Crittografia Moderna

L'algoritmo DH (Diffie-Hellman)

L'algoritmo Diffie-Hellman risale al 1976 ed è quindi uno dei più antichi algoritmi a chiave pubblica; gli autori furono anche i primi a proporre l'idea dei cifrari a chiave pubblica. L'algoritmo è particolarmente adatto alla generazione di una chiave segreta tra due corrispondenti che comunicano attraverso un canale non sicuro (pubblico). La sua sicurezza si basa sulla complessità computazionale del calcolo del logaritmo discreto.
Supponiamo di avere i soliti Aldo e Bruno che vogliono scambiarsi una chiave segreta in modo sicuro.
Aldo genera e comunica pubblicamente un numero primo N molto elevato (p.es. 1024 bit, circa 300 cifre decimali) e un generatore g (che esiste sicuramente essendo N primo); si potrebbero anche usare due numeri N e g pubblicati da un qualche ente.
A questo punto:

1. Aldo genera un numero casuale $a < N$ e calcola $A = g^a \bmod N$. Il numero A viene comunicato pubblicamente a Bruno.
2. In modo del tutto analogo Bruno genera due numeri b e $B = g^b \bmod N$ e invia B ad Aldo.
3. Aldo calcola il numero $k = B^a \bmod N$.
4. Bruno calcola $k = A^b \bmod N$.

Inutile aggiungere che le due chiavi k sono uguali! Infatti entrambe valgono $g^{ab} \bmod N$.
A questo punto Aldo e Bruno possono usare la chiave per comunicare con un cifrario simmetrico p.es. il DES.

Esempio pratico

Aldo e Bruno scelgono $N = 17$ e $g = 3$ (3 è generatore di 17).

1. Aldo sceglie $a = 6$ e quindi $A = 3^6 \bmod 17 = 15$.
2. Biagio sceglie $b = 11$ e quindi $B = 3^{11} \bmod 17 = 7$.
3. Aldo calcola $k = 7^6 \bmod 17 = 9$.
4. Biagio calcola $k = 15^{11} \bmod 17 = 9$.
5. La chiave segreta è quindi 9.

Sicurezza

La cosa importante per la sicurezza è che un terzo che intercettasse i quattro numeri N, g, A, B non sarebbe in grado di ottenere $k = g^{ab}$ non conoscendo né a né b. In effetti $a = \log_g(A)$ e $b = \log^g(B)$ ma essendo il calcolo del logaritmo discreto computazionalmente proibitivo come una fattorizzazione è pressoché impossibile calcolare questi logaritmi per numeri di 1024 bit.

Come altri sistemi a chiave pubblica anche DH è però esposto all'attacco del terzo uomo interposto. Nel nostro esempio immaginiamo che Carlo intercetti il numero A che Aldo invia a Bruno e, fingendo di essere Bruno, generi un suo numero B e lo invii ad Aldo. A questo punto Aldo e Carlo generano la chiave segreta k e comunicano via DES; e Carlo può tranquillamente leggere tutti i messaggi che Aldo crede di inviare a Bruno! Peggio ancora:

Carlo può ripetere il gioco anche con Bruno fingendosi Aldo e di qui in avanti, intercettare e leggere tutta la corrispondenza tra gli inconsapevoli Aldo e Bruno.

Per prevenire simili attacchi la soluzione è anche qui quella di usare un ente certificatore che garantisca l'identità dei corrispondenti. Aldo e Bruno potranno p.es. identificarsi con un meccanismo di firma digitale, utilizzando le chiavi pubbliche fornite dall'ente certificatore.

Sistema RSA

L'algoritmo RSA è stato descritto nel 1977 da *Ron Rivest*, *Adi Shamir* e *Len Adleman* al MIT; le lettere RSA vengono proprio dalle iniziali dei cognomi.

Clifford Cocks, un matematico britannico che lavorava per un dipartimento di spionaggio, il GCHQ, descrisse un sistema equivalente in un documento interno nel 1973. I documenti furono posti sotto segreto e, visto il costo relativamente alto delle macchine necessario a quel tempo per implementarlo, non ci furono ulteriori indagini né prove pratiche e la cosa fu considerata come una curiosità, per quanto se ne sa. La scoperta di Cocks fu resa pubblica solo nel 1997.

Nel 2005 un gruppo di ricerca riuscì a scomporre un numero di 640 bit (193 decimali) in due numeri primi da 320 bit, impiegando per cinque mesi un cluster Opteron con 80 processori da 2,2 GHz, potenzialmente decifrando un messaggio codificato con *RSA-640*.

L'algoritmo all'opera

RSA è basato sul problema complesso della fattorizzazione in numeri primi. Il suo funzionamento base è il seguente:

1. si scelgono due numeri primi, p e q , abbastanza grandi da garantire sicurezza
2. si calcola il loro prodotto $n = p*q$, chiamato modulo (dato che tutta l'aritmetica seguente è modulo n)
3. si sceglie poi un numero e , (chiamato esponente pubblico), più piccolo e co-primo di $(p-1)*(q-1)$
4. si calcola il numero d , (chiamato esponente privato) tale che

   $e * d \equiv 1 \pmod{(p-1)(q-1)}$

La chiave pubblica è (n, e) mentre la chiave privata è (n, d). I fattori p e q possono essere distrutti, anche se spesso vengono mantenuti all'interno della chiave privata. La forza dell'algoritmo è che per calcolare d da e (così come il contrario) non basta la conoscenza di n, ma serve il numero $(p-1)(q-1)$; infatti fattorizzare (cioè scomporre un numero nei suoi divisori) è in operazione è molto lenta, soprattutto se n è un numero grande a sufficienza, poiché non si conoscono algoritmi efficienti.

Un messaggio m viene cifrato attraverso l'operazione $m^e \ mod(n)$, mentre il messaggio c viene decifrato con $c^d = m^{e*d} = m^1 \ mod(n)$. Il procedimento funziona solo se la chiave e utilizzata per cifrare e la chiave d, utilizzata per decifrare sono legate tra loro dalla relazione $e*d \equiv 1 \ mod(n)$, e quindi quando un messaggio viene cifrato con una delle due

chiavi può essere decifrato solo utilizzando l'altra. Tuttavia proprio qui si vede la debolezza dell'algoritmo: si basa sull'assunzione mai dimostrata (nota come assunzione RSA, o RSA assumption) che il problema di calcolare $\sqrt[e]{c} \ mod \ n$ con n numero composto di cui non si conoscono i fattori sia computazionalmente non trattabile.

La firma digitale non è altro che l'inverso: il messaggio viene crittato con la chiave privata, in modo che chiunque possa, utilizzando la chiave pubblica conosciuta da tutti, decifrarlo e, oltre a poterlo leggere in chiaro, essere certo che il messaggio è stato mandato dal possessore della chiave privata corrispondente a quella pubblica utilizzata per leggerlo.

Per motivi di efficenza e comodità normalmente viene inviato il messaggio in chiaro con allegata la firma digitale di un *hash* del messaggio stesso; in questo modo il ricevente può direttamente leggere il messaggio (che è in chiaro) e può comunque utilizzare la chiave pubblica per verificare che l'*hash* ricevuto sia uguale a quello calcolato localmente sul messaggio ricevuto. Se i due *hash* corrispondono anche il messaggio completo corrisponde (questo, ovviamente, solo se l'*hash* utilizzato è crittograficamente sicuro).

Fondamenti matematici

La decrittazione del messaggio è assicurata grazie ad alcuni teoremi matematici, infatti dal calcolo noi otteniamo: $c^d = (m^e)^d = m^{e*d} \ \ (mod \ m)$

Ma sappiamo che $e*d \equiv 1 \ (mod \ p-1)$ *e che* $e*d \equiv 1 \ (mod \ q-1)$ quindi, per il piccolo teorema di Fermat[1] $m^{e*d} \equiv m \ (mod \ p)$ ed anche $m^{e*d} \equiv m \ (mod \ q)$

Siccome p e q sono numeri diversi e primi, possiamo applicare il *Teorema cinese del resto*[2], ottenendo che

$m^{e*d} \equiv m \ (mod \ p*q)$

e quindi che

$c^d \equiv m \ (mod \ n)$

Esempio

Ecco un esempio di criptazione e decriptazione RSA. I numeri scelti sono artificialmente primi, ma nella realtà sono usati numeri dell'ordine di 10^{100}.

Generazione delle chiavi:

1. $p = 61$ e $q = 53; \ (p-1)*(q-1) = 3120$
2. $n = p*q = 61*53 = 3233$
3. $e = 17; \ e < n$ inoltre non è divisibile né per 61, né per 53 (in questo caso e è primo, ma in generale non è necessario)
4. $d = 2753$ infatti $e*d = 2753*17 = 46801 \equiv 1 \ (mod \ (p-1)*(q-1)) \equiv 1 (mod \ 3120)$ poiché $46801/3120 = 15$ con resto 1

Quindi abbiamo che la chiave pubblica è la coppia $(3233, 17)$, e la chiave privata è data dalla coppia $(3233, 2753)$

1 Si vedano la dispensa sui *"Richiami algebrici"*
2 Si vedano la dispensa sui *"Richiami algebrici"*

Crittazione e decrittazione:

Prendiamo ora in considerazione il messaggio $m = 123$ e crittiamolo per ottenere il messaggio crittato c , ovviamente possiamo usare i numeri 3233 e 17, ma non 2753 che fa parte della chiave privata.

Quindi abbiamo $c = m^e \pmod n = 123^{17} \pmod{3233} = 885$

abbiamo ottenuto $c = 855$.

Ora percorriamo la decrittazione di c per ottenere m. Utilizzeremo il valore 2753, componente essenziale della chiave privata.

Dunque: $m = c^d \pmod n = 855^{2753} \pmod{3233} = 123$

Crittosistema di El Gamal

Consideriamo un alfabeto di s lettere. Sia P l'insieme dei messaggi in chiaro, C l'insieme dei messaggi criptati, entrambi costituiti da pacchetti di lunghezza k. Sia

$$e : P \rightarrow C$$

la funzione di codifica e

$$d : C \rightarrow P$$

la funzione di decodifica.

Descriviamo l'algoritmo.
In base alle necessità si costruisce il campo $F_{p^n} = F_p[x] / (f)$, con $f(x)$ primitivo di grado n e sia g una radice primitiva. Ovviamente dobbiamo avere che $s^k < p^n - 1$

Input dell'algoritmo: $p, g, x \in P$

Output dell'algoritmo: $y \in C, x \in P$

Passi dell'algoritmo:

1. A sceglie $x_A = random(p-1)$ e pubblica $a = g^{x_A} \pmod p$
2. B sceglie $x_B = random(p-1)$ e pubblica $b = g^{x_B} \pmod p$

   FASE DI CODIFICA:
3. A calcola $e(x) = x * b^{x_A} \equiv y \pmod p$ e pubblica y

   FASE DI DECODIFICA:
4. B calcola $d(y) = y * a^{-x_B} \equiv (x * b^{x_A}) * a^{-x_B} \equiv x \pmod p$

Esempio numerico

Siano $q = 31$ e $g = 3$ (una radice positiva):

1. Alice sceglie $x_a = 10$, calcola $a = 3^{10} \equiv 25 \pmod{31}$ e pubblica $a = 25 \pmod{31}$
2. Bob sceglie $x_b = 7$, calcola $b = 3^7 \equiv 17 \pmod{31}$ e pubblica $a = 17 \pmod{31}$
3. Sia $x \in P$, $x = 12$. Allora $y = e(x) = x * b^{x_A} = 12 * (17)^{10} \equiv 21 \pmod{31}$

4. $d(y) = y * a^{-x_B} = 21 * (27)^{-7} \equiv 21 \pmod{31}$

Firma Digtale

La firma digitale, o *firma elettronica qualificata*, basata sulla tecnologia della crittografia a chiavi asimmetriche, ha lo stesso scopo della firma convenzionale.

La prima cosa che può venire in mente, pensando di simulare quello che accade con carta e penna, è di creare una firma digitale che sia una rappresentazione elettronica di quella convenzionale. Una firma digitale siffatta non sarebbe altro che un insieme di byte, scelti in un qualche modo, da inserire nel file di un documento da firmare. Questa soluzione semplicemente non garantisce alcuna sicurezza poiché chiunque può tagliare la parte di file contenente questa "firma" per poi aggiungerla illegalmente ad un altro file.

Le proprietà che vorremmo avesse una firma digitale sono:

1. semplicità di creazione da parte del legittimo firmatario;
2. impossibilità di falsificazione;
3. semplicità di verifica da parte di chiunque.

Sistemi per la creazione e la verifica di firme digitali

Il sistema per la creazione e la verifica di firme digitali sfrutta le caratteristiche dei sistemi crittografici a due chiavi. Un sistema crittografico garantisce la riservatezza del contenuto dei messaggi, rendendoli incomprensibili a chi non sia in possesso di una "chiave" (intesa secondo la definizione crittologica) per interpretarli. Nei sistemi crittografici a due chiavi, detti anche a chiave pubblica o asimmetrici, ogni utente ha una coppia di chiavi: una chiave privata, da non svelare a nessuno, con cui può decodificare i messaggi che gli vengono inviati, e una chiave pubblica, che altri utenti utilizzano per codificare i messaggi da inviargli. Per ogni utente, le due chiavi vengono generate da un apposito algoritmo con la garanzia che la chiave privata sia la sola in grado di poter decodificare correttamente i messaggi codificati con la chiave pubblica associata. Lo scenario in cui un mittente vuole spedire un messaggio ad un destinatario in modalità sicura è il seguente: il mittente utilizza la chiave pubblica del destinatario per la codifica del messaggio da spedire, quindi spedisce il messaggio codificato al destinatario; il destinatario riceve il messaggio codificato e adopera la sua chiave privata per ottenere il messaggio "in chiaro".

Grazie ad un'ulteriore proprietà delle due chiavi, inversa rispetto a quella descritta, un sistema di questo tipo è adatto anche per ottenere dei documenti firmati, infatti: la chiave pubblica di un utente è la sola in grado di poter decodificare correttamente i documenti codificati con la chiave privata di quell'utente. Se un utente vuole creare una firma per un documento, procede nel modo seguente: con l'ausilio di una funzione *hash* ricava l'impronta digitale del documento, il *message digest*, un file di dimensione fissa che riassume le informazioni contenute nel documento, dopodiché utilizza la propria chiave privata per codificare quest'impronta digitale: il risultato di questa codifica è la creazione di una firma. La funzione *hash*, è fatta in modo da rendere minima la probabilità che da testi diversi si possa ottenere il medesimo valore dell'impronta, inoltre è *one-way*, a senso unico,

questo significa che dall'impronta è pressoché impossibile ottenere nuovamente il testo originario. La firma prodotta dipende dall'impronta digitale del documento e, quindi, dal documento stesso, oltre che dalla chiave privata dell'utente. A questo punto la firma viene allegata al documento.

Chiunque può verificare l'autenticità di un documento: per farlo, decodifica la firma del documento con la chiave pubblica del mittente, ottenendo l'impronta digitale del documento, e poi confronta questa con quella che si ottiene applicando la funzione *hash*, pubblica, al documento; se le due impronte sono uguali, l'autenticità del documento è garantita.

Vantaggi e svantaggi della crittografia a chiave pubblica

Vantaggi

La chiave pubblica può essere trasmessa tramite un canale insicuro, in quanto la sua conoscenza da parte di terzi non è sufficiente a mettere in pericolo la sicurezza dei dati crittografati con essa. Ciascuna chiave segreta resta sotto la responsabilità del solo utente proprietario. In un sistema a chiave pubblica deve esistere una coppia di chiavi per ogni possibile utente.

Svantaggi

Generalmente gli algoritmi asimmetrici sono molto più lenti da eseguire, rispetto a quelli simmetrici, per cui risulta poco agevole il loro uso per crittografare lunghi messaggi.