Sistemi di Elaborazione: Sicurezza su Reti **Anno Acc. 1998-99**
PROF. ALFREDO DE SANTIS

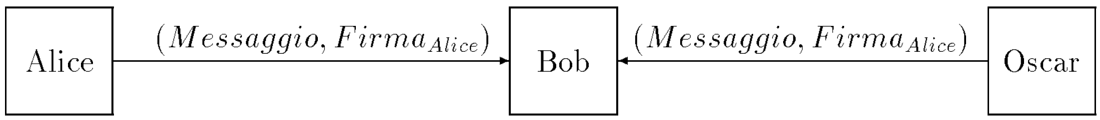
FIRME DIGITALI

a cura di R. De Rito, F. Laino, S. Passariello e M. Torella *22 marzo 1999*

Indice

1 Introduzione

In molte circostanze che si presentano quotidianamente, come prelievo di soldi, firme di contratti, firme di documenti legali, etc. una firma *convenzionale* scritta a mano viene apposta su un documento. Tale firma, come di consueto, serve per specificare la persona responsabile del documento che è stato firmato. Ciò che noi tratteremo in questa sezione sono gli *schemi di firme*, anche noti come firme digitali, i quali consentono di poter firmare documenti o messaggi memorizzati in forma elettronica. Nel prosieguo della trattazione indicheremo con 'Alice' e 'Bob' due utenti che vogliono scambiarsi dei messaggi firmati, e con 'Oscar' un utente (nemico) che cerca di falsificare una firma.



Alice vuole mandare un messaggio M a Bob e in qualche modo insieme a questo messaggio M vuole produrre una certa firma elettronica per indicare che quel messaggio è proprio il suo.

Ciò sappiamo facilmente farlo se la cosa funziona su supporto cartaceo, infatti se abbiamo un certo documento allora alla fine apponiamo la nostra firma, quello che vorremmo fare è la stessa cosa se anzichè avere un supporto cartaceo avessimo un file, cioè una sequenza di bit.

La prima idea che viene in mente è che potrei digitalizzare la firma di Alice e appenderla alla fine di ogni messaggio M da firmare.

Questo metodo non funziona perchè essendo la firma una sequenza di bit, chiunque può tagliare quest'ultima parte e appenderla a qualunque altro file, firmando il documento illegalmente.

Questo dipende essenzialmente perchè la firma così prodotta è indipendente dal messaggio.

Le proprietá che noi vorremmo da una firma elettronica sono :

1. La firma digitale deve poter essere prodotta solo da Alice.

2. Nessun altro utente deve poter riprodurre la firma di Alice.

3. Chiunque deve poter essere in grado di verificare che la firma sia di Alice.

Quindi nasce l'esigenza di individuare schemi di firma che siano sicuri da possibili attacchi, i quali, se efficaci, potrebbero dare adito, ad un nemico, di falsificare delle firme.

Studieremo diversi schemi di firme, ma è opportuno dapprima discutere alcune fondamentali differenze tra firme *convenzionali* e firme digitali.

La prima differenza concerne le modalità con cui un messaggio viene firmato: nella *firma convenzionale*, la firma è parte integrante del messaggio, mentre con la firma digitale essa non è attaccata fisicamente al messaggio da firmare, così l'algoritmo, che viene usato per firmare, la deve in qualche modo "legare" al messaggio. In quest'ultimo caso il messaggio firmato è costituito dalla coppia (Messaggio,Firma).

La seconda differenza riguarda la verifica della firma. Una firma *convenzionale* è verificata confrontandola con un'altra firma autenticata. Certamente questo non è un metodo sicuro poichè qualcuno potrebbe facilmente falsificare la firma.

Le firme digitali possono essere verificate usando un algoritmo di verifica pubblicamente noto, così "ognuno" può verificare una firma digitale. Un'altra differenza fondamentale è che una copia di un messaggio digitale firmato è identica all'originale, mentre una copia di un documento firmato può di solito essere distinta dall'originale.

Questo ci fa capire che uno schema di firme digitali, deve essere tale da prevenire il possibile riutilizzo di un dato documento. Per esempio, se Alice firma un messaggio digitale col quale autorizza Bob a prelevare \$ 1000 dal suo conto bancario, allora desidera che tale operazione venga eseguita una sola volta, così il messaggio dovrebbe contenere delle informazioni, tale come la data, che ne impedirebbe il suo riutilizzo.

Dalle considerazioni fatte emerge che i due elementi fondamentali in uno schema di firma sono: un *algoritmo di firma* e un *algoritmo di verifica*. Alice può firmare un messaggio M usando un algoritmo di firma *sig* che usa alcune informazioni segrete che sono note solo a Alice. La firma risultante $y = sig_K(M)$ può successivamente essere verificata usando l'algoritmo di verifica pubblico ver_K, il quale data una coppia *(M, F)* dove M è il messaggio ed F è la firma, ritorna una risposta "vero" o "falso" rispettivamente, a secondo che la firma sia autentica oppure no. Tale verifica viene realizzata grazie a delle informazioni che Alice ha memorizzato in un apposito file pubblico e, condizione fondamentale è che da queste informazioni non deve essere possibile, ovviamente, poter ricavare le informazioni segrete di Alice. Una definizione formale di uno schema di firme è la seguente:

Definizione 1.1 *Uno schema di firme digitale è una quintupla* $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$*, dove le seguenti condizioni sono soddisfatte:*

1. $\mathcal{P}$ *è un insieme finito di possibili messaggi*
2. $\mathcal{A}$ *è un insieme finito di possibili firme*
3. $\mathcal{K}$*, lo spazio delle chiavi, è un insieme finito di possibili chiavi*
4. *Per ogni* $K \in \mathcal{K}$ *c'è un algoritmo di firma* $sig_K \in \mathcal{S}$ *e un corrispondente algoritmo di verifica* $ver_K \in \mathcal{V}$. *Ogni* $sig_K : \mathcal{P} \rightarrow \mathcal{A}$ *e* $ver_K : \mathcal{P} \times \mathcal{A} \rightarrow \{vero, falso\}$ *sono funzioni tali che la seguente equazione è soddisfatta per ogni messaggio* $M \in \mathcal{P}$ *e per ogni firma* $F \in \mathcal{A}$*:*

$$ver_K(M, F) = \begin{cases} vero & se\ F = sig_K(M) \\ falso & se\ F \neq sig_K(M). \end{cases}$$

Per ogni $K \in \mathcal{K}$ le funzioni ver_K e sig_K devono essere funzioni facili da calcolare per una maggiore efficienza dello schema, mentre dovrebbe essere computazionalmente difficile per Oscar "falsificare" la firma di Alice su di un messaggio M. Cioè, dato M, solo Alice deve essere in grado di calcolare la firma F tale che $ver_K(M, F) = vero$.

Uno schema di firma è detto incondizionatamente sicuro se, qualunque siano le risorse (tempo e hardware) di Oscar, è impossibile falsificare le firme. Uno schema di firme non può essere incondizionatamente sicuro, poichè Oscar potrebbe testare tutte le possibili firme y per un messaggio M usando l'algoritmo pubblico ver_K, fino a quando egli trova la giusta firma. Naturalmente questo tipo di attacco risulta essere computazionalmente oneroso soprattutto se l'insieme delle firme è molto grande. Ma, dato sufficiente tempo, Oscar può sempre riuscire a falsificare la firma di Alice, così il nostro scopo è di individuare schemi che siano computazionalmente sicuri.

Nelle successive sezioni tratteremo diversi schemi di firme quali RSA, ElGamal, DSS, Lamport e Bos-Chaum.

2 Schema di firme RSA

Il crittosistema a chiave pubblica RSA [5] può anche essere usato per fornire firme digitali. Alice può firmare un messaggio M usando la regola di decifrazione d_K dell'RSA, ed è la sola persona che può creare la firma poichè $d_K = sig_K$ è segreta.

Bob invece usa la regola di cifratura e_K dell'RSA in quanto $e_K = ver_K$ è pubblica.

2.1 Parametri del RSA

La chiave é $K = \{n, p, q, d, e\}$ dove: p e q sono due primi scelti a caso, $n = pq$, d ed e sono due interi tali che $ed \equiv 1 \bmod \phi(n)$. Le coppie (n, d) e (n, e) sono rispettivamente la chiave privata e la chiave pubblica. Inoltre $\mathcal{P} = \mathcal{A} = Z_n$.

2.2 Generazione dei parametri

Alice per prima cosa genera le chiavi pubblica e privata: la chiave privata la utilizza per firmare il messaggio M, mentre la pubblica viene inviata ad un'autorità fidata per gli utenti che desiderano verificare la firma. Schematicamente:

1. Alice genera due primi p e q.
2. Calcola $n = pq$.
3. Genera due interi e, d tali che $ed \equiv 1 \bmod \phi(n)$.
4. Pubblica la coppia (n, e) come sua chiave pubblica.
5. Tiene nascosta la coppia (n, d) come sua chiave segreta.

2.3 Generazione della firma

Algoritmo di firma RSA

La firma F di un messaggio M è :

$$sig_K(M) = M^d \bmod n$$

Si è assunto che $0 \leq M < n$, infatti questa condizione è necessaria affinchè non si abbiano due messaggi distinti M, M' con la medesima firma. Nella prossima sezione vedremo come firmare messaggi M maggiori di n.

Si noti che la grandezza della firma di un qualsiasi messaggio M non può essere più grande di M stesso, infatti dopo aver effettuato l'esponenziazione la firma F è computata modulo n quindi $0 \leq F < n$.

Le proprietà che avevamo richiesto da una firma digitale sono soddisfatte, infatti:

- Per Alice è facile calcolare la firma, infatti deve soltanto calcolare un'esponenziazione modulare, che è possibile effettuare in modo efficiente.
- La verifica può essere effettuata da chiunque poichè la coppia (n, e) è pubblica.
- La firma è "difficile" da falsificare per un nemico, perchè ciò significa rompere il crittosistema a chiave pubblica RSA.

2.4 Procedura di verifica

Per verificare la firma F del messaggio M, Bob deve conoscere la chiave pubblica (n, e) di Alice.

> **Algoritmo di verifica RSA**
>
> La verifica della firma F di un messaggio M è :
>
> $$ver_K(M, F) = \begin{cases} vero & \text{se } M = F^e \bmod n \\ falso & \text{altrimenti.} \end{cases}$$

Verifichiamo adesso la correttezza della procedura di verifica:
dato che $F = M^d$ allora si ha che

$$\begin{aligned} F^e \bmod n &= (M^d)^e \bmod n \\ &= M^{ed} \bmod n \\ &= M \bmod n \qquad \textit{poichè} \qquad ed \equiv 1 \bmod \phi(n) \\ &= M \qquad \textit{poichè} \qquad 0 \leq M < n. \end{aligned}$$

2.5 Firma di messaggi di lunghezza arbitraria

Valutiamo il caso in cui il messaggio M è di lunghezza superiore ad n. Firmare direttamente messaggi di tale lunghezza può portare ad una collisione di firme, cioè per messaggi distinti otteniamo la medesima firma. Questo perchè, avendo $\mathcal{P}$ una cardinalità maggiore di $\mathcal{A}$, non esiste una corrispondenza biunivoca tra l'insieme dei messaggi e l'insieme delle firme. Per ovviare a tale problema esistono due modalità di firma:

1. suddividere il messaggio M in diverse parti di lunghezze minore o uguale ad n, e poi firmare ogni singola parte;
2. utilizzare una funzione hash one-way che, dato M, restituisce $h(M)$ avente lunghezza minore o uguale ad n.

Analizziamo ora il primo caso: se abbiamo un messaggio M maggiore di n possiamo dividerlo in blocchi di lunghezza minore di n e firmare ogni singolo blocco. Con tale metodo vi sono però alcuni inconvenienti. Infatti bisogna fare tante firme quanti sono i blocchi che compongono il messaggio a discapito dell' efficienza. Un secondo inconveniente è che ogni firma si riferisce ad un unico blocco, quindi ogni blocco è una parte a se stante della firma. Questo potrebbe dare adito ad un nemico di scambiare tra loro i blocchi ottenendo così la firma di un nuovo messaggio. Infatti supponiamo di avere due messaggi differenti $M = B_1 \cdots B_t$ $M' = C_1 \cdots C_k$ tale che $|B_i| = |C_j|$ in cui in uno di questi Alice autorizza Bob a prelevare \$ 100.000 e che tale cifra sia contenuta nel blocco B_i, ed in un secondo blocco C_j di M' ci sia la cifra \$ 1.000.000, allora Bob scambiando il blocco B_i con C_j nel messaggio in chiaro e i corrispondenti blocchi della firma riesce ad autorizzarsi a prelevare \$ 1.000.000. Quindi un sistema di questo tipo può essere facilmente attaccabile scambiando semplicemente i blocchi del messaggio. Analiziamo ora il secondo caso: l'idea è quella di comprimere il messaggio originale ad una grandezza minore di n, tale compressione viene fatta usando una funzione hash one-way. Applicando la funzione h al messaggio M mi calcolo M' e quindi non firmo più M ma M'. In tale modo ho risolto due problemi : un problema è che non posso fare più il gioco di prima sostituendo i blocchi, e l'altro è che anzichè fare più firme ne faccio una soltanto.

Analizziamo brevemente le proprieta della funzione hash:

- Dato M, è facile calcolare $h(M)$.
- È difficile trovare due messaggi M ed M', tali che $h(M) = h(M')$.

L'ultimo punto è la proprietà più forte delle funzioni hash, infatti supponiamo che ci siano due valori M_1, M_2 tali che $h(M_1) = h(M_2)$ allora conoscendo la firma di M_1 potrei firmare il messaggio M_2 illegalmente. La tecnica della funzione hash è la più utilizzata.

2.6 Possibili attacchi allo schema di firma RSA

La sicurezza dello schema di firme RSA si basa in gran parte sulla difficoltà della fattorizzazione di interi molto grandi. Infatti se un nemico riuscisse a fattorizzare n potrebbe calcolarsi il valore segreto d da e in quanto e è l'inverso moltiplicativo di $d \bmod \phi(n)$ dove $\phi(n) = (p-1)(q-1)$. Per raggiungere una buona sicurezza nello schema di firme RSA bisogna, quindi, necessariamente lavorare con interi di 512 bit o 1024 bit se si vuole una sicurezza ancora maggiore.

I possibili attacchi di Oscar allo schema di firme RSA sono i seguenti:

- Oscar cerca di falsificare la firma F tentando di calcolare la chiave segreta d.
- Oscar cerca di falsificare la firma di Alice su un messaggio casuale calcolando prima la firma F e poi $M = e_k(F)$.

Il primo attacco equivale a rompere il crittosistema a chiave pubblica RSA il che è un problema che si suppone molto “difficile” da risolvere.

Per evitare attacchi del secondo tipo ci sono due modi: uno è quello di introdurre una ridondanza nei messaggi che firmo, quindi sono validi solo un sottoinsieme ristretto di tutti i messaggi possibili.

Quando si fa un attacco del secondo tipo si deve allora computare qualche elemento del sottoinsieme di messaggi. La probabilità di tale evento è bassissima se il sottoinsieme dei messaggi validi è piccolo.

Un'alternativa consiste nella scelta di una funzione *hash one-way*, come già descritto nella sezione 2.5.

Se Alice desidera firmare un messaggio M, applica la funzione hash ad M, poi firma $h(M)$, cioè $F = sig_K(h(M))$, e quindi spedisce a Bob la coppia (M,F). Bob può, allora, verificare la firma calcolando dapprima $h(M)$ e successivamente utilizzare l'algoritmo di verifica ver_K su $h(M)$ e F. In questo modo se Oscar volesse falsificare un messaggio "casuale" M non noto, seguendo il secondo tipo di attacco, ciò che riesce a valutare è una firma su $h(M)$.

La difficoltà maggiore che gli si presenta successivamente è quella di dover ricavare da $h(M)$ il messaggio M. Ovviamente tale operazione è difficile da compiere data la complessità nell'invertire una funzione hash one-way. In definitiva, la scelta di una funzione hash one-way rafforza notevolmente la sicurezza del nostro schema di firme.

3 Schema di firme ElGamal

Lo schema ElGamal [1][2], proposto nel 1984, è stato progettato specificatamente per scopi di firme, a differenza dell'RSA, il quale può essere usato sia come crittosistema, sia come schema di firme.

Lo schema ElGamal è non deterministico; questo significa che esistono molte firme valide per un generico messaggio, e ciò dipende, come vedremo nella descrizione dello schema, dalla scelta di un particolare valore segreto r.

In virtù di quest'ultima caratteristica, l'algoritmo di verifica deve essere in grado di poter riconoscere, come autentica, una qualunque firma valida.

3.1 Generatori di Z_n^* e proprietà

In questa sezione definiamo il concetto di generatore del gruppo moltiplicativo Z_n^* ed alcune sue proprietà che saranno utili anche per lo schema di firme DSS. Dalla teoria dei numeri la cardinalità del gruppo moltiplicativo Z_n^* è $\phi(n)$, dove $\phi(n)$ è detta funzione di Eulero, in particolare la cardinalità di Z_p^* è $\phi(p) = p - 1$ dove p è un numero primo. Preso un elemento α qualsiasi in Z_n^*, l'ordine di α è il più piccolo intero positivo r tale che $\alpha^r \equiv 1 \bmod n$. Se, in particolare, l'ordine di $\alpha \in Z_n^*$ è $\phi(n)$, ogni elemento di Z_n^* è esprimibile come una potenza di α.

Definizione 3.1 *Un gruppo G è* **ciclico** *se esiste $\alpha \in G$ tale che per ogni $b \in G$ esiste un intero i tale che $b=\alpha^i$. Tale elemento α è chiamato* **generatore** *di G.*

Data, quindi, la definizione di gruppo ciclico segue che, preso $\alpha \in Z_n^*$, se l'ordine di α è $\phi(n)$, allora α è detto generatore o elemento primitivo di Z_n^*. Se Z_n^* ha un generatore, allora Z_n^* è detto ciclico.
Diamo alcune proprietà dei generatori di Z_n^*:

1. Z_n^* ha un generatore se e solo se $n = 2, 4, p^k, 2p^k$, dove p è primo e $k \geq 1$. In particolare, se p è primo, allora Z_p^* ha un generatore.

2. Se α è un generatore di Z_n^*, allora Z_n^*=$\{\alpha^i$ mod n tale che $0\leq i\leq n-1\}$.

3. Supponiamo che α sia un generatore di Z_n^*. Allora $b\equiv\alpha^i$ mod n è ancora un generatore di Z_n^* se e solo se gcd(i, $\phi(n)$)=1. Poichè Z_n^* è ciclico, allora il numero di generatori è $\phi(\phi(n))$.

3.2 Parametri di ElGamal

La chiave è $K = \{p, \alpha, s, \beta\}$ dove: p è un numero primo, α è un generatore in Z_p^*, s è un valore scelto a caso in Z_{p-1}^*, $\beta \equiv \alpha^s \bmod p$. La terna (p, α, β) è la chiave pubblica mentre la chiave privata è formata dagli stessi valori più l'aggiunta del valore s. Inoltre $\mathcal{P} = Z_p^*$, $\mathcal{A} = Z_p^* \times Z_p^*$.

3.3 Generazione della firma

Algoritmo di firma ElGamal

La firma F di un messaggio M è :

1. Sceglie a caso un valore $r \in \mathbf{Z}_{p-1}^*$.
2. Calcola $\gamma = \alpha^r \bmod p$.
3. Calcola $\delta = (M - s\gamma)r^{-1} \bmod (p-1)$.

$$sig_K(M, r) = (\gamma, \delta)$$

Si noti che $0 \leq \gamma < p-1 \quad$ e $\quad 0 \leq \delta < p-1$.

Quindi la firma è una coppia (γ, δ) scelta in modo tale che l'equazione

$$\alpha^M \equiv \beta^\gamma \gamma^\delta \bmod p$$

sia soddisfatta. La cosa da notare in questo schema è che un messaggio può avere più firme, poichè i valori γ e δ sono casuali essendo r scelto a caso.

Nel passo (2), i valori che possiamo dare a r sono $0, 1, ..., p-1$ in quanto, essendo α un generatore di Z_p^*, l'ordine di α è $p-1$ e quindi, scegliendo un valore $r > p-1$, α^r sarà uguale a qualche elemento dell'insieme $\{\alpha^0, \alpha^1, ..., \alpha^{p-1}\}$.

Nel passo (3), l'inverso di $r \bmod p-1$ deve esistere necessariamente poichè $r \in Z_{p-1}^*$ e l'inverso di r mod $p-1$ esiste se e solo se r e $p-1$ sono relativamente primi, ma per definizione Z_{p-1}^* è l'insieme degli elementi relativamente primi rispetto a $p-1$.

Un problema di tale schema è che la firma del messaggio sarà di 1024 bit, infatti sia γ che δ sono dell'ordine di 512 bit, se viene scelto un p di 512 bit. Avere firme corte è conveniente in sistemi dove occorre minimizzare la memoria, per esempio le smart-card.

3.4 Procedura di verifica

Prima di verificare la firma del messaggio i valori γ e δ più il messaggio M devono essere noti.

Algoritmo di Verifica

La verifica della firma F di un messaggio M è :

$$ver_K(M, \gamma, \delta) = \begin{cases} vero & \text{se } \beta^{\gamma}\gamma^{\beta} \equiv \alpha^{M} \bmod p \\ falso & \text{altrimenti.} \end{cases}$$

Se la firma è stata calcolata correttamente, allora la verifica avrà successo, poichè

$$\begin{aligned} \beta^{\gamma}\gamma^{\delta} &\equiv \alpha^{s\gamma}\alpha^{r\delta} \bmod p \\ &\equiv \alpha^{s\gamma+r\delta} \bmod p \\ &\equiv \alpha^{M} \bmod p \qquad \textit{poichè } \; s\gamma + r\delta = M \bmod (p-1). \end{aligned}$$

Esempio 3.1 *Supponiamo che* $p = 467$, $\alpha = 2$, $s = 127$; *allora* $\beta = \alpha^s = 132$. *Supponiamo che Alice voglia firmare il messaggio* $M = 100$ *e scelga il valore casuale* $r = 213$ *(notiamo che* $\gcd(213, 466) = 1$ *e* $213^{-1} \bmod 466 = 431$.*) Allora*

$$\gamma = 2^{213} \bmod 467 = 29$$

e

$$\delta = (100 - 127 \times 29)431 \bmod 466 = 51.$$

Chiunque può verificare questa firma controllando che

$$132^{29}29^{51} \equiv 189 \bmod 467$$

e

$$2^{100} \equiv 189 \bmod 467.$$

Quindi la firma è valida.

3.5 Possibili attacchi allo schema di firme ElGamal

Non esiste nessuna dimostrazione formale sulla sicurezza di questo schema, ovvero non è stata provata alcuna equivalenza tra la complessità di rompere questo schema e quella del calcolo del logaritmo discreto, o della soluzione del problema di Diffie-Hellman. In ogni caso, il miglior modo per rompere il sistema richiede il calcolo di alcuni logaritmi discreti, pertanto la sicurezza di questo schema si basa proprio sulla difficoltà di calcolare in maniera efficiente questi logaritmi. Nonostante ci siano stati diversi attacchi, esso non è mai stato rotto nè sono state trovate delle debolezze. In virtù di questa sicurezza negli Stati Uniti il NIST ha proposto come standard una variante di tale schema: lo schema di firme DSS.

Introduciamo ora alcuni dei possibili attacchi per uno schema di firme di ElGamal.

Nessuno degli attacchi che mostreremo sembra rompere lo schema di firme.

Gli attacchi possono essere divisi in due gruppi: il primo gruppo include alcuni attacchi per scoprire i valori della chiave segreta, il secondo include attacchi per falsificare la firma senza conoscere i valori segreti.

Attacchi del primo gruppo:

- Tentare di ricavare il valore segreto s dalla congruenza $\beta \equiv \alpha^s \bmod p$. Questo equivale a risolvere il problema del logaritmo discreto che è difficile da risolvere, per cui tale attacco è destinato a fallire;

- Scegliere un valore γ e poi tentare di calcolare il corrispondente δ dalla condizione di verifica $\beta^\gamma \gamma^\delta \equiv \alpha^M \bmod p$, cioè il nemico deve calcolare il logaritmo discreto $log_\gamma \alpha^M \beta^{-\gamma}$;

- Scegliere prima δ e poi tentare di ricavare γ dalla seguente equazione modulare: $\beta^\gamma \gamma^\delta \equiv \alpha^x \bmod p$. Questo è un problema per il quale nessun metodo efficiente per determinare la soluzione è noto; comunque tale problema non sembra essere collegato a nessun ben noto problema, come quello del logaritmo discreto o della fattorizzazione;

- Scegliere simultaneamente i valori γ e δ, in modo che la coppia (γ, δ) sia una firma per M. Nessuno ha finora scoperto un modo per fare questo, viceversa nessuno ha mai dimostrato che questo non possa essere fatto.

Attacchi del secondo gruppo:

Se, alternativamente, Oscar volesse calcolare la firma per un messaggio casuale M, allora potrebbe scegliere tra le seguenti modalità di falsificazione:

- Scegliere γ e δ, e poi cercare di risolvere per un messaggio casuale M. Allora ci si trova ancora di fronte al problema del logaritmo discreto, cioè il calcolo di $log_\alpha \beta^\gamma \gamma^\delta$. Quindi non si può firmare usando questo approccio;

- Firmare scegliendo simultaneamente γ, δ e M; in maniera tale che l'equazione di verifica risulti soddisfatta. Scegliendo opportunamente i valori γ, δ, M è possibile falsificare la firma solo per determinati tipi di messaggi e questo avviene indipendentemente dalla chiave segreta s. In questo modo si riescono a falsificare determinati messaggi che vedremo adesso.

  Il nemico sceglie due interi i e j tali che $0 \leq i \leq p-2$, $0 \leq j \leq p-2$, e $\gcd(j, p-1) = 1$. Quindi esegue le seguenti operazioni:

$$\begin{aligned}
&\gamma = \alpha^i \beta^j \bmod p \\
&\delta = -\gamma j^{-1} \bmod (p-1) \\
&M = -\gamma i j^{-1} \bmod (p-1),
\end{aligned}$$

  dove j^{-1} è calcolato modulo $(p-1)$ (ecco perchè richiediamo che j sia relativamente primo a $(p-1)$). Si può dimostrare che la coppia (γ, δ) è una firma valida per M:

$$\begin{aligned}\beta^\gamma\gamma^\delta &\equiv \beta^\gamma(\alpha^i\beta^j)^{-\gamma j^{-1}} \bmod p\\ &\equiv \beta^\gamma\alpha^{-ij^{-1}\gamma}\beta^{-j\gamma j^{-1}} \bmod p\\ &\equiv \alpha^{-ij^{-1}\gamma} \bmod p\\ &\equiv \alpha^{-\gamma ij^{-1}} \bmod p\\ &\equiv \alpha^M \bmod p.\end{aligned}$$

Esempio 3.2 *Supponiamo* $p = 467$, $\alpha = 2$ *e* $\beta = 132$. *Supponiamo che Oscar scelga* $i = 99$ *e* $j = 179$*; allora* $j^{-1} \bmod 466 = 151$. *Dovrebbe effettuare i seguenti calcoli:*

$$\begin{aligned}\gamma &= 2^{99}132^{179} \bmod 467 &&= 117\\ \delta &= -117 \times 151 \bmod 466 &&= 41\\ M &= 99 \times 41 \bmod 466 &&= 331\end{aligned}$$

Allora $(117, 41)$ *è una firma valida per il messaggio* 331*, verificando che*

$$132^{117}117^{41} \equiv 303 \bmod 467$$

e

$$2^{231} \equiv 303 \bmod 467.$$

Quindi la firma è valida.

Notiamo che questo tipo di attacco ha successo solo se lo spazio dei messaggi di senso compiuto è denso rispetto a $\mathcal{P}$. Per ovviare a questo tipo di attacco si può rendere tale spazio sparso rispetto a $\mathcal{P}$, oppure utilizzare una funzione hash come descritto nella sezione 2.5 nello schema di firme RSA.

- Scegliere λ, μ ed x' nell'ipotesi che Oscar conosca una firma (γ, δ) per un messaggio M. Per esempio M potrebbe essere un messaggio precedentemente firmato da Alice. Allora è possibile per Oscar firmare altri messaggi (casuali). Supponiamo che h, i e j siano interi tali che $0 \leq h, i, j \leq p-2$ e $\gcd(h\gamma - j\delta, p-1) = 1$. Egli calcola messaggio e firma nel seguente modo :

$$\begin{aligned}\lambda &= \gamma^h\alpha^i\beta^j \bmod p\\ \mu &= \delta\lambda(h\gamma - j\delta)^{-1} \bmod (p-1)\\ M' &= \lambda(hM + i\delta)(h\gamma - j\delta)^{-1} \bmod (p-1),\end{aligned}$$

  dove $(h\gamma - j\delta)^{-1}$ è calcolato modulo $(p-1)$. È immediato controllare che è soddisfatta la condizione di verifica:

$$\beta^\lambda\lambda^\mu \equiv \alpha^{M'} \bmod p.$$

  Quindi (λ, μ) è una valida firma per M'.

Infine, menzioniamo due modi attraverso i quali lo schema di firme ElGamal può essere rotto se non usato in maniera accurata:

- Il valore segreto r, utilizzato per la computazione della firma, non deve essere mai rivelato. Infatti, se r è noto, è semplice calcolare il valore segreto s come :

$$s = (M - r\delta)\gamma^{-1} \bmod (p-1);$$

e quindi Oscar può falsificare firme a piacimento;

- Un altro cattivo uso del sistema è di usare lo stesso valore di r per firmare due differenti messaggi. Questo facilita Oscar nel calcolo del valore s e quindi nel rompere il sistema. Ciò può essere fatto come segue: supponiamo che (γ, δ_1) sia una firma per un messaggio M_1 e che (γ, δ_2) sia una firma per M_2; quindi abbiamo

$$\beta^{\gamma}\gamma^{\delta_1} \equiv \alpha^{M_1} \bmod p$$

e

$$\beta^{\gamma}\gamma^{\delta_2} \equiv \alpha^{M_2} \bmod p.$$

Quindi dividendo la seconda per la prima si ottiene:

$$\alpha^{M_2 - M_1} \equiv \alpha^{r(\delta_2 - \delta_1)} \bmod p$$

la quale, dal teorema del logaritmo discreto, è equivalente a

$$M_2 - M_1 \equiv r(\delta_2 - \delta_1) \bmod (p-1). \tag{1}$$

Sia $d = \gcd(\delta_2 - \delta_1, p-1)$. Poichè $d|(p-1)$ e $d|(\delta_2 - \delta_1)$, segue che $d|(M_2 - M_1)$. Definiamo:

$$M' = \frac{M_2 - M_1}{d}$$
$$\delta' = \frac{\delta_2 - \delta_1}{d}$$
$$p' = \frac{p-1}{d}.$$

Allora la congruenza diviene :

$$M' \equiv r\delta' \bmod p'.$$

Poichè $\gcd(\delta', p') = 1$, possiamo calcolare l'inverso di δ', $\epsilon = (\delta')^{-1} \bmod p'$. Allora il valore di r modulo p' è dato da:

$$r \equiv M'(\delta')^{-1} \bmod p' \Leftrightarrow r \equiv M'\epsilon \bmod p'.$$

3 Dall'equazione modulare (1), si deduce che r può avere d valori distinti:

$$r \equiv (M'\epsilon + ip') \bmod p$$

per $0 \leq i \leq d-1$. Di questi d valori di r, quello corretto può essere determinato testando la condizione

$$\gamma \equiv \alpha^{r} \bmod p. \tag{2}$$

Osserviamo che se il valore di d è grande, i possibili valori di r in $\mathbf{Z}^*_{p-1}$ che sono congrui ad $M'\epsilon \bmod p'$ sono molti, il che implica che abbiamo poca informazione sul valore esatto di r. Viceversa, se d è piccolo abbiamo molta informazione sul valore di r, che possiamo verificare effettuando un test sulla congruenza (2). Tutto questo dipende dal valore di d, per cui se Oscar ha fortuna ottenendo un valore di d piccolo, allora può rompere il sistema calcolando il valore s. In conclusione la scelta casuale di r è condizione importante per garantire la sicurezza di questo schema di firme.

Un inconveniente dello schema di firme di ElGamal, oltre a quello di avere una firma la cui lunghezza è doppia di quella del messaggio, è dovuto al fatto che le operazioni effettuate, anche se in maniera veloce, sono comunque più lente di quelle dello schema di firme RSA. Va notato comunque che alcune operazioni non devono necessariamente essere computate all'atto della firma: il calcolo di γ è indipendente dal messaggio e potrebbe essere effettuato prima della firma del messaggio, in altre parole possiamo precomputare dei valori opportuni di γ, memorizzandoli da qualche parte ed usarli quando si deve firmare. Anche per il calcolo di δ possiamo precomputare r^{-1} e $s\gamma$. In questo modo all' atto della firma ci limitiamo solo a fare delle operazioni di addizione e moltiplicazione. Possiamo quindi separare la precomputazione dalla computazione on-line per firma. Con questa tecnica lo schema di firme di ElGamal può essere enormemente velocizzato.

4 Digital Signature Standard (DSS)

Nell'agosto del 1991, il National Institute of Standard and Technology (NIST) propose il Digital Signature Algorithm (DSA), per essere usato nel nuovo Digital Signature Standard (DSS). Questo standard fu rivisto nel 1993 [6], in risposta alle numerose critiche mossegli contro.

Questo standard specifica un algoritmo di firme digitali a chiave pubblica (DSA) usato per le applicazioni di firme digitali.

Prima di fare confusione, diamo uno sguardo alla nomenclatura: il DSA è l'algoritmo; il DSS è lo standard. Lo standard esegue l'algoritmo e l'algoritmo è una parte dello standard.

In questa sezione descriveremo il DSS per la generazione e la verifica di firme digitali, e l'algoritmo per la generazione di chiavi pubbliche e private. In entrambi i processi di generazione e firma il messaggio M è compresso dalla funzione hash (SHA) che fornisce come output un messaggio di 160 bit.

4.1 Calcolo di un generatore di Z_p^*

Vediamo ora come calcolare un generatore di Z_p^*. Una prima idea potrebbe essere quella di scegliere un elemento $g \in Z_p^*$ e vedere, calcolando le potenze $g^1, g^2, g^3, ...$, se genera tutti i possibili valori di Z_p^*. La complessità di tale algoritmo sarà proporzionale alla cardinalità di Z_p^* e lavorerá in tempo esponenziale rispetto alla lunghezza di p. Se p è di 512 bit allora la cardinalità di Z_p^* è dell' ordine di 2^{512}.

L'unico algoritmo conosciuto, in grado di determinare efficientemente un generatore di un gruppo ciclico Z_p^* presuppone la conoscenza della fattorizzazione di $p-1$, in quanto si basa sul seguente teorema:

Teorema 4.1.1. Sia dato un gruppo ciclico G con $|G|=p-1$ e sia data la fattorizzazione di $p-1=p_1^{e_1} p_2^{e_2} ... p_l^{e_l}$. Allora:

$$\alpha \text{ è un generatore di } G \text{ se e solo se } \begin{cases} \alpha^{\frac{p-1}{p_1}} \neq 1 \bmod p \\ \vdots \\ \alpha^{\frac{p-1}{p_l}} \neq 1 \bmod p \end{cases}$$

Dimostrazione: Supponendo che α sia un generatore di G, se esistesse un $i \in \{1,..,n\}$ tale che $\alpha^{\frac{p-1}{p_i}} = 1 \bmod p$, l'ordine del sottogruppo generato da α è al più uguale a $p_1^{e_1}...p_{i-1}^{e_{i-1}}p_i^{e_i-1}p_{i+1}^{e_{i+1}}...p_l^{e_l} < p-1$. Assurdo.
Inversamente, per il teorema di Lagrange l'ordine di α divide $|G|$. Ma $|G|= p-1$ e gli unici elementi che dividono $p-1$ sono per ipotesi $p_1, ..., p_l$. Se, quindi, valgono le equazioni dell'enunciato deve risultare necessariamente che l'ordine di α è uguale a $|G|$.

□

Mostriamo l'algoritmo per la generazione di α che si basa sul precedente teorema:

Algoritmo per trovare un generatore di un gruppo ciclico

INPUT: Un gruppo ciclico G di ordine $p-1$, con p primo;
fattorizzazione di $p-1 = p_1^{e_1}p_2^{e_2}...p_l^{e_l}$

1. Scegli un elemento a caso $\alpha \in G$.
2. **for** $i = 1$ **to** l **do**
   calcola $b = \alpha^{\frac{p-1}{p_i}}$
   if $b = 1$ **then goto** passo 1
3. OUTPUT: α come generatore del gruppo ciclico G.

Notiamo che il test al passo 2 dell'algoritmo si basa sul Teorema 4.1.1.
Dalla proprietà 3 della sezione 3.1 segue che il numero di generatori modulo un numero primo p è $\phi(\phi(p)) = \phi(p-1)$. Inoltre, dalla teoria dei numeri risulta che $\phi(n) > n/(6\ln\ln n)$ per tutti gli interi $n \geq 5$. Segue quindi che $\phi(\phi(p)) = \phi(p-1) > (p-1)/(6\ln\ln(p-1))$.
Pertanto, la probabilità che un elemento scelto a caso in Z_p^* sia un generatore risulta essere $\geq (1/(6\ln\ln(p-1)))$.

4.2 I parametri del DSS

La chiave è $K = \{p, q, \alpha, \beta, s\}$ dove: p è un numero primo la cui lunghezza è compresa tra 512 e 1024 bit, multiplo di 64; q è un numero primo di 160 bit che divide $p-1$; $\alpha \in Z_p^*$ è una radice q-esima di 1 mod p; $s < q$ è un numero casuale segreto di 160 bit; $\beta = \alpha^s \bmod p$ è un numero di 160 bit. La chiave pubblica è (p, q, α, β) mentre quella privata è formata dagli stessi valori più l'aggiunta del valore s. Inoltre $\mathcal{P} = Z_p^*$, $\mathcal{A} = Z_q \times Z_q$.

4.3 Generazione della firma

La firma del messaggio M è una coppia di numeri (γ, δ) calcolati secondo i seguenti passi.

> **Algoritmo di firma DSS**
>
> La firma F di un messaggio M è :
>
> 1. calcola un numero casuale r tale che $1 \leq r \leq q-1$
> 2. $\gamma = (\alpha^r \bmod p) \bmod q$
> 3. $\delta = (SHA(M) + s\gamma) r^{-1} \bmod q$
>
> $$sig_K(M, r) = (\gamma, \delta).$$

Nel passo 3, l'inverso di r esisterà certamente poichè $1 \leq r \leq q-1$, e $gcd(r, q) = 1$ in quanto q è primo. Il valore di SHA(M) è di 160 bit.

4.4 Procedura di verifica

Per verificare la firma F del messaggio M, Bob deve conoscere la chiave pubblica (p, q, α) di Alice.

> **Algoritmo di verifica DSS**
>
> La verifica della firma F di un messaggio M è :
>
> 1. $e_1 = M\delta^{-1} \bmod q$
> 2. $e_2 = \gamma\delta^{-1} \bmod q$
>
> $$ver_K(M, \gamma, \delta) = \begin{cases} vero & \text{se } (\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q = \gamma \\ falso & \text{altrimenti.} \end{cases}$$

Verifichiamo adesso la correttezza della procedura di verifica:

$$\begin{aligned} (\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q &= (\alpha^{M\delta^{-1}}\alpha^{s\gamma\delta^{-1}} \bmod p) \bmod q \\ &= (\alpha^{M\delta^{-1}+s\gamma\delta^{-1}} \bmod p) \bmod q \\ &= (\alpha^r \bmod p) \bmod q \quad \textit{poichè} \quad \delta^{-1}(SHA(M) + s\gamma) = r \bmod q \\ &= \gamma. \end{aligned}$$

4.5 Differenze tra DSS ed ElGamal

Il DSS modifica lo schema di ElGamal in un modo ingegnoso, infatti nello schema di ElGamal la firma consisteva di una coppia (γ, δ) dove sia δ che γ erano dell' ordine di 512 bit, producendo quindi una firma di 1024 bit su messaggi di qualsiasi dimensione, mentre il DSS produce sempre una firma di 320 bit perchè le operazioni sono modulo q, dove q è dell'ordine di 160 bit.

I cambiamenti operati nello schema di ElGamal sono:

- Il primo cambiamento che operiamo è modificare il segno "$-$" in "$+$" nella definizione di δ, così
$$\delta = (M + s\gamma)r^{-1} \bmod (p-1).$$
Questo muta la condizione di verifica in questo modo:
$$\alpha^M \beta^\gamma \equiv \gamma^\delta \bmod p. \tag{3}$$
L'esistenza dell'inverso di $\delta \bmod (p-1)$ è garantita dalle seguenti equivalenze:
$$\delta y \equiv 1 \bmod (p-1)$$
$$\Leftrightarrow$$
$$(M + s\gamma)r^{-1}y \equiv 1 \bmod (p-1)$$
$$\Leftrightarrow$$
$$(M + s\gamma)y \equiv r \bmod (p-1);$$
l'ultima equazione ammette una unica soluzione se, e solo se, $d = \gcd(M+s\gamma, p-1) = 1$. Moltiplicando gli esponenti della (3) per δ^{-1} si ha:
$$\alpha^{M\delta^{-1}} \beta^{\gamma\delta^{-1}} \equiv \gamma \bmod p. \tag{4}$$

- La seconda modifica riguarda la scelta di:
  - Un primo q tale che $q|(p-1)$
  - Una radice α q-esima di 1 modulo p.

  Queste condizioni assicurano che β e γ sono radici q-esime dell' unità , da cui segue che un qualunque esponente di α, β, γ può essere ridotto modulo q senza influenzare la precedente condizione di verifica. Quindi quello che si fa è di computare gli esponenti ed il lato destro della (4) modulo q. Si noti che γ appare come esponente sul lato sinistro della (4), ma non come esponente sul lato destro. Quindi se γ è ridotto modulo q, allora dobbiamo ridurre l'intero lato sinistro della (4) allo scopo di eseguire la verifica. Queste computazioni possono essere effettuate, si è così giunti alla seguente condizione di verifica del DSS:
$$(\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q \equiv \gamma.$$
  In questo modo abbiamo ridotto la grandezza della firma da 1024 bit a 320 bit grazie alle computazioni modulo q.

4.6 Critiche allo schema DSS

Quando il DSS è stato proposto dal NIST come standard, ci furono diverse critiche. Sfortunatamente, queste critiche furono più politiche che accademiche. RSA Data Security Inc., autrice dell'algoritmo RSA, condusse la maggior parte delle critiche contro il DSS. Essa voleva RSA, e non un altro algoritmo, usato come standard, questo perchè le licenze d'uso dell'algoritmo RSA avrebbero fornito grossi proventi alla RSADSI, mentre uno standard di firme digitali libero da licenze avrebbe comportato una diminuzione del suo fatturato. Lo standard proposto fu sviluppato dalla National Security Agency (NSA) senza l'ausilio dell'industria statunitense, e molti criticarono questo approccio a "porte chiuse" senza tener conto dei meriti del sistema proposto. Diverse critiche tecniche, invece, furono fatte sul sistema :

- DSS non può essere usato per cifrare o per distribuire chiavi.

  Questo è vero, ma ciò non è il punto dello standard, poichè esso è stato progettato esclusivamente per le firme digitali.

- DSS è stato sviluppato dall'NSA e potrebbe esserci una trap-door nel l'algoritmo.

- DSS è più lento di RSA.

  Nel DSS i tempi di firma sono più veloci dei tempi di verifica. Per contro, se RSA è usato come schema di firme e l'esponente pubblico b è molto piccolo (per esempio 3), allora la verifica può essere eseguita molto più velocemente della firma. In definitiva la maggiore critica che viene fatta nei confronti del DSS riguarda la verifica. Infatti, mentre l'operazione di firma viene realizzata una sola volta, la verifica, sulla stessa firma, può essere effettuata più volte nel tempo. Questo suggerisce che uno schema di firme con algoritmo di verifica più veloce sia desiderabile. La risposta del NIST fu che i tempi di verifica del DSS erano tali da poter essere sostenuti.

  Le implementazioni reali del DSS possono essere velocizzate attraverso precomputazioni. Notiamo che il parametro γ è indipendente dal messaggio; si può quindi creare un insieme di possibili valori del parametro r e per ognumo di loro precomputare il parametro γ e il valore r^{-1}, e porli in un database. Quando un messaggio deve essere firmato è sufficiente prelevare un valore di r e il suo corrispondente γ, e poi calcolare il parametro δ. Questa precomputazione velocizza considerevolmente il DSS. Riportiamo, qui di seguito, un confronto computazionale tra DSS e RSA per una implementazione su smart-card [3].

	DSS	*RSA*	*DSS con p, q, g comuni*
Precomputazioni	14 sec.	—	4 sec.
Firma	0.3 sec.	15 sec.	0.3 sec.
Verifica	16 sec. 1-5 sec. Off Cards	1.5 sec.	10 sec. 1-3 sec. Off Cards

  Le computazioni Off Cards sono state eseguite su un 80386 a 33MHz.

- La misura del modulo p è troppo piccola.

  La taglia del modulo p fu fissata a 512 bit nel 1991 e molte persone ritenevano che tale modulo dovesse essere di taglia non fissata, allo scopo di garantire una maggiore sicurezza. In risposta a tale critica, il NIST nel 1995 suggerì che una variazione della lunghezza del modulo p era permessa, scegliendo tale misura in modo tale da essere divisibile per 64, e compresa tra 512 e 1024 bit.

- La scelta dei parametri p e q deve essere sicura.

  Un'ulteriore critica riguardava, infatti, la scelta dei valori p e q. Tale critica nacque dal fatto che esistono dei valori p e q che permettono la falsificazione.

  Il NIST ha risolto questo problema fornendo un algoritmo per la determinazione dei parametri p e q.

4.7 Generazione delle chiavi

Mostriamo ora un algoritmo per la generazione di chiavi:

> **Algoritmo di generazione delle chiavi**
>
> 1. Seleziona un numero primo q tale che $2^{159} < q < 2^{160}$
> 2. Scegli t tale che $0 \leq t \leq 8$ e seleziona un numero primo p tale che $2^{511+64t} < q < 2^{512+64t}$ con la proprietà che $q|\,(p-1)$
> 3. Selezione di un generatore α dell'unico gruppo ciclico di ordine q in Z_p^*
>   3.1 Scegli a caso un elemento $g \in Z_p^*$ e calcola $\alpha = g^{\frac{p-1}{q}}$ mod p
>   3.2 **if** $\alpha = 1$ **then goto** passo 3.1
> 4. Seleziona un intero a caso s tale che $1 \leq s \leq q-1$
> 5. Calcola $\beta = \alpha^s \bmod p$

La lunghezza di q è di 160 bit, la lunghezza di p va da 512 a 1024 bit utilizzando multipli di 64 bit (512, 576,...).
Il passo 3 trova un elemento α che è di ordine q, infatti $\alpha^q = (g^{\frac{p-1}{q}})^q \equiv 1 \mod p$ e, quindi, per definizione α è un elemento di ordine q. Infine, l'algoritmo calcola i valori β e s della chiave.
I passi 1 e 2 richiedono la selezione di numeri primi; vediamo, quindi, di seguito un algoritmo che ci permette di generare tali primi.

4.7.1 Algoritmo per la generazione dei primi

Prima di descrivere il funzionamento del seguente algoritmo, osserviamo che un approccio alternativo potrebbe essere quello di scegliere a caso il valore di p, e poi determinare un fattore q di $p-1$ che sia di 160 bit. Naturalmente questo tipo di soluzione risulta essere computazionalmente onerosa, in quanto involve la fattorizzazione di $p-1$ per calcolare q. Nell'algoritmo proposto dal NIST viene scelto dapprima il valore q (160 bit) in maniera casuale, e successivamente si determina p; vediamo quali sono i passi attraverso i quali tale algoritmo lavora:

Choose(L)
Sia L la lunghezza di p
(1) Siano n e b interi tali che $L - 1 = n \times 160 + b$
(2) **repeat**
(3) **repeat**
(4) Scegli una sequenza casuale, S, di almeno 160 bit
(5) $g = |S|$
(6) $U = \mathbf{SHA}(S) \oplus \mathbf{SHA}((S+1) \bmod 2^g)$
(7) Forma q da U ponendo il MSB ed il LSB ad 1
(8) **until** q primo
(9) $C = 0$
(10) $N = 2$
(11) **repeat**
(12) **for** $k = 0$ **to** n **do** $V_k = \mathbf{SHA}((S + N + k) \bmod 2^g)$
(13) $W = V_0 + V_1 \times 2^{160} + \ldots + V_{n-1} \times 2^{(n-1)\times 160} + (V_n \bmod 2^b) \times 2^{n \times 160}$
(14) $X = W + 2^{L-1}$
(15) $p = X - ((X \bmod 2q) - 1)$
(16) **until** (p primo) **or** ($p < 2^{L-1}$)
(17) **if** $p < 2^{L-1}$
(18) $C = C + 1$
(19) $N = N + n + 1$
(20) **If** $C < 4096$ **then goto step** (12)
(21) **else** Help=falso
(22) **else** Help=vero
(23) **until** Help
(24) **return** p, q, S, C

Illustriamo i passi di tale algoritmo. Il ciclo (3)-(8) calcola il valore di q utilizzando la funzione one-way SHA che previene la possibilità di invertire i passi della computazione di q. Nel ciclo (11)-(16) viene calcolato un valore per p. Se p è primo e la sua lunghezza è esattamente L, il ciclo (2)-(23) termina con la determinazione dei primi p e q. Altrimenti se p non è primo o la sua lunghezza non è esattamente L, allora il ciclo (11)-(16) viene ripetuto fino a quando un primo p viene trovato oppure il numero di iterazioni è minore di 4096. Se, dopo tale numero di iterazioni, non è stato ancora determinato un primo p di lunghezza L, allora viene ripetuto il ciclo (2)-(23) ricalcolando, quindi, il primo q. I parametri S e C rappresentano rispettivamente la sequenza casuale di bit per generare q, e il numero di iterazioni effettuate per generare p. Come tali sono dei testimoni della validità di p e q.
Questo algoritmo potrebbe comunque calcolare dei valori truffa di p e q, ma questo non è un problema per due motivi: innanzitutto i moduli per i quali tale proprietà è vera sono facili da scoprire; inoltre questi moduli sono così rari che la probabilità di calcolarne uno è più piccola della probabilità di generare un numero composto utilizzando una routine probabilistica di generazione di numeri primi.

5 Schema di firme di Lamport

La sicurezza degli schemi di firma, visti fin'ora, non è stata mai quantizzata, cioè nessuno ha mai dimostrato che, per esempio, falsificare firme in DSS è tanto difficile quanto risolvere un certo problema (come quello del logaritmo discreto o della fattorizzazione). È possibile, allora, costruire schemi di firme digitali per i quali un attacco presenta la medesima difficoltà di un problema difficile? Ciò può essere fatto, ma nel contempo, tale requisito rende un siffatto schema valido solo da un punto di vista teorico, poichè l'elevata sicurezza va a discapito della praticità. Illustreremo adesso degli schemi di firme detti *ONE-TIME*, i quali utilizzano funzioni one-way. Il termine one-time significa che solo un messaggio può essere firmato (naturalmente la firma può, anche in questo caso, essere verificata un numero arbitrario di volte). Il primo schema che analizzeremo in questa sezione è quello di Lamport, mentre nella successiva sezione ci occuperemo dello schema di firme di Bos-Chaum.

5.1 Parametri di Lamport

La chiave è $K = \{W, W'\}$ dove W, W' sono due matrici di dimensione $t \times 2$. Le matrici W', W sono rispettivamente la chiave pubblica e quella privata.

5.2 Generazione dei parametri

Sia $M \in \{0,1\}^t$ un messaggio da firmare e sia $f : Y \to Z$ una funzione one-way. Allora Alice per firmare M esegue i seguenti passi:

- Costruisce una matrice W di dimensioni $t \times 2$ i cui elementi sono dei valori scelti a caso in Y.
- Costruisce una seconda matrice W' tale che $W'_{i,j} = f(W_{i,j})$ che rende pubblica.

5.3 Generazione della firma

Algoritmo di firma

La firma F di un messaggio $M = (m_1, m_2, \cdots, m_t)$ dove $m_i \in \{0,1\}$ è :

$$sig_K(M) = (W_{1,m_1}, \cdots, W_{t,m_t})$$

5.4 Procedura di verifica

Algoritmo di verifica

La verifica della firma $F = (F_1, \cdots, F_t)$ dove $F_i \in Z$ di un messaggio M è :

$$ver_K(M, F) = \begin{cases} vero & \text{se } f(F_1) = W'_{1,m_1}, \cdots, f(F_t) = W'_{t,m_t} \\ falso & \text{altrimenti.} \end{cases}$$

Lo schema lavora in questo modo. Supponiamo di avere un problema "difficile" cioè abbiamo in pratica una funzione f one-way, per esempio la fattorizzazione. Alice esegue i seguenti passi:

1. Si calcola diversi numeri primi $p_1, q_1, \cdots, p_t, q_t$
2. Si costruisce la matrice W mettendo nella prima colonna la coppia (p_i, q_i) con i dispari e nella restante colonna la coppia (p_i, q_i) con i pari.
3. Si calcola gli elementi della matrice W' come i prodotti $n_i = p_i q_i$ della precedente matrice e la rende pubblica.
4. Per firmare il messaggio M Alice userà i fattori dei numeri che si trovano sulla prima colonna della matrice W per firmare lo 0 e quelli della seconda colonna per firmare l'1.
5. Bob per verificare la firma effettua i prodotti a coppia e verifica che siano uguali ai valori della matrice pubblica W'.

Esempio 5.1 *Siano* $p_1 = 3, q_1 = 23, p_2 = 5, q_2 = 31, p_3 = 7, q_3 = 53, p_4 = 13, q_4 = 47, p_5 = 59, q_5 = 41, p_6 = 17, q_6 = 29, p_7 = 37, q_7 = 19, p_8 = 11, q_8 = 13$ *i primi scelti da Alice.*

Alice calcola adesso le matrici W *ed* W' *di dimensione* 4×2

$$W = \begin{bmatrix} (3,23) & (5,31) \\ (7,53) & (13,47) \\ (59,41) & (17,29) \\ (37,19) & (11,13) \end{bmatrix} \qquad W' = \begin{bmatrix} 69 & 155 \\ 371 & 611 \\ 2419 & 493 \\ 703 & 143 \end{bmatrix}$$

La firma del messaggio $M = 0011$ *corrisponde alla sequenza:* $(p_1, q_1), (p_3, q_3), (p_6, q_6), (p_8, q_8) = (3,23), (7,53), (17,29), (11,13)$.

Bob per verificare la firma controlla che i prodotti delle coppie ricevute $(3,23), (7,53), (17,29), (11,13)$ *siano uguali ai valori* $(69, 371, 493, 143)$ *della matrice pubblica.*

Esempio 5.2 *Sia* $p = 7879$ *un primo e* $\alpha = 3$ *un elemento primitivo in* $\mathbf{Z}_{7879}$*. Definiamo:*

$$f(x) = 3^x \bmod 7879.$$

Supponiamo che Alice desideri firmare un messaggio M=(1,1,0). Allora sceglie sei numeri casuali segreti: $w_1 = 5831, w_2 = 735, w_3 = 803, w_4 = 2467, w_5 = 4285, w_6 = 6449$ *Quindi calcola le immagini delle* w *attraverso la funzione* f*, cioè* $w_1' = 2009, w_2' = 3810, w_3' = 4672, w_4' = 4721, w_5' = 268, w_6' = 5731$.

Si costruisce quindi le due matrici W *e* W' *con i valori precedentemente calcolati.*

$$W = \begin{bmatrix} 5831 & 735 \\ 803 & 2467 \\ 4285 & 6449 \end{bmatrix} \qquad W' = \begin{bmatrix} 2009 & 3810 \\ 4672 & 4721 \\ 268 & 5731 \end{bmatrix}$$

La matrice W' *è pubblica. La firma per* M *è* $(w_2, w_4, w_5) = (735, 2467, 4285)$*. Per verificare questa firma, è sufficiente effettuare i seguenti calcoli:*

$$3^{735} \bmod 7879 = 3810$$
$$3^{2467} \bmod 7879 = 4721$$
$$3^{4285} \bmod 7879 = 268$$

e verificare che siano uguali ai corrispondenti elementi di W'.

Oscar non può falsificare una firma dal momento che è difficile invertire la funzione one-way f. Comunque, lo schema di firme può essere utilizzato per firmare solo un messaggio, poichè disponendo delle firme di due messaggi differenti, è facile per Oscar poter "costruire" delle firme per altri messaggi. Per esempio supponiamo che i messaggi $(0,1,1)$ e $(1,0,1)$ siano entrambi firmati usando questo schema. Il messaggio $(0,1,1)$ ha come firma la tripla (w_1, w_4, w_6), e il messaggio $(1,0,1)$ è firmato con (w_2, w_3, w_6). Quindi Oscar può costruire firme per i messaggi $(1,1,1)$, cioè (w_2, w_4, w_6), e $(0,0,1)$, cioè (w_1, w_3, w_6). Anche se questo schema è molto elegante, non è di grande uso pratico data la grandezza delle firme che esso produce. Per esempio, se noi usiamo la funzione di esponenziazione modulare, come nell'esempio precedente, allora una implementazione sicura dovrebbe richiedere che p sia di almeno 512 bit. Questo significa che ogni bit del messaggio viene firmato usando 512 bit, e dunque la firma risulta essere 512 volte la lunghezza del messaggio originario.

6 Schema di firme di Bos-Chaum

Una modifica dello schema precedente è dovuta a Bos e Chaum[4], i quali hanno fornito uno schema che permette di avere, rispetto a quello di Lamport, delle firme più corte. Riportiamo la descrizione di tale schema.

6.1 Generazione e verifica della firma

Indichiamo con t la lunghezza del messaggio da firmare.

Algoritmo di firma

Siano n e t tali da soddisfare le seguenti disuguaglianze:

$$n < t, \qquad 2^t \leq \binom{2n}{n}$$

Si scelgono, in $\mathbf{Z}_p$, dei valori casuali y, costruendo così l'insieme $Y = \{y_i : i = 1, \ldots, 2n\}$. Supponiamo che f sia una funzione one-way. Definiamo l'insieme

$$Z = \{z_i = f(y_i) : i = 1, \ldots, 2n\}.$$

Sia B la famiglia di tutti i sottoinsiemi di Y di cardinalità n, e sia $\psi(x)$ una funzione iniettiva pubblicamente nota, che associa ad ogni messaggio M (di lunghezza t) un elemento di B:

$$\psi : \{0,1\}^t \longrightarrow B.$$

Per $K = (y_i, z_i, 1 \leq i \leq 2n)$ definiamo :

$$sig_K(x_1, \ldots, x_t) = \psi(x_1, \ldots, x_t)$$

e

$$ver_K(x_1, \ldots, x_t, a_1, \ldots, a_n) = vero$$

$$\Leftrightarrow$$

$$\{f(a_i), 1 \leq i \leq 2n\} = \{z_j = f(y_j) : y_j \in \psi(x_1, \ldots, x_t)\}$$

Asintoticamente, fissato t, il minimo n tale che $2^t \leq \binom{2n}{n}$, è circa $t/2$, così otteniamo una riduzione del 50% nella lunghezza della firma usando lo schema di Bos-Chaum.

Come si evince da tale schema di firme, la funzione ψ gioca un ruolo fondamentale nella determinazione della firma. Infatti, tale funzione associa ad ogni messaggio M, di lunghezza t, un particolare elemento in B, che rappresenta la firma del messaggio dato. Mostreremo, quindi, una possibile funzione ψ, e successivamente descriveremo un algoritmo che, dato il messaggio M, calcola $\psi(x)$.

Ricordiamo che B è la famiglia di tutti i sottoinsiemi di Y di cardinalità n; su tali sottoinsiemi è possibile stabilire una relazione d'ordine nel seguente modo: $\{y_{r_1}, \ldots, y_{r_n}\} \in B$,

$r_1 < r_2 < \ldots < r_n$, precede $\{y_{s_1}, \ldots, y_{s_n}\} \in B$, $s_1 < s_2 < \ldots < s_n$, se e solo se esiste un indice h, $1 \leq h \leq n$, tale che $r_1 = s_1, \ldots, r_{h-1} = s_{h-1}$ e $r_h < s_h$. Una possibile funzione ψ è quella che associa all'i-esimo elemento della sequenza ordinata dei messaggi, l'i-esimo elemento di B rispetto alla relazione d'ordine. Tale corrispondenza è mostrata di seguito:

$$
\begin{array}{ccc}
\text{Messaggio} & & \text{Firma} \\
(0,0,0,0,0,0) & \longrightarrow & \{y_1, y_2, y_3, y_4\} \\
(0,0,0,0,0,1) & \longrightarrow & \{y_1, y_2, y_3, y_5\} \\
(0,0,0,0,1,0) & \longrightarrow & \{y_1, y_2, y_3, y_6\} \\
(0,0,0,0,1,1) & \longrightarrow & \{y_1, y_2, y_3, y_7\} \\
(0,0,0,1,0,0) & \longrightarrow & \{y_1, y_2, y_3, y_8\} \\
(0,0,0,1,0,1) & \longrightarrow & \{y_1, y_2, y_4, y_5\} \\
 & \ldots & \\
(1,1,1,1,0,0) & \longrightarrow & \{y_3, y_4, y_7, y_8\} \\
(1,1,1,1,0,1) & \longrightarrow & \{y_3, y_5, y_6, y_7\} \\
(1,1,1,1,1,0) & \longrightarrow & \{y_3, y_5, y_6, y_8\} \\
(1,1,1,1,1,1) & \longrightarrow & \{y_3, y_5, y_7, y_8\}
\end{array}
$$

Il seguente algoritmo effettua tale associazione.

Map(x, n, t)
(1) $X = \sum_{i=1}^{t} x_i 2^{t-i}$
(2) $\psi(x) = \emptyset$
(3) $q = 2n - 1$
(4) $e = n - 1$
(5) $i = 1$
(6) **while** $e \geq 0$ **do**
(7) **if** $X \geq \binom{q}{e}$ **then** $X = X - \binom{q}{e}$
(8) **else** $e = e - 1$
(9) $\psi(x) = \psi(x) \bigcup \{y_i\}$
(10) $q = q - 1$
(11) $i = i + 1$
(12) **return** $\psi(x)$

I passi (1)-(5) costituiscono la fase di inizializzazione, mentre il ciclo (6)-(11) fissa, ogni volta che la condizione $X \geq \binom{q}{e}$ non è verificata, un elemento della firma. Se all'i-esima iterazione $X < \binom{q}{e}$, allora il messaggio M ricade tra quei possibili $\binom{q}{e}$ messaggi per i

quali le corrispondenti firme contengono y_i e tutti gli y_j, con $j < i$, determinati nelle iterazioni precedenti.
Per meglio comprendere l'idea dell'algoritmo mostriamo un esempio:
Siano $x = (1,1,0,0,1,0)$, $t = 6$ ed $n = 4$. Nella fase di inizializzazione abbiamo che $X = 50$, $q = 2n-1 = 7$, $e = n-1 = 3$ ed $i = 1$, per cui $\psi(x)$ risulta essere il 50-esimo elemento in B secondo l'ordinamento lessicografico determinato dalla relazione d'ordine precedentemente stabilita. Alla prima iterazione del ciclo **while**, $X > \binom{q}{e} = \binom{7}{3} = 35$, per cui la firma $\psi(x)$, in virtù dell'ordinamento stabilito, non ricade tra le prime $\binom{7}{3}$ firme, le quali hanno in comune l'elemento $y_i = y_1$. Quindi y_1 non fa parte di $\psi(x)$, e la determinazione della firma prosegue effettuando la ricerca tra tutte quelle firme non contenenti y_1. Il valore di X viene quindi decrementato della quantità $\binom{7}{3}$, e il suo nuovo valore $X = 50 - 35 = 15$ rappresenta la posizione di $\psi(x)$ nel sottoinsieme ordinato di B i cui elementi non contengono y_1. I valori di q ed i diventano $q = 6$ ed $i = 2$, e alla seconda iterazione $X < \binom{6}{3} = 20$ e quindi la firma di M ricade tra le $\binom{6}{3}$ firme aventi in comune l'elemento y_2. Dunque y_2 fa parte della firma $\psi(x)$. Iterando si ottiene la firma $\psi(x) = \{y_2, y_4, y_6, y_8\}$. Infine osserviamo che essendo $|Y| = 2n$, tale algoritmo può essere implementato con una complessità di $O(n)$ operazioni aritmetiche.

Riferimenti

[1] T. ElGamal, *A public-key cryptosystem and a signature scheme based on discrete logarithms*, Advances in Cryptology-Crypto '84 Proceedings, Berlin: Springer-Verlag, 1985, pp. 10-18.

[2] T. ElGamal, *A public-key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transaction on Information Theory, vol. IT-31, n.4, 1985, pp. 469-472.

[3] M.E. Smid, *The DSS e the SHS*, Federal Digital Signature Application Symposium, Rockville, MD, 17-18 Feb 1993.

[4] J.N.E. Bos and D. Chaum. *Provably unforgeable signatures.* Lecture Notes in Computer Science 740 (1993) pp. 1-14. Crypto '92.

[5] R.A. Rivest, A. Shamir, e I. Adleman, *A method for obtaining digital signatures and public-key cryptosystems,* Communications of the ACM, vol.21, 1978, pp. 120-126.

[6] FIPS PUB 186, *Digital Signature Standard (DSS),* 19 maggio 1994.