Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Matematica

FIRMA DIGITALE E ALGORITMO DSA

Tesi di Laurea in

Algoritmi della Teoria dei Numeri e Crittografia

Relatore:
Prof.
Davide Aliffi

Presentata da:
Silvia Serra

Sessione II
Anno accademico 2013/2014

A chi crede in me e mi è sempre vicino ...

Introduzione

Per anni sono stati utilizzati diversi tipi di firma per associare l'identità delle persone ai documenti: per esempio, nel Medioevo i nobili usavano uno stampo di cera riportante il loro stemma. Al giorno d'oggi accade frequentemente di dover firmare un documento, un titolo di credito o di dover autenticare una fotocopia. In tutti questi casi è richiesta la propria firma per testimoniare la propria identità. L'identità viene quindi certificata attraverso la nostra grafia, che viene ritenuta un elemento identificativo della persona.

Con lo sviluppo del commercio elettronico, però, questi metodi non sono più sufficienti. Ad esempio, nel caso in cui si voglia firmare un documento elettronico, non è possibile digitalizzare solamente la propria firma e aggiungerla al documento in quanto chiunque abbia accesso al documento potrebbe semplicemente rimuoverla e aggiungerla ad un qualsiasi altro documento. La firma classica si potrebbe ritagliare dal documento originale (oppure fotocopiarla) e incollarla in altri testi, ma non sarebbe ritenuta valida a differenza di una falsificazione elettronica che è piuttosto semplice e non può essere distinta dall'originale. Per questo, è necessario avere delle firme digitali che non possano essere separate da un messaggio e incollate altrove; ciò significa che la firma non è legata solo al firmatario ma anche al messaggio da firmare. Inoltre la firma digitale deve essere facilmente verificabile da altri soggetti. Quindi si hanno due fasi: quella di generazione e quella di verifica.

La firma digitale è uno degli sviluppi più importanti della crittografia a chiave pubblica, che permette di implementarne le funzionalità di sicurezza. La crittografia a chiave pubblica, introdotta nel 1976 da Diffie ed Hellman,

è stata l'unica grande rivoluzione nella storia della crittografia. Essa si distacca in modo radicale da ciò che l'ha preceduta, sia perché i suoi algoritmi si basano su funzioni matematiche e non su operazioni di sostituzione e permutazione, ma sopratutto perché è asimmetrica: prevede l'uso di due chiavi distinte (mentre nelle crittografia simmetrica si usa una sola chiave condivisa tra le parti).

In particolare, le funzioni matematiche su cui si basa tale crittografia sono funzioni ben note nella Teoria dei Numeri: ad esempio fattorizzazione, calcolo del logaritmo discreto. La loro importanza deriva dal fatto che si ritiene che siano "computazionalmente difficili" da calcolare.

Dei vari schemi per la firma digitale basati sulla crittografia a chiave pubblica, si è scelto di studiare quello proposto dal NIST (National Institute of Standard and Technology): il Digital Signature Standard (DSS), spesso indicato come DSA (Digital Signature Algorithm) dal nome dell'algoritmo che utilizza.

Il presente lavoro è strutturato in tre capitoli. Nel Capitolo 1 viene introdotto il concetto di logaritmo discreto (centrale nell'algoritmo DSA) e vengono mostrati alcuni algoritmi per calcolarlo. Nel Capitolo 2, dopo una panoramica sulla crittografia a chiave pubblica, si dà una definizione di firma digitale e delle sue caratteristiche. Chiude il capitolo una spiegazione di un importante strumento utilizzato negli algoritmi di firma digitale: le funzioni hash. Nel Capitolo 3, infine, si analizza nel dettaglio il DSA nelle tre fasi che lo costituiscono (inizializzazione, generazione, verifica), mostrando come il suo funzionamento e la sua sicurezza derivino dai concetti precedentemente illustrati.

Indice

Capitolo 1

Logaritmo discreto

L'importanza dei logaritmi discreti deriva dal fatto che, essendo flessibili, possono essere implementati su un qualsiasi gruppo in cui sia difficile calcolarli. Verranno analizzate definizione e alcune proprietà, dopodiché verranno spiegati vari algoritmi per calcolarli.

1.1 Definizione

Il logaritmo discreto è uno dei problemi della teoria dei numeri con applicazioni crittografiche. Sono l'analogo dei logaritmi sui numeri reali ma operano nell'aritmentica modulare.

Definizione 1.1. Fissato un primo p, siano α e β due interi non nulli modulo p tali che

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

Il problema di trovare x è detto **problema del logaritmo discreto**.
Se n è il più piccolo intero positivo per cui $\alpha^n \equiv 1 \pmod{p}$ si può considerare $0 \leq x < n$ e scrivere

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

dove x è chiamato logaritmo discreto di β rispetto ad α.

Esempio 1.1. Siano $p = 11$ e $\alpha = 2$. Poiché $2^6 \equiv 9 \pmod{11}$ risulta $L_2(9) = 6$. Essendo $2^6 \equiv 2^{16} \equiv 2^{26} \equiv 9 \pmod{11}$, i valori 6, 16, 26 si possono considerare logaritmi discreti, ma si sceglie il più piccolo non negativo, quindi 6.

Spesso α è una radice primitiva modulo p, così ogni β è una potenza di $\alpha \pmod{p}$; se, al contrario, α non è una radice primitiva il logaritmo discreto non sarà definito per certi valori di β. Per alcuni aspetti il logaritmo discreto si comporta come il classico logaritmo: in particolare se α è una radice primitiva modulo p allora vale

$$L_\alpha(\beta_1\beta_2) \equiv L_\alpha(\beta_1) + L_\alpha(\beta_2) \pmod{p-1}$$

Se p è piccolo, è facile calcolare i logaritmi discreti tramite una ricerca esaustiva fra tutti i possibili esponenti; mentre se p è grande questo non è più possibile. È per questo motivo che, in generale, si pensa che i logaritmi discreti siano computazionalmente difficili da calcolare.
Per questo l'elevamento a potenza modulare è un esempio di funzione (probabilmente) unidirezionale, infatti è facile calcolare $\alpha^x \pmod{p}$, ma risolvere $\alpha^x \equiv \beta$ rispetto a x è intrattabile.

Funzione unidirezionale

Una **funzione unidirezionale** mappa un dominio in un codominio in modo che ogni valore della funzione abbia un inverso univoco, con la condizione che il calcolo della funzione sia facile mentre il calcolo dell'inverso sia intrattabile, cioè:

$$\begin{aligned} y &= f(x) && \text{facile} \\ x &= f^{-1}(y) && \text{intrattabile} \end{aligned}$$

In generale con *facile* si intende un problema che può essere risolto in un tempo polinomiale rispetto alla lunghezza dell'input. Quindi, se la lunghezza dell'input è di n bit, il tempo necessario per calcolare la funzione deve essere

proporzionale a n^a dove a è una costante fissa. Questi algoritmi appartengono alla **Classe** $\mathbb{P}$ cioè la **classe polinomiale**.
Mentre il termine *intrattabile* è più difficile da definire; si può dire che un problema è intrattabile se l'impegno per risolverlo cresce più velocemente di un polinomio in funzione delle dimensioni dell'input. Ad esempio, se la lunghezza dell'input è di n bit e il tempo necessario per calcolare la funzione è proporzionale a 2^n, il problema è considerato intrattabile.

1.2 Calcolo del logaritmo discreto

Esistono vari metodi per calcolare il logaritmo discreto, alcuni dei quali verrano analizzati nel dettaglio nei prossimi paragrafi.

1.2.1 Algoritmo elementare

Sia α una radice primitiva modulo p, in modo che $p-1$ sia il più piccolo esponente positivo n per cui $\alpha^n \equiv 1 \pmod{p}$. Allora

$$\alpha^{m_1} \equiv \alpha^{m_2} \pmod{p} \Leftrightarrow m_1 \equiv m_2 \pmod{p-1}$$

Si abbia

$$\beta \equiv \alpha^x \quad dove \quad 0 \leq x < p-1.$$

Il problema è determinare x. L'algoritmo permette di determinare $x \pmod 2$. Poiché

$$(\alpha^{(p-1)/2})^2 \equiv \alpha^{p-1} \equiv 1 \pmod{p}$$

si ha $\alpha^{(p-1)/2} \equiv \pm 1 \pmod{p}$.

Essendo $p-1$ il più piccolo esponente che dà $+1$, si deve avere

$$\alpha^{(p-1)/2} \equiv -1 \pmod{p}.$$

Elevando entrambi i membri della congruenza $\beta \equiv \alpha^x \pmod{p}$ alla potenza $(p-1)/2$ si ha

$$\beta^{(p-1)/2} \equiv \alpha^{x(p-1)/2} \equiv (-1)^x \pmod{p}.$$

Quindi se $\beta^{(p-1)/2} \equiv +1$ allora x è pari, altrimenti x è dispari.

Esempio 1.2. Si deve risolvere $2^x \equiv 9 \pmod{11}$. Dato che

$$\beta^{(p-1)/2} \equiv 9^5 \equiv 1 \pmod{11}$$

ne segue che x deve essere pari. Infatti, come si vede dall'esempio precedente, $x = 6$.

1.2.2 Algoritmo di Silver-Pohling-Hellman

Siano q e p due primi con p che divide $q-1$; l'obiettivo è trovare x, $0 \leq x < q-1$, tale che $b^x = y \pmod{q}$. Se

$$q - 1 = \prod_p p^\alpha$$

è la scomposizione in fattori primi di $q-1$, allora è sufficiente trovare $x \pmod{p^\alpha}$ per ogni p. Da ciò segue che x è univocamente determinato usando il teorema cinese del resto.

Innanzitutto, per ogni primo p, si calcolano le radici p-esime dell'unità $r_{p,j} = b^{j(q-1)/p}$ per $j = 0, 1, ..., p-1$.
Si fissi un primo p che divide $q-1$ e si supponga che, rispetto alla base p,

$$x \pmod{p^\alpha} = x_0 + x_1 p + x_2 p^2 + ... + x_{\alpha-1} p^{\alpha-1} \quad con \quad 0 \leq x_i < p.$$

Per trovare x_0 si calcola $y^{(q-1)/p}$, che è una radice p-esima di 1 poiché $y^{q-1} = 1$.
Dato che $y = b^x$, si ha

$$y^{(q-1)/p} = b^{x(q-1)/p} = b^{x_0(q-1)/p} = r_{p,x_0}.$$

Quindi si confronta $y^{(q-1)/p}$ con $(r_{p,j})_{0 \leq j < p}$ e si pone x_0 uguale al valore di j per cui $y^{(q-1)/p} = r_{p,j}$.
Successivamente, per trovare x_1 si sostituisce y con $y_1 = y/b^{x_0} = b^{x-x_0}$.
Allora y_1 ha logaritmo discreto

$$x - x_0 \equiv x_1 p + ... + x_{\alpha-1} p^{\alpha-1} \pmod{p^\alpha}$$

dato che

$$b^{x-x_0} = b^x/b^{x_0} = y/b^{x_0} = y_1,$$

quindi

$$y_1^{(q-1)/p^2} = b^{(x-x_0)(q-1)/p^2} = b^{(x_1+x_2p+...)(q-1)/p} = b^{x_1(q-1)/p} = r_{p,x_1}.$$

Quindi si può confrontare $y_1^{(q-1)/p^2}$ con $\{r_{p,j}\}$ e si pone x_1 uguale al valore di j per cui $y_1^{(q-1)/p^2} = r_{p,j}$.
Si procede in questo modo fino a $x_{\alpha-1}$; cioè per ogni $i = 1, 2, ..., \alpha-1$ si pone

$$y_i = y/b^{x_0+x_1p+...+x_{i-1}p^{i-1}}$$

che ha logaritmo discreto congruente $\pmod{\alpha^p}$ a $x_ip^i + ... + x_{\alpha-1}p^{\alpha-1}$.
Allora

$$y_i^{(q-1)/p^{i+1}} = b^{(x_i+x_{i+1}p+...)(q-1)/p} = b^{x_i(q-1)/p} = r_{p,x_i}.$$

Quindi si pone x_i uguale al valore di j per cui $y_i^{(q-1)/p^{i+1}} = r_{p,j}$.
Così si ha $x \pmod{\alpha^p}$. Dopo aver fatto ciò per ogni $p \mid q-1$ si usa il teorema cinese del resto per trovare x.

Poiché questo algoritmo richiede di conoscere la fattorizzazione di $q-1$, funziona bene quando tutti i primi che dividono $q-1$ sono piccoli mentre richiede molto tempo se i primi che dividono $q-1$ sono molto grandi.

Esempio 1.3. Siano $b = 2$ e $q = 37$. Si vuole trovare il logaritmo discreto di 28. Si ha

$$37 - 1 = 2^2 * 3^2$$

Si calcola $2^{18} \equiv 1 \pmod{37}$ e si ottiene $r_{2,0} = 1$, $r_{2,1} = -1$.
$2^{36/3} \equiv 26$ e $2^{2*36/3} \equiv 10 \pmod{37}$ da cui $\{r_{3,j}\} = \{1, 26, 10\}$.

Si considera $28 \equiv 2^x \pmod{37}$.
Si pone $p = 2$ e si trova $x \pmod 4$ che si scrive come $x_0 + 2x_1$.

Si calcola $28^{36/2} \equiv 1 \pmod{37}$ ottendendo $x_0 = 0$.
Inoltre calcolando $28^{36/4} \equiv -1 \pmod{37}$ si ha $x_1 = 1$ cioè $x \equiv 2 \pmod 4$.

Successivamente si pone $p = 3$ e si trova $x \pmod 9$ che si scrive come $x_0 + 3x_1$.

Per trovare x_0 si determina $28^{36/3} \equiv 26 \pmod{37}$ allora $x_0 = 1$.

Poiché $(28/2)^{36/9} = 14^4 \equiv 10 \pmod{37}$ si ha $x_1 = 2$.

Allora $x \equiv 1 + 2 * 3 = 7 \pmod 9$.

Mettendo insieme $x \equiv 2 \pmod 4$ e $x \equiv 7 \pmod 9$ si ottiene $x = 34$.

1.2.3 Index Calculus

Sia p un primo grande e α una radice primitiva, si vuole risolvere

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

Sia B un intero positivo e siano $p_1, p_2, ..., p_m$ i primi minori di B. Tale insieme è chiamato **base di fattorizzazione**.

Si calcola $\alpha^k \pmod p$ per certi valori di k e si prova a scrivere ognuno di questi numeri come prodotto di primi minori di B. Nel caso in cui non si riuscisse a fare ciò α^k viene scartato, in caso contrario, cioè $\alpha^k \equiv \prod p_i^{a_i} \pmod p$, risulta

$$k \equiv \sum a_i L_\alpha(p_i) \pmod{p-1}.$$

Quando si ottiene un numero sufficiente di queste relazioni, il corrispondente sistema può essere risolto in modo da avere $L_\alpha(p_i)$ per ogni i. Per interi r scelti a caso, si calcola $\beta\alpha^r \pmod p$ e si prova a scrivere ognuno di questi numeri come prodotto di primi minori di B. In caso di successo si ha $\beta\alpha^r \equiv \prod p_i^{b_i} \pmod p$, da cui segue

$$L_\alpha(\beta) \equiv -r + \sum b_i L_\alpha(p_i) \pmod{p-1}.$$

Questo algoritmo può essere utilizzato in pratica se p non è troppo grande, cioè significa che affinché il problema del logaritmo discreto sia intrattabile, p deve essere di almeno 200 cifre.

Esempio 1.4. Si considerino $p = 131$, $\alpha = 2$ e scelto $B = 10$ si usano i primi $2, 3, 5, 7$. Si ha

$$2^1 \equiv 2 \pmod{131}$$

$$2^8 \equiv 5^3 \pmod{131}$$

$$2^{12} \equiv 5 * 7 \pmod{131}$$

$$2^{14} \equiv 3^2 \pmod{131}$$

$$2^{34} \equiv 3 * 5^2 \pmod{131}.$$

Quindi

$$1 \equiv L_2(2) \pmod{130}$$

$$8 \equiv 3L_2(5) \pmod{130}$$

$$12 \equiv L_2(5) + L_2(7) \pmod{130}$$

$$14 \equiv 2L_2(3) \pmod{130}$$

$$34 \equiv L_2(3) + L_2(5) \pmod{130}.$$

Dalla seconda congruenza si ottiene $L_2(5) \equiv 46 \pmod{130}$.
Sostituendo nella terza congruenza si ha $L_2(7) \equiv -34 \equiv 96 \pmod{130}$.
Dalla quarta congruenza si può ottenere il valore di $L_2(3) \pmod{65}$ dato che $MDC(2, 130) \neq 1$; quindi si hanno due possibilità per $L_2(3) \pmod{130}$.
Si può controllare direttamente quale di esse è quella giusta oppure si può

usare la quinta congruenza e ottenere $L_2(3) \equiv 72 \pmod{130}$.
Ciò conclude la fase di precalcolo.
Ora, se si vuole determinare $L_2(37)$, provando con qualche esponente scelto a caso, si ha $37 * 2^{43} \equiv 3 * 5 * 7 \pmod{131}$ e allora

$$L_2(37) \equiv -43 + L_2(3) + L_2(5) + L_2(7) \equiv 41 \pmod{130}.$$

Infine $L_2(37) = 41$.

1.2.4 Attacco del compleanno al logaritmo discreto

Paradosso del compleanno

Date 23 *persone, la probabilità che due di esse festeggino il compleanno nello stesso giorno è poco superiore al* 50%.
Infatti: come prima cosa, si calcola la probabilità che i compleanni cadano tutti in giorni diversi. Poichè il compleanno della prima persona cadrà in un qualche giorno dell'anno, la probabilità che il compleanno della seconda persona cada in un giorno diverso è $(1 - 1/365)$. Se i compleanni delle prime due persone cadono in due giorni diversi, la probabilità che il compleanno della terza persona cada in un giorno diverso dai primi due è $(1 - 2/365)$. Quindi la probabilità che il compleanno di tutte e tre le persone cada in giorni diversi è $(1 - 1/365)(1 - 2/365)$. Dunque, continuando in questo modo, si ha che la probabilità che i compleanni delle 23 persone cadano tutti in giorni diversi è

$$(1 - \frac{1}{365})(1 - \frac{2}{365})...(1 - \frac{22}{365}) \cong 0,493$$

Allora la probabilità che almeno due compleanni cadano nello stesso giorno è

$$1 - 0,493 = 0,507.$$

Più in generale, si supponga che ci siano N oggetti (con N grande) e che ci siano r persone ognuna delle quali sceglie un oggetto. Allora:

$$\text{Prob (esista una collisione)} \approx 1 - e^{-r^2/2N}.$$

Questa è solo un'approssimazione che vale per N grande, mentre per N piccolo è consigliabile usare il prodotto dato sopra e ottenere una risposta esatta. Scegliendo $r^2/2N = ln2$ si ha che, se $r \approx 1,177\sqrt{N}$, allora la probabilità che almeno due persone scelgano lo stesso oggetto è del 50%. Quindi, se ci sono N possibilità e si hanno $\sqrt{N}$ scelte, la probabilità di avere una collisione è buona.

Attacco del compleanno al logaritmo discreto

Se p è un primo grande, per risolvere $\alpha^x \equiv \beta \pmod{p}$ con probabilità alta, si costruiscono due liste, entrambe di lunghezza $\sqrt{p}$:

1. la prima lista contiene i numeri $\alpha^k \pmod{p}$ per circa $\sqrt{p}$ valori di k scelti a caso;

2. la seconda lista contiene i numeri $\beta\alpha^{-l} \pmod{p}$ per circa $\sqrt{p}$ valori di l scelti a caso.

La probabilità che esista una corrispondenza tra qualche elemento della prima lista e qualche elemento della seconda è $\cong 50\%$. In questo caso si ha

$$\alpha^k \equiv \beta\alpha^{-l} \quad ossia \quad \alpha^{k+l} \equiv \beta \pmod{p}.$$

Quindi $x \equiv k + l \pmod{p-1}$ è il logaritmo discreto cercato.

Questo metodo ha un *tempo di esecuzione* e un'*occupazione di memoria* proporzionali a $\sqrt{p}$. È probabile (ma non garantito) che produca una risposta, cioè è un algoritmo probabilistico. Infine, dato che l'esponente k è scelto a caso, α^k deve essere ricalcolato ogni volta e ciò rende l'algoritmo più lento.

Capitolo 2

Firma digitale

La firma digitale rappresenta uno degli sviluppi più importanti della crittografia a chiave pubblica. Dopo un'introduzione su questo tipo di crittografia, si analizzeranno definizione e proprietà della firma digitale. Si studieranno infine le funzioni hash, importante strumento per la sua realizzazione.

2.1 Crittografia a chiave pubblica

La **crittografia a chiave pubblica** o crittografia asimmetrica è un tipo di crittografia in cui la crittografia e la decrittografia usano due chiavi differenti: una chiave pubblica e una privata. Più precisamente, il testo in chiaro viene trasformato in testo cifrato utilizzando l'algoritmo di crittografia con una delle due chiavi, al contrario, il testo in chiaro è ottenuto decrifrando il testo cifrato con l'altra chiave. Si suppone che sia computazionalmente impossibile determinare la chiave di decrittografia conoscendo l'algoritmo di crittografia e la chiave di crittografia. La crittografia a chiave pubblica può essere utilizzata sia per la segretezza che per l'autenticazione.

Segretezza

Come si vede in Figura 2.1, uno schema di crittografia a chiave pubblica prevede le seguenti componenti:

- *testo in chiaro:* il messaggio o i dati leggibili inviati come input all'algoritmo.
- *algoritmo di crittografia:* l'algoritmo che opera varie trasformazioni sul testo in chiaro.
- *chiave pubblica e privata:* sono scelte in modo che una venga utilizzata per la crittografia e l'altra per la decrittografia.
- *testo cifrato:* il messaggio codificato che dipende dal testo in chiaro e dalla chiave usata.
- *algoritmo di decrittografia:* accetta il testo cifrato e la chiave di decrittografia e genera il testo in chiaro.

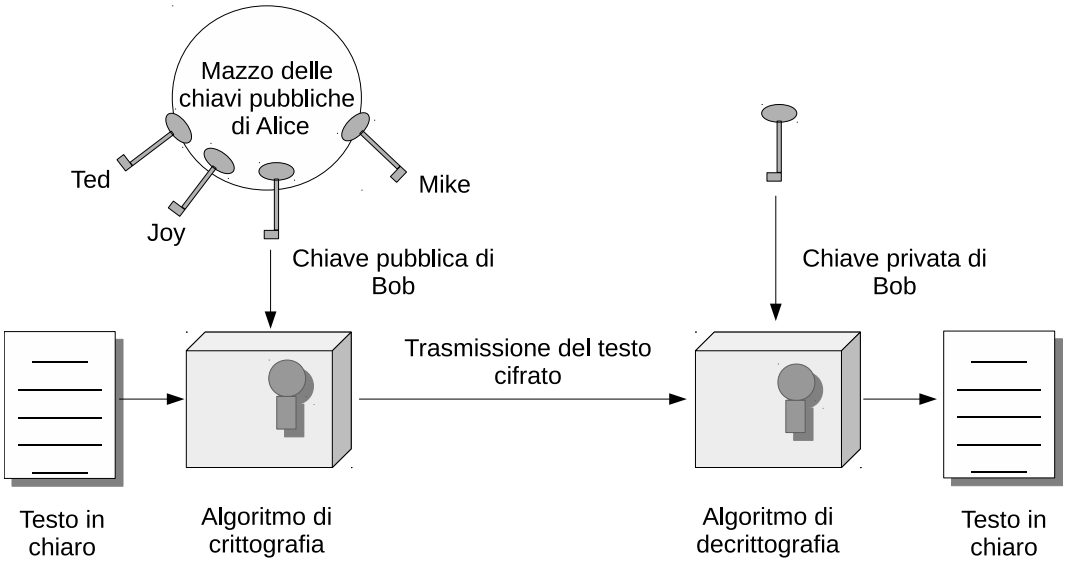

Figura 2.1: Crittografia a chiave pubblica: segretezza.

Nella fase preliminare si eseguono le seguenti operazioni:

- Ogni utente genera due chiavi: una utilizzata per la crittografia, l'altra per la decrittografia.
- Ogni utente inserisce una delle due chiavi in un registro pubblico o in un file accessibile e l'altra chiave viene mantenuta segreta.
- Se il mittente vuole inviare un messaggio confidenziale al destinatario, deve crittografarlo usando la chiave pubblica del destinatario.
- Quando il destinatario riceve il messaggio, deve decrittografarlo utilizzando la propria chiave privata. Nessun altro può decrittografare il messaggio perchè solo lui conosce la propria chiave privata.

Da ciò segue che è possibile distribuire le chiavi pubbliche, mentre quelle private vengono generate localmente da ogni utente e non devono mai essere distribuite. Fino a quando la chiave privata rimane segreta, tutte le comunicazioni sono sicure. Inoltre, un sistema può cambiare in qualsiasi momento la propria chiave privata e pubblicare la nuova chiave pubblica per sostituire quella vecchia.

La Figura 2.2 è utilizzata per descrivere gli elementi principali di uno schema di crittografia a chiave pubblica. Alice produce un messaggio in chiaro $X = [X_1, X_2, ..., X_m]$. Gli m elementi di X sono lettere di un alfabeto finito. Il messaggio deve essere inviato a Bob. Egli genera una coppia di chiavi correlate: una chiave pubblica PU_b e una chiave privata PR_b; la prima deve essere accessibile ad Alice, mentre la seconda è conosciuta solo da Bob. Utilizzando X e la chiave di crittografia PU_b come input, Alice genera il testo cifrato $Y = [Y_1, Y_2, ..., Y_n]$. Quindi

$$Y = E(PU_b, X)$$

Bob può invertire la trasformazione:

$$X = D(PR_b, Y)$$

Eva, conoscendo Y e PU_b, vuole scoprire X e/o PR_b; si suppone che conosca anche gli algoritmi di crittografia E e di decrittografia D.
Se fosse interessata a conoscere solo un determinato messaggio, si concentrerebbe sul recupero di X generando una stima di testo in chiaro $\hat{X}$, ma spesso è interessata a leggere anche tutti i messaggi futuri quindi cercherà di trovare PR_b generando una stima $\hat{P}R_b$.

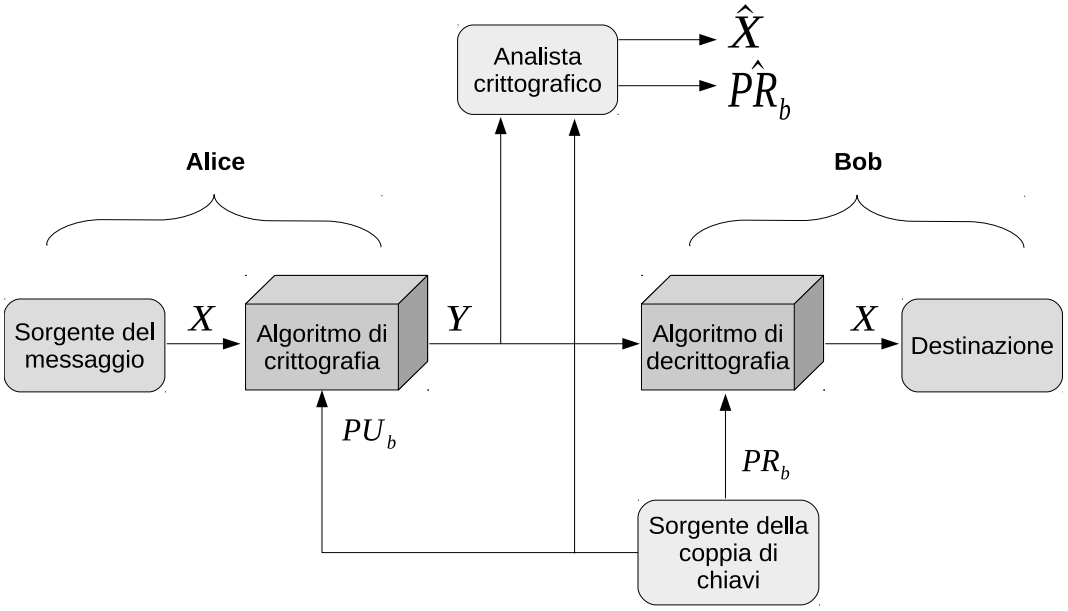

Figura 2.2: Sistemi crittografici a chiave pubblica: segretezza.

Autenticazione

Le Figure 2.3 e 2.4 mostrano l'ultilizzo della crittografia a chiave pubblica per garantire l'autenticazione.
Alice prepara un messaggio X per Bob e lo codifica utilizzando PR_a prima di mandarglielo, cioè:

$$Y = E(PR_a, X)$$

Bob decrittografa il messaggio usando PU_a, cioè:

$$X = D(PU_a, Y)$$

Poichè il messaggio è stato crittografato usando PR_a, solo Alice può averlo preparato. Quindi funge da firma digitale. Inoltre è impossibile alterare il messaggio senza accedere a PR_a, quindi questo è autenticato sia in termini di mittente che in termini di integrità dei dati.

Ciascun documento deve essere mantenuto in chiaro per poter essere utilizzato; una copia firmata deve però essere conservata in modo da poter verificare l'origine e il contenuto in caso di disputa.

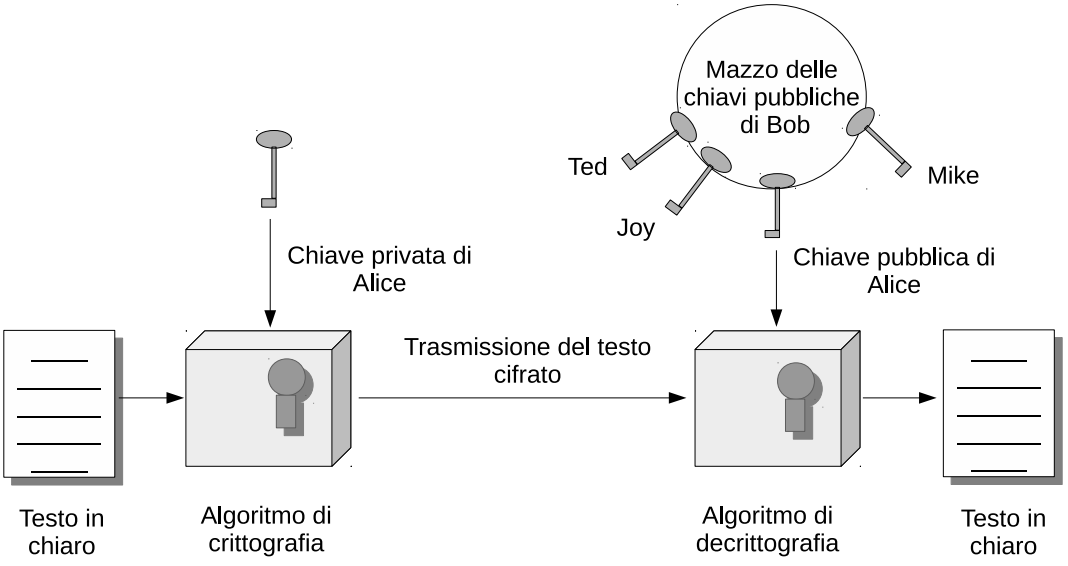

Figura 2.3: Crittografia a chiave pubblica: autenticazione.

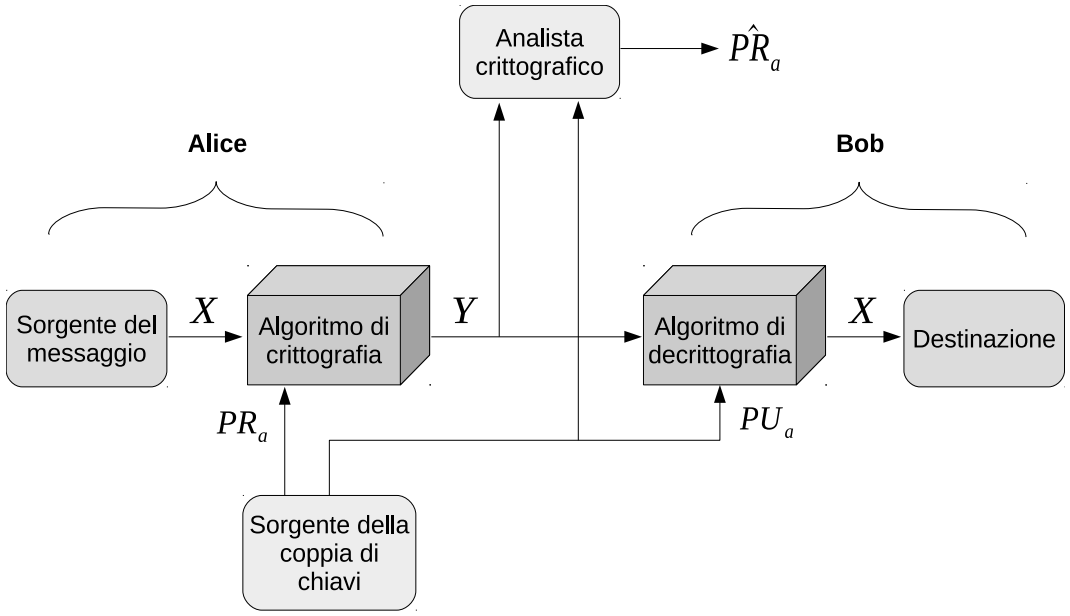

Figura 2.4: Sistemi crittografici a chiave pubblica: autenticazione.

2.2 Firma digitale

2.2.1 Definizione e proprietà

Definizione 2.1. La **firma digitale** è un meccanismo di autenticazione che permette all'autore di un messaggio di generare un codice che garantisca la sua identità in modo analogo a una firma tradizionale.

La firma digitale ha le seguenti proprietà:

- *Autenticità*: se un documento è firmato digitalmente si può essere certi dell'identità del firmatario;
- *Integrità*: sicurezza che il documento non sia stato modificato dopo essere stato firmato;
- *Non ripudiabilità*: il firmatario non può rinnegare la paternità dei documenti firmati; cioè la firma attesta la volontà del firmatario di sottoscrivere quanto contenuto nel documento e quindi anche la conoscenza del suo contenuto;

- *Non riusabilità*: la firma è parte integrante del documento e non deve essere utilizzabile su un altro documento.

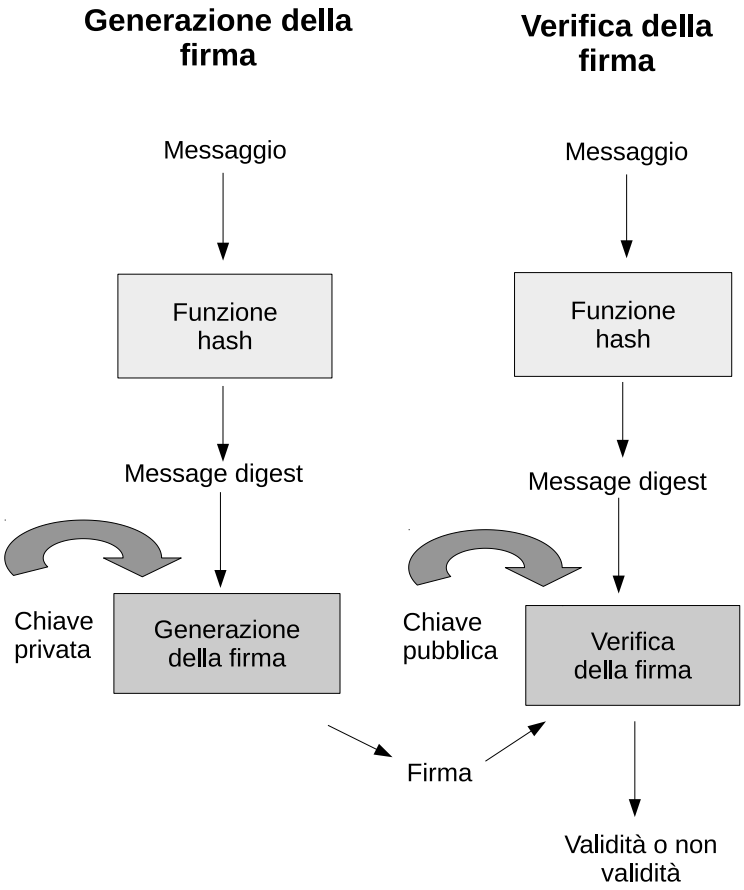

Figura 2.5: Processo di firma digitale

Come mostrato in figura 2.5, gli schemi di firma digitale sono composti da due fasi distinte: quella di generazione della firma e quella di verifica.
Il firmatario utilizza il processo di generazione per generare una firma digitale, mentre il verificatore utilizza il processo di verifica per verificare l'autenticità della firma.
Ogni firmatario possiede una chiave pubblica e una privata ed è il proprietario di quella coppia di chiavi. La chiave privata viene utilizzata nel processo di generazione della firma. Il proprietario della coppia di chiavi è l'unica

entità autorizzata ad usare la chiave privata per generare firme digitali. Per impedire ad altri soggetti di generare firme fraudolente, la chiave privata deve rimanere segreta. Invece la chiave pubblica viene usata nel processo di verifica della firma e non deve essere tenuta segreta. Chiunque può verificare un messaggio firmato correttamente utilizzando la chiave pubblica.
Inoltre, il firmatario può eventualmente verificare la firma utilizzando il processo di verifica della firma e la chiave pubblica associata. Questa verifica opzionale serve come controllo finale per rilevare eventuali gli errori di calcolo avvenuti durante la generazione della firma altrimenti non individuati; questo controllo è utile quando si firma un messaggio di alto valore, quando più utenti sono tenuti a verificare la firma, o nel caso in cui il verificatore verifichi la firma dopo molto tempo.
Sia per la generazione che per la verifica, il messaggio viene convertito in una stringa di lunghezza fissa (message digest) mediante una funzione hash. Ciò verrà spiegato nella sezione 2.3.3.
Sia il messaggio originale che la firma digitale sono messi a disposizione di un verificatore. Egli richiede:

- la garanzia che la chiave pubblica (utilizzata per verificare la firma) appartenga davvero al firmatario; cioè la garanzia che il firmatario sia il proprietario effettivo della coppia di chiavi;

- la garanzia che il firmatario fosse in possesso della chiave privata (utilizzata per generare la firma) nel momento in cui la firma è stata generata;

- la garanzia della validità della chiave pubblica, cioè deve essere matematicamente corretta.

Ottendendo queste garanzie, il verificatore è certo che, se la firma digitale viene verificata correttamente con la chiave pubblica, allora è valida. Tali garanzie sono richieste per le seguenti ragioni:

- se il verificatore non ottiene la garanzia che il firmatario è il proprietario effettivo della coppia di chiavi, ha la certezza dell'integrità dei dati ma non dell'autenticità;

- se un'infrastruttura a chiave pubblica non può fornire garanzie (ad un verificatore) che il proprietario di una coppia di chiavi ha dimostrato la conoscenza di una chiave privata corrispondente alla chiave pubblica del proprietario, allora può essere possibile per un'entità senza scrupoli di avere la propria identità legata ad una chiave pubblica che è (o è stata) utilizzata da un'altra parte. L'entità senza scrupoli può poi pretendere di essere la fonte di alcuni messaggi firmati dall'altra parte.

Tecnicamente, una coppia di chiavi utilizzata da un algoritmo di firma digitale potrebbe essere utilizzata anche per scopi diversi (ad esempio per la creazione della chiave); tuttavia, come specificato nel DSS, questo non deve succedere.

2.3 Funzioni hash

2.3.1 Definizione e proprietà

Definizione 2.2. Una **funzione hash** h è una funzione che prende in input un messaggio di lunghezza variabile e come output produce una stringa di lunghezza fissa detta *message digest*.

Quest'ultimo non dipende da una chiave ma è funzione solo del messaggio di input. Idealmente, ogni bit dell'output dipende da ciascun bit dell'input e ciò fornisce una funzionalità di rilevamento degli errori: una variazione in uno o più bit del messaggio genera una variazione nel message digest.

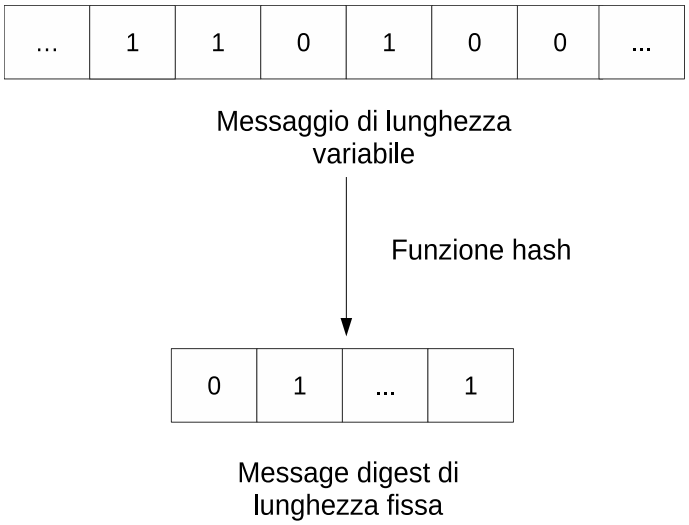

Figura 2.6: Funzione hash.

Le funzioni hash devono soddisfare le seguenti proprietà:

- Dato un messaggio m, deve essere possibile calcolare rapidamente il message digest $h(m)$;
- Deve essere computazionalmente intrattabile trovare un messaggio che abbia generato un dato hash; cioè dato y, trovare un m' tale che $h(m') = y$. In questo caso h è detta *funzione unidirezionale* o *resistente alle controimmagini*.
- Deve essere computazionalmente intrattabile trovare due messaggi che abbiano lo stesso hash; cioè m_1 e m_2 tali che $h(m_1) = h(m_2)$. In questo caso h è detta *fortemente resistente alle collisioni*.
  Questa proprietà si può indebolire richiedendo che H sia *debolmente resistente alle collisioni*, cioè deve essere computazionalmente intrattabile modificare un messaggio senza modificare il relativo hash. In altre parole, dato x deve essere computazionalmente intrattabile trovare un $x' \neq x$ con $H(x') = H(x)$.
  Ciò significa che, dal punto di vista computazionale, H deve comportarsi come se fosse iniettiva.

2.3.2 Semplice esempio di hash e accenno a SHA

Sia m un messaggio di lunghezza variabile L; lo si spezza in blocchi di n bit dove n è molto più piccolo di L. Con m_j si indicano questi blocchi di n bit e si ha $m = [m_1, m_2, ..., m_l]$ dove $l = [L/n]$ e l'ultimo blocco m_l è completato con degli zeri in modo da avere n bit. Si può considerare la matrice formata dai blocchi

$$m_j = [m_{j1}, m_{j2}, m_{j3}, ..., m_{jn}]$$

scritti come vettori riga, dove ogni m_{ji} è un bit. L'hash $h(m)$ è formato da n bit ed è definito in modo che l'i-esimo bit sia la somma modulo 2 degli elementi della i-esima colonna di questa matrice, ossia $h_i = m_{1i} \oplus m_{2i} \oplus ...m_{li}$. Questa funzione hash prende in input un messaggio di lunghezza variabile e come output produce un message digest di n bit. Tuttavia non è crittograficamente sicura, dato che è molto facile trovare due messaggi che hanno lo stesso hash. Le funzioni hash crittografiche usate in pratica, utilizzano sempre molte operazioni sui bit, in modo che sia più difficile trovare collisioni. Per esempio, l'algoritmo SHA (Secure Hash Algorithm), utilizza le seguenti operazioni: rotazione (o scorrimento), $\wedge$ (AND), $\vee$ (OR), $\oplus$ (somma modulo 2 bit a bit), $\neg$ (cambia gli 1 in 0 e viceversa)).

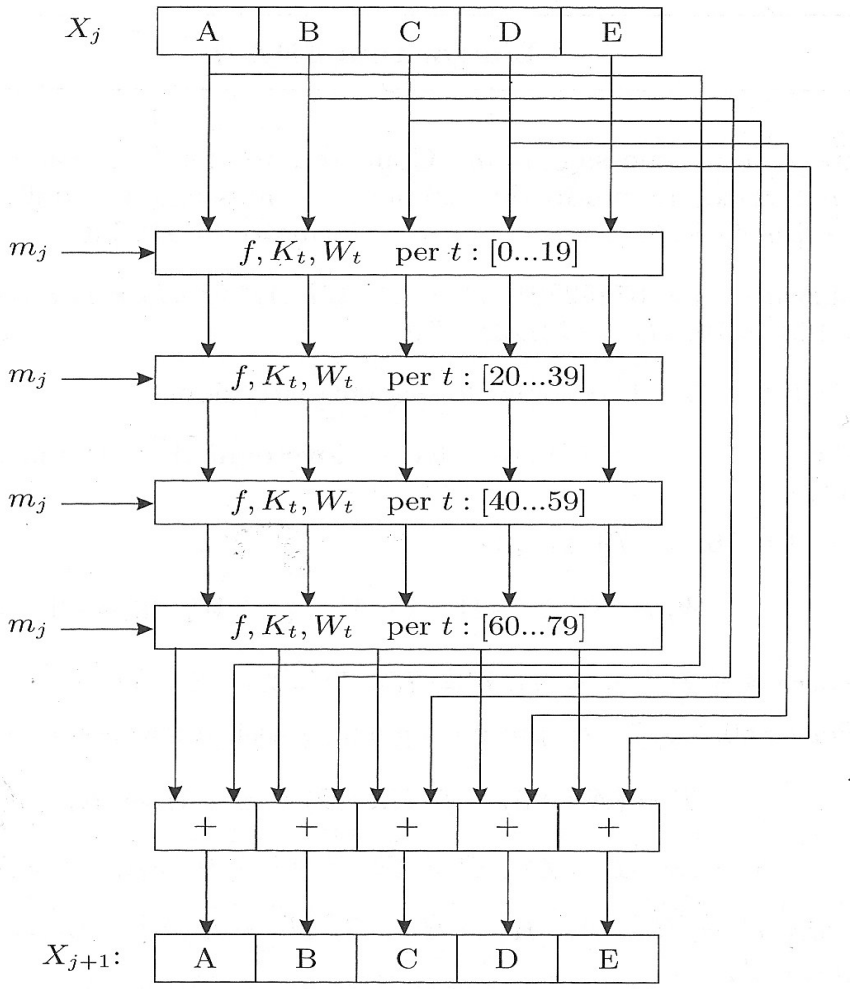

Figura 2.7: Operazioni eseguite da SHA-1 su un blocco di messaggio m_j.

2.3.3 Funzioni hash per firme digitali

Le funzioni hash crittografiche vengono usate principalmente nei protocolli per le firme digitali; infatti dato che una firma digitale può essere lunga quanto il documento che deve essere firmato, è molto più efficiente e conveniente firmare l'hash del documento piuttosto che il documento intero.

La procedura di **generazione** della firma avviene nel seguente modo.
Sia h una funzione hash pubblica e m un messaggio. Alice:

1. Calcola l'hash $h(m)$ (che ha dimensioni inferiori rispetto a m e può essere firmato molto più velocemente rispetto all'intero messaggio);

2. Firma l'hash $h(m)$ con la sua chiave privata, ottendendo $sig(h(m))$, e invia l'hash appena firmato insieme ad m.

La procedura di **verifica** della firma avviene nel seguente modo. Bob:

1. Riceve m e l'hash firmato;

2. Calcola l'hash del messaggio ricevuto;

3. Utilizza la chiave pubblica di Alice per decifrare l'hash firmato che ha ricevuto e lo confronta con l'hash che ha calcolato nel punto 2. Se i due valori hash coincidono allora Bob ha verificato che il messaggio non è stato alterato. Inoltre, poichè la chiave pubblica di Alice è stata utilizzata per la decrittografia, Bob è anche sicuro dell'identità di Alice.

Questo sistema, rispetto allo schema originale, presentato ad esempio in [6] o [7], è più veloce da creare e richiede meno risorse per la trasmissione o memorizzazione.

Per ciò che riguarda la sicurezza, si supponga che Eva abbia il messaggio $(m, sig(h(m))$ firmato da Alice. Per aggiungere la firma di Alice su un altro messaggio m', Eva deve avere che $sig(h(m') = sig(h(m)$, in particolare deve avere $h(m') = h(m)$. Se la funzione hash è unidirezionale allora Eva avrà difficoltà a trovare un qualsiasi m' che soddisfi l'equazione e le probabilità che il messaggio m', in suo possesso, vada bene sono molto basse. Inoltre, poiché le funzioni hash devono essere fortemente resistenti alle collisioni, è improbabile che Eva riesca a trovare due messaggi $m_1 \neq m_2$ che ammettano la stessa firma. Nel caso in cui riuscisse a fare ciò, farebbe firmare ad Alice il messaggio m_1 e trasferirebbe la firma a m_2, ma Alice si insospettirebbe perchè è estremamente improbabile che m_1 (e m_2) siano messaggi dotati di senso.

2.3.4 Attacco del compleanno alle firme digitali

Se la lunghezza dell'hash è troppo piccola, Eva può ingannare Alice.
Alice sta per firmare un documento m usando uno schema nel quale si firma un hash del documento; si supponga che l'output della funzione hash sia, ad esempio, di soli 50 bit. Alice teme che Eva possa ingannarla facendole firmare un contratto aggiuntivo fraudolento, ma la probabilità che questo abbia lo stesso hash di m è una su 2^{50} (che è circa una su 10^{15}).
Eva trova in m 30 punti in cui può apportare leggere modifiche quali riformulare una frase o aggiungere uno spazio alla fine di ogni riga, senza modificarne il significato; in ognuno di questi Eva ha due possibilità: tenere la versione originale oppure effettuare la modifica. Quindi ha a disposizione 2^{30} documenti essenzialmente identici a m e contro i quali Alice non avrebbe obiezioni.
A questo punto calcola l'hash di ognuno di questi 2^{30} documenti e li conserva; allo stesso modo formula 2^{30} versioni del contratto fraudolento e conserva i rispettivi hash.
Se si considera il problema del compleanno generalizzato con $r = 2^{30}$ e $n = 2^{50}$, si ottiene $r = \sqrt{\lambda n}$ con $\lambda = 2^{10} = 1024$. Questo significa che la probabilità che m e il contratto fraudolento abbiano lo stesso hash è circa $1 - e^{-1024} \approx 1$.
Ora Eva chiede ad Alice di firmare m con l'intenzione di apporre la firma al contratto fraudolento. Poichè i due documenti hanno lo stesso hash, la firma è valida per entrambi. Ma Alice (sospettosa) rimuove una virgola e firma il documento modificato, che ha un hash completamente diverso dal documento fraudolento. Così Eva non è riuscita ad ingannare Alice (in quanto è sostanzialmente impossibile trovare una versione del documento fraudolento che abbia lo stesso hash del messaggio firmato da Alice).

Per prevenire questo tipo di attacco è necessario usare funzioni hash con un output di lunghezza almeno doppia rispetto a quella che si potrebbe ritenere necessaria per evitare un attacco per forza bruta, poichè l'effetto dell'attacco

del compleanno è un dimezzamento virtuale del numero di bit presenti nei digest di messaggi.
Nel caso in cui la funzione hash non soddisfi questa condizione, il modo migliore per sventare un attacco del compleanno è cambiare leggermente un documento subito prima di firmarlo.

Capitolo 3

Algoritmo DSA

DSA, ovvero Digital Signature Algorithm, è un algoritmo per la firma digitale proposto dal NIST (National Institute of Standard and Technology) nell'agosto del 1991 e definitivamente adottato nel 1994 [5].

È uno schema di firma con appendice, cioè il messaggio non può essere recuperato a partire dalla firma e deve essere incluso nella procedura di verifica. Inoltre la firma è applicata a un message digest generato da una funzione hash.

La sicurezza di questo algoritmo si basa sulla difficoltà di calcolare i logaritmi discreti. Può essere utilizzato solo per le operazioni relative alle firme digitali, non per la crittografia dei dati o per la distribuzione delle chiavi.
L'algoritmo DSA richiede una serie di parametri.

3.1 Parametri

Per generare una firma digitale con l'algoritmo DSA è necessario utilizzare una serie di parametri di dominio, una chiave privata, un numero segreto, i dati da firmare e una funzione hash. Una firma digitale viene verificata utilizzando gli stessi parametri di dominio, una chiave pubblica che è matematicamente associata a quella privata, i dati da verificare e la stessa funzione

hash che è stata utilizzata durante la generazione della firma.
Questi parametri sono definiti come segue:

- p: numero primo tale che $2^{L-1} < p < 2^L$ dove L è la lunghezza in bit di p;
  Il problema del logaritmo discreto deve essere difficile da risolvere per questo valore di p.

- q: numero primo tale che $q \mid (p-1)$ con $2^{N-1} < q < 2^N$ dove N è la lunghezza in bit di q;
  I valori di L e di N sono forniti nella sezione 3.1.1.

- g: radice primitiva mod p;

- a: chiave privata che deve rimanere segreta; è un numero intero generato in modo casuale o pseudocasuale tale che $0 < a < q$ cioè $a \in [1, q-1]$

- k: numero segreto unico per ogni messaggio; è un numero intero generato in modo casuale o pseudocasuale tale che $0 < k < q$ cioè $k \in [1, q-1]$.

- h: funzione hash.

3.1.1 Scelta dei formati dei parametri

Questo Standard specifica le seguenti scelte per la coppia (L, N):

$$L = 1024, N = 160;$$
$$L = 2048, N = 224;$$
$$L = 2048, N = 256;$$
$$L = 3072, N = 256.$$

Il processo si articola in 3 fasi:

1. Inizializzazione
2. Generazione della firma
3. Verifica della firma

3.2 Inizializzazione

In questa fase vengono generate le chiavi da usare in DSA. Alice:

1. Calcola $\alpha \equiv g^{(p-1)/q} \pmod{p}$. Quindi $\alpha^q \equiv 1 \pmod{p}$.
2. Calcola $\beta \equiv \alpha^a \pmod{p}$.
3. Pubblica (p, q, α, β) e tiene segreto a.

3.3 Generazione della firma

Alice vuole inviare un messaggio m a Bob; lo firma eseguendo le seguenti operazioni:

1. Calcola $r = (\alpha^k \pmod{p}) \pmod{q}$.
2. Calcola $s \equiv k^{-1}(h(m)+ar) \pmod{q}$ dove k^{-1} è l'inverso moltiplicativo di k rispetto alla moltiplicazione modulo q, quindi $0 < k^{-1} < q$.
   I valori di r e s devono essere controllati per determinare se uno dei due vale 0; se ciò accade deve essere generato un nuovo valore di k e la firma deve essere ricalcolata. È comunque estremamente improbabile che $r = 0$ o $s = 0$ se le firme sono state create correttamente.
3. Ottiene la firma (r, s) che invia a Bob insieme al messaggio m.

3.4 Verifica della firma

Bob riceve il messaggio m e la firma (r, s) che verifica eseguendo le seguenti operazioni:

1. Scarica la chiave pubblica di Alice β e p, q, α.
2. Calcola $u_1 \equiv s^{-1}h(m) \pmod{q}$ e $u_2 \equiv s^{-1}r \pmod{q}$.
3. Calcola $v = (\alpha^{u_1}\beta^{u_2} \pmod{p}) \pmod{q}$.
4. Accetta la firma se e solo se $v = r$.

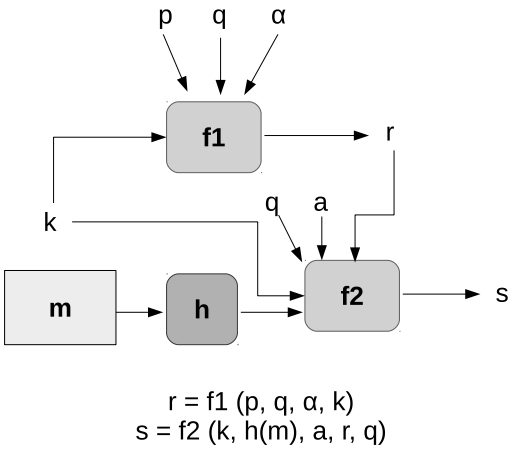

Figura 3.1: Generazione della firma.

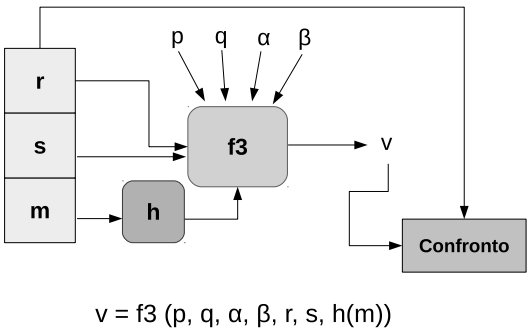

Figura 3.2: Verifica della firma.

La procedura di verifica funziona, vediamo infatti che, se la firma è stata generata in modo corretto, $v = r$.

Dimostrazione. Dalla definizione di s si ottiene

$$h(m) \equiv (-ar + ks) \pmod{q}$$

e moltiplicando entrambi i membri per s^{-1} si ha

$$s^{-1}h(m) \equiv (-ars^{-1} + k) \pmod{q}.$$

Quindi

$$\begin{aligned} k &\equiv s^{-1}h(m) + ars^{-1} \pmod{q} \\ &\equiv u_1 + au_2 \pmod{q}. \end{aligned}$$

Perciò

$$a^k = \alpha^{u_1+au_2} = (\alpha^{u_1}\beta^{u_2} \pmod{p}) \pmod{q}.$$

Quindi

$$v = r.$$

□

In caso contrario, cioè se $v \neq r$, significa che il messaggio o la firma potrebbero essere stati modificati; ci potrebbe esser stato un errore nel processo di generazione oppure un impostore potrebbe aver tentato di falsificare la firma. Quindi questa deve essere considerata non valida!

Bob conosce la chiave pubblica β, la firma (r, s) e il messaggio m e poiché p, q e α sono parametri comuni, li conosce. Inoltre, dato che $\beta \equiv \alpha^a \pmod{p}$ e $r = (\alpha^k \pmod{p}) \pmod{q}$, per determinare a e k potrebbe usare il logaritmo discreto. Ma, visto che si pensa che il logaritmo discreto sia un problema difficile, si pensa che sia computazionalmente infattibile determinarli.
Anche supponendo di riuscire a calcolare il logaritmo discreto mod p, conoscere r non permette di ottenere il valore di k ma solo quello di k mod q; ci sono circa $2^{512-160} = 2^{342}$ numeri mod p che si riducono allo stesso valore mod q e questo garantisce una maggiore sicurezza anche nel caso in cui un

avversario sia in grado di risolvere il problema del logaritmo discreto mod p. Quindi partendo da s non è possibile trovare a.
È fondamentale che l'intero a venga tenuto segreto dato che, chiunque lo conosca, può firmare qualsiasi documento.
Se Alice usasse due volte lo stesso valore di k sarebbe possibile trovare a. La verifica richiede solo due elevamenti a potenza in modo da ridurre i tempi nel caso in cui si debbano verificare molte firme in un tempo breve.

Poiché questi standard devono durare decenni si cerca di crearne uno flessibile in cui la componente matematica possa essere sostituita senza cambiarlo interamente; è per questo motivo che l'algoritmo DSA utilizza proprio il logaritmo discreto dato che può essere realizzato matematicamente su gruppi diversi. Inoltre, se si vuole aumentare la sicurezza si può utilizzare sempre il logaritmo discreto ma su un gruppo in cui sia più difficile calcolarlo.

Bibliografia

[1] W. Stallings, *Crittografia e sicurezza delle reti*, McGraw-Hill, 2007.

[2] W. Trappe - L.C.Washington, *Crittografia con elementi di teoria dei codici*, Pearson-Prentice Hall, 2009.

[3] N. Koblitz, *A Course in Number Theory and Cryptography*, Springer, 1994.

[4] Digital Signature Illustrated by J.Orlin Grabbe.
https://billstclair.com/grabbe/digsig.htm

[5] FIPS 186-4, *Digital Signature Standard (DSS)*, Federal Information Processing Standards Publication 186, U.S. Department of Commerce/National Institute of Standards and Technology, July 2013.

[6] W. Diffie and M. Hellman, "New directions in cryptography", *IEEE Transaction in Information Theory*, Vol 22, pp. 644-654, 1976.

[7] R.L. Rivest, A. Shamir, L. Adleman, *"A method for obtaining digital signature and public-key cryptosystem"*, Communication of ACM Vol 21, pp. 120-126, 1978.