, r, s)

Verifica

$$\text{Verifica}(P, r, s) = \begin{cases} v_1 \equiv \beta^r r^s \\ v_2 \equiv \alpha^P \end{cases} \pmod p$$

se $v_1 \equiv v_2 \bmod p$ allora la firma è valida, altrimenti è fraudolenta.

Infatti $\beta^r r^s \equiv \alpha^{ar} \cdot \alpha^{ks} \equiv \alpha^{ar+ks} \equiv \alpha^P$

poiché $ar + ks \equiv P \pmod{p-1}$

infatti per definizione

$$s \equiv k^{-1}(P - qr) \quad : \quad ks \equiv P - qr \pmod{p-1}$$

$$ar + ks \equiv P \pmod{p-1}$$

L'attacco ha la complessità dei log. discreti
k è un NONCE, da usare una sola volta.
Supponiamo che Alice usi k per P_1 e P_2 allora lo stesso r è usato nelle due firme e Oscar si accorge che k è lo stesso. I valori di s sono differenti: s_1 e s_2
Oscar sa che

$$s_1 k - P_1 \equiv -qr \equiv s_2 k - P_2 \pmod{p-1}$$

per cui

$$(1) \quad (s_1 - s_2)k \equiv P_1 - P_2 \pmod{p-1}$$

sia $d = mcd(s_1 - s_2, p-1)$: ci sono d soluzioni per la congruenza (1) e possono essere calcolate.
Usualmente d è piccolo e una di sono volte d valori per k. Oscar calcola α^k per ogni d soluzione di (1) finché non ottiene β. A quel punto Oscar ha trovato k e risolve

$$(2) \quad ar \equiv P_1 - k s_1 \pmod{p-1}$$

per a. Ci sono $mcd(r, p-1)$ soluzioni di (2)
Oscar calcola α^a per ognuna finché non ottiene β e finalmente trova "a" e può falsificare la firma di Alice.

Esempio di attacco del mese scorso — Forza Elgamal

$P_1 = 151,405$ $\quad p = 225,119$

$P_1 = \text{ONE}$ (hex) $\quad \alpha = 11$ radice primitiva

$a = ?$

Alice calcola $\beta = \alpha^a = 18,191 \mod p$

e sceglie $k = ?$

calcola $\begin{cases} r = \alpha^k = 164,130 \mod p \\ s_1 = k^{-1}(P_1 - ar) \equiv 130,777 \pmod{p-1} \end{cases}$

la tripla della forma è

$(151,405,\ 164,130,\ 130,777) \equiv (P_1, r, s_1)$

= Ora Alice usa lo stesso k per il messaggio

$P_2 = 202,315$ e la firma è $\quad P_2 \equiv \text{two}$ (hex)

$(202,315,\ 164,130,\ 164,899) \equiv (P_2, r, s_2)$

Oscar si accorge che r è lo stesso nelle due firme e che Alice ha usato lo stesso k, e scrive

(1) $\quad -34,122\,k \equiv (s_1 - s_2)k \equiv P_1 - P_2 \equiv -50,910 \pmod{p-1}$

poiché $\text{mcd}(-34,122,\ p-1) = 2$ a meno di soluzioni. Divido (1) per 2

$(p-1) = 225,118$

$\frac{p-1}{2} \equiv 112,559$

$-17061\,k \equiv -25455 \pmod{\frac{p-1}{2}}$

ha due soluzioni $k \equiv 239$ e $k \equiv 239 + \frac{p-1}{2} \equiv 112,798$

Calcola $\alpha^{239} \equiv 164,130 \quad \alpha^{112,798} \equiv 59,924 \pmod{p}$

allora $r = 164,130$ e $k = 239$! BINGO !

Oscar scrive poi $s_1 k \equiv P_1 - qr \pmod{p-1}$

$164{,}130\, q \equiv r q \equiv P_1 - s_1 k \equiv 187{,}104 \pmod{p-1}$

Poiché $\mathrm{mcd}(164{,}139, p-1) = 2$ ci sono due soluzioni $a = 28{,}862$ e $a = 141{,}421$

$\alpha^{28{,}862} \equiv 206{,}928$; $\alpha^{141{,}421} \equiv 18{,}191 \pmod{p}$

poiché $\beta = 18{,}191$ allora $a = 141{,}421$ Ott.

La firma di Elgamal è con appendice (P, r, s)

La firma RSA è uno schema di recupero del messaggio (y)

P si ricava da y e non va mandato $\quad P \equiv y^{e_A} \pmod{n}$

Hashing & Signing

la coppia $(P, sig(h(P)))$

è sicura se $h(P)$ è one-way e strongly collision free

Attacco del compleanno alle firme

"Prima di firmare un documento elettronico effettuare un piccolo cambio al documento stesso (ad esempio spazio aggiungere una virgola)"

Alice firma l'hash di un documento elettronico

$$sig_A(h(P))$$

(P è il contratto tra Alice e Bob)

$h(P)$ è lungo 50 bit $(b = 50 \text{ bit})$

Oscar effettua l'attacco del compleanno trova 30 punti del documento P ove può fare piccole modifiche. Così può costruire 2^{30} diversi documenti falsi P_i ma simili e non facilmente distinguibili dall'originale. Oscar calcola l'$h(P_i)$ di tutti i falsi 2^{30}

Nell'attacco del compleanno $r = 2^{30}$; $N = 2^{50}$ allora $r = \sqrt{2^{10} N}$ e quindi la probabilità che la versione corretta abbia hash uguale a quello di uno dei 2^{30} documenti falsi è $\left(\frac{r^2}{2N} = 2^{9}\right)$:→ $P\{\text{collisione}\} \approx 1 - e^{-512} \approx 1$

Oscar trova il match e chiede ad Alice di firmare la versione originale. Poi prende il P_i^* falso $h(P_i^*) = h(P)$ e gli offende

$$P_i^*, sig_A(h(P))$$

Per "fregare" Oscar, Alice che conosce questo attacco, prima di firmare l'originale, toglie una virgola e poi firma.

DIGITAL SIGNATURE ALGORITHM

DSA standard 1994

L'hash è di 160 bit
la firma è sul $h(P)$ di 160 bit che per semplicità chiamiamo $P \equiv$ messaggio

Fase di inizializzazione

1. Alice trova q primo a 160 bit e sceglie p primo tale che $q \mid p-1$. In genere p è di 512 bit o più lunghi
2. Sia g una radice primitiva mod p e sia
$$\alpha \equiv g^{\frac{p-1}{q}} \pmod p \quad \text{per cui}$$
$$\alpha^q \equiv 1 \pmod p$$
3. Alice sceglie il segreto a; $1 \leq a \leq q-1$: $a \in \mathbb{Z}_q^*$
e calcola
$$\beta \equiv \alpha^a \pmod p$$
4. Alice pubblica (p, q, α, β) e tiene segreto a

Firma

del messaggio P (in realtà è l'hash di P) lungo 160 bit

1. Sceglie il nonce k $\quad 1 \leq k \leq q-1 \quad k \in \mathbb{Z}_q^*$
2. Calcola $r \equiv (\alpha^k \pmod p) \pmod q$
3. Calcola $s = k^{-1}(P + ar) \pmod q$
4. La firma di P è (r, s)

Verifica Bob verifica (P, r, s)

1. Download p, q, α, β
2. calcola $\begin{cases} u_1 = s^{-1} P \mod q \\ u_2 = s^{-1} r \mod q \end{cases}$
3. calcola $v \equiv (\alpha^{u_1} \beta^{u_2} \pmod p) \pmod q$
4. Accetta la firma se e solo se $v = r$.

Infatti $P \equiv (-qr + ks) \pmod q$

$s^{-1} P \equiv (-qr s^{-1} + k) \pmod q$

Per cui $k \equiv s^{-1} P + qr s^{-1} \equiv$

$\equiv u_1 + q u_2 \pmod q$

per cui $\alpha^k \equiv \alpha^{u_1 + q u_2} \equiv (\alpha^{u_1} \beta^{u_2} \pmod p) \pmod q$

e quindi $v = r$.

(1) q è segreto, k è noto altrimenti dell'attacco come per Elgamal

(2) r non porta tutta l'informazione su k diversamente dal caso Elgamal - Se conosco r conosco il valore mod q. Ci sono circa $2^{512-160} = 2^{342}$ numeri mod p che si riducono a un prescritto numero mod q

(3) Perché usare $\alpha^q \equiv 1 \pmod p$ invece di una radice primitiva? q è grande e forte contro Pholig-Hellman - Nello schema Elgamal un'attaccante può determinare a (mod 2^t) dove 2^t è la potenza più grande che divide $p-1$. In DSA si ignorano tutte le informazioni su a tranne quelle mod q.

(4) →

Sia $p=11$ $q=5$ $\alpha=3$ $k=3$

si osserva che:

$$(\alpha^k \pmod p) \pmod q \neq (\alpha^k \pmod q) \pmod p$$

$$(3^3 \pmod{11}) \pmod 5 \equiv 5 \bmod 5 = 0$$

$$(3^3 \pmod 5) \pmod{11} \equiv 2 \bmod 11 = 2.$$

L'attacco alla forma di ElGamal di falsificazione esistenziale
existential forgery

verifica

→ (4) in DSA 2 esponenziali modulari

in ElGamal 3 esp. modulari

---

quindi DSA è meno costoso da verificare.

Firma di ElGamal

$$\begin{cases} p = 43 \\ a = 10 \\ k = 11 \end{cases} \qquad 1 \le 10 \le 41 \qquad \text{mcd}(11, 42) = 1$$

$p-1 = 42 = 2 \cdot 3 \cdot 7$ $\quad \alpha$ numero di $\mathbb{Z}_{43}^*$

$\varphi(p-1) = 12$

$\alpha = 3$

$\beta = 3^{10} = 10 \bmod 43$

$$\begin{cases} r = \alpha^k \bmod p = 3^{11} \bmod 43 = 30 \\ s_1 = k^{-1}(P_1 - ar) \bmod 42 = 39 \end{cases}$$

$k^{-1} \equiv 11^{11} \bmod 42 \equiv 23$

$= 23(15 - 10 \times 30) \bmod 42 = 39$

$P_1 = 15$

$P_2 = 20$

$$\begin{cases} v_{11} = \beta^r r^{s_1} \equiv 10^{30}\, 30^{39} = 22 \bmod 43 \\ v_{12} = \alpha^{P_1} \equiv 3^{15} \bmod 43 = 22 \bmod 43 \end{cases}$$

oppure

$v_1 = v_2$ oppure

$s_2 = 23(20 - 10 \cdot 30) \bmod 42 = -6\,440 \bmod 42$

$= -14 = 28$

$v_{21} = \beta^{s_2} r^{s_2} = 10^{30} \cdot 30^{28} = 41 \cdot 36 = 14 \bmod 43$

$V_{22} = \alpha^{P_2} = 3^{20} \bmod 43 = 14$ $\quad v_{21} = v_{22} = ok!$

Attacco del mittente ripetuto

attacco a k

$$(s_1 - s_2)k \equiv P_1 - P_2 \pmod{p-1}$$

$$(39-28)k \equiv -5 \pmod{42}$$

$$11k \equiv 37 \pmod{42}$$

$\mathrm{mcd}(11,42)=1$, allora ricavo:

$11 = \varphi(42) - 1$

$$k \equiv 11^{-1} \cdot 37 \bmod 42 \equiv 11$$

$$\equiv 11^{11} \cdot 37 \equiv 23 \cdot 37 \equiv 10 \pmod{42}$$

infatti $r = \alpha^k = 3^{11} \equiv 30 \pmod{43}$ BINGO!

infatti $r = \alpha^k = \beta^{11} = 30$

BINGO!

ora offro

$$ar \equiv P_1 - S_1 k \pmod{p-1}$$

$$30 \cdot a \equiv 15 - 39 \cdot 11 \pmod{42}$$

$\mathrm{mcd}(30, 42) = 6 = d$

$d \mid 6$ ok

$\equiv 15 - 429 \equiv -414 \equiv -36$

$\equiv 6$

$$30 \cdot a \equiv 6 \pmod{42}$$

allora a_0 è per $5a \equiv 1 \pmod 7$

$a_0 \equiv 5^{-1} \equiv 5^5 \equiv 3 \mod 7$

le altre 5 soluzioni sono $a_0 = 3$

$a_1 = a_0 + 7 = 10$

$a_2 = a_0 + 14 = 17$

$a_3 = a_0 + 21 = 24$

$a_4 = a_0 + 28 = 31$

$a_5 = a_0 + 35 = 38 \pmod{42}$

quindi @ con mora $a_1 = 10$

$\beta = \alpha^{a_1} \equiv 3^{10} \equiv 10 \pmod{43}$

BINGO!

FIRMA
DSA

in genere { p è a 512-1024 cifre decimali
q è a 159-160 cifre decimali

$q \mid p-1 \longrightarrow p-1 = kq$

g radice primitiva $\in \mathbb{Z}_p^*$ $\quad g \in \mathbb{Z}_p^*$

calcolo $\alpha \equiv g^{\frac{p-1}{q}} \pmod p$; $\alpha \in \mathbb{Z}_p^*$

perciò $\alpha^q \equiv 1 \pmod p$

$1 \le a \le q-1 \quad a \in \mathbb{Z}_q^*$; $a \equiv \{1, 2, \dots (q-1)\}$

$\beta = \alpha^a \pmod p$

ALICE PUBBLICA (p, q, α, β)

Forma di P

1. Alice sceglie k $\quad 1 \le k \le q-1$ $\quad [k \in \mathbb{Z}_q^* : \exists\, k^{-1}]$
2. calcola $r \equiv (\alpha^k \bmod p) \pmod q$
3. calcola $s = k^{-1}(P + ar) \pmod q$

Forma di P $\{P, (r, s)\}$ $\quad \exists\ k^{-1} \bmod q : mcd(k, q) = 1$

Bob verifica

calcola $\begin{cases} u_1 = s^{-1} P \bmod q \\ u_2 = s^{-1} r \bmod q \end{cases}$

e verifica se $v = (\alpha^{u_1} \beta^{u_2} \bmod p) \bmod q = r$

FIRMA VALIDA se $v = r$

---

esempio

$p = 43$

$p - 1 = 42 = 2 \cdot 3 \cdot 7$

$q = 7$

$\begin{cases} \frac{p-1}{2} = 21 \\ \frac{p-1}{3} = 14 \\ \frac{p-1}{7} = 6 \end{cases}$

$\alpha^q \bmod p = 4^7 \bmod 43 = 1$

$\alpha = 4$

$\alpha = 4 = 20^{\frac{42}{7}} = 20^6$

$g^{42} \equiv 1$

$g = 20$

$20^{21} \equiv -1 \not\equiv 1$

$20^{14} \equiv 36 \not\equiv 1$

$20^6 \equiv 4 \not\equiv 1$

Alice sceglie $1 \le a \le 6$: $a = 3$

Alice sceglie $1 \le k \le 6$!

$k^{-1} \equiv 4^{-1} \equiv 4^5 \equiv 2 \quad k = 4$

$2 \cdot 4 \equiv 8 \equiv 1 \bmod 7$

PUBBLICA
43, 7, 4, 21

$\beta = 4^3 \equiv 21 \pmod{43}$

$P = 5$

$$\begin{cases} r = 4^4 \bmod 43 = 41 \bmod 7 = 6 \\ s = 2(5 + 3 \cdot 6) \bmod 7 = \end{cases}$$

$$= 46 \bmod 7 = 4$$

FIRMA di Alice $\{5, (6, 4)\} \equiv \{P, (r, s)\}$

$s^{-1} \equiv 4^{-1} \equiv 4^5 \equiv 2$

$4 \cdot 2 \equiv 8 \equiv 1 \bmod 7$

Bob verifica

$$\begin{cases} u_1 = 2 \cdot 5 \bmod 7 = 3 \\ u_2 = 2 \cdot 6 \bmod 7 = 5 \end{cases}$$

$v = (4^3 \cdot 21^5 \bmod 43) \bmod 7 =$

$(21 \cdot 41 \bmod 43 \equiv 41 \bmod 43) = 41 \pmod 7 = 6 \equiv r$

OK !

DSA

$p-1 = k \cdot q$ k intero

Si osserva che $g \in \mathbb{Z}_p^*$ elemento generatore del campo finito (e ne sono $\varphi(p-1)$) e da questi si ricavano altrettanti

$$\alpha \equiv g^{\frac{p-1}{q}} \pmod p$$

questi $\varphi(p-1)$ α non sono generatori di $\mathbb{Z}_p^*$ infatti $\alpha = g^i$ con $i = \frac{p-1}{q}$, ma

$$\mathrm{mcd}\left(\frac{p-1}{F}, p-1\right) = q \neq 1$$

α infatti è di ordine q in $\mathbb{Z}_p^*$

$$\alpha^q \equiv 1 \pmod p$$

$q \ll p-1$

Esempio $p = 43$ $\quad p-1 = 42 = 6 \cdot 7$ $\quad q = 7 \ll p$

$g = 20$; $\alpha = 4$

$\mathrm{ord} g = 42$; $\mathrm{ord}\alpha = 7$

ma

$$\beta = \alpha^a \bmod p \quad \text{per } a \in \mathbb{Z}_q^* \quad 1 \leq a \leq q-1$$

α genera $q-1$ elementi $\beta \bmod p$ di valore numerico $\in \mathbb{Z}_p^*$ che costituiscono un sotto-campo finito del primo ordine

di $\mathbb{Z}_p^*$. $1 \le i \le 6$ campo palindromico "senza" pivot

esempio

α^i

$\alpha^1 \equiv 4^1 \equiv 4$ — i inversi

$4^2 \equiv 16$

$4^3 \equiv 21$

$4^4 \equiv 41$ $\pmod{43}$

$4^5 \equiv 35$

$4^6 \equiv 11$

$11 \equiv 4^{-1} \equiv 4^{41} \equiv 4^6$

$\mathbb{C}\mathbb{F}_{\text{ord}\,q}$ → è un campo finito, un gruppaccio di ciclo $= q$.

$4^7 \equiv 1$

4^8

sotto campo di $\mathbb{Z}_p^*$

quindi

$$\alpha^q \equiv \beta \in \mathbb{C}\mathbb{F}_{\text{ord}\,q} \in \mathbb{Z}_p^* \pmod{p}$$

tutti gli indici i di α^i $1 \le i \le q-1$ sono $\text{mcd}(i, q) = 1$ e quindi tutti i $q-1$ elementi del campo sono generatori (campo del primo ordine).

Ad esempio $\alpha \equiv 21$

$21^1 \equiv 21$

$21^2 \equiv 11$

$21^3 \equiv 16$

$21^4 \equiv 35$

$21^5 \equiv 4$

$21^6 \equiv 41$

$21^7 \equiv 1$

$\mathbb{C}\mathbb{F}_{\text{ord}\,q}$ ordine dispari $\pmod{43}$

Ricordo che l'ordine del campo $\mathbb{Z}_p^*$ è $\text{ord}(p-1)!$

ordine pari (con "pivot")