Appunti di Esercitazioni

Corso di Sicurezza dei Sistemi Informatici
Corso di laurea specialistica (LS) in Ingegneria Informatica
5-CFU, COD. 9010

Gerardo Pelosi

Dipartimento di Ingegneria dell'Informazione e Metodi Matematici
Università degli Studi di Bergamo
email: gerardo.pelosi@unibg.it

Indice

Capitolo 1

Cenni Essenziali di Teoria dei Numeri e Algebra dei Gruppi

1.1 Elementi di Teoria dei Gruppi

Una struttura algebrica (interna) è una coppia $< S, \Omega >$, dove S è un insieme, detto sostegno della struttura, Ω è un insieme finito di operazioni interne su S, assiomi (caratteristici delle varie strutture) specificano le proprietà di cui le operazioni devono godere.
Se il sostegno S della struttura è finito, si dice che la struttura è finita e $|S|$ si chiama ordine della struttura.
Due tipi di strutture sono particolarmente importanti in crittografia: i gruppi ed i campi.

Def 1.1.1. *(Gruppo).* Si dice *gruppo* una coppia $< G, * >$, dove G è un insieme (sostegno del gruppo), $*$ è un'operazione interna binaria su G, cioè una legge che ad ogni coppia (a, b) di elementi di G associa uno ed un solo elemento $c \in G$ (denotato $a * b$), per la quale valgono i seguenti assiomi:

- proprietà associativa: $\forall a, b, c \in G \ (a * b) * c = a * (b * c)$,
- esistenza dell'elemento neutro: $\exists e \in G : \forall a \in G \ a * e = e * a = a$
- esistenza dell'inverso: $\forall a \in G \ \exists \overline{a} \in G : a * \overline{a} = \overline{a} * a = e$

Un gruppo per cui vale anche il seguente assioma

- proprietà commutativa: $\forall a, b \in G \ a * b = b * a$

si dice *gruppo abeliano*

È stato utilizzato il simbolo $*$ per indicare l'operazione binaria sul gruppo per sottolineare il fatto che l'operazione può essere qualunque, ma nel seguito useremo, se non diversamente specificato, la notazione moltiplicativa, per cui l'operazione $*$ sarà chiamata prodotto e denotata con $\cdot$ (o con

il concatenamento dei due operandi), l'elemento neutro sarà chiamato unità del gruppo e a volte denotato con 1 e l'inverso di a sarà indicato con a^{-1}. Per i gruppi commutativi si usa spesso anche la notazione additiva per cui l'operazione sarà denotata con $+$, l'elemento neutro è chiamato zero ed indicato con 0 e l'inverso di a è indicato con $-a$ e chiamato opposto di a. Osserviamo che grazie alla proprietà associativa , può essere definito senza problemi di parentesizzazione il prodotto di più di due elementi di G, e quindi possono essere definite le potenze ad esponente intero di un elemento $a \in G$ ponendo:

- $a^n = a \cdot a \cdot ... \cdot a$
  n volte, se $n > 0$
- $a^n = e$, se $n = 0$
- $a^n = a^{-1} \cdot a^{-1} \cdot ... \cdot a^{-1}$
  $-\ n$ volte, se $n < 0$

la proprietà associativa e le proprietà definitrici degli elementi neutro ed inverso garantiscono la validità delle usuali proprietà delle potenze:

- $a^n \cdot a^m = a^{n+m}$
- $(a^n)^m = a^{nm}$.

Def 1.1.2. *(Sottogruppo)* Un sottoinsieme H di un gruppo $< G, \cdot >$ si dice *sottogruppo* di G se è a sua volta gruppo rispetto alla stessa operazione $\cdot$ definita su G.

Es 1.1.3.
1. *I sottoinsiemi* $\{e\}$ *e* G *sono sottogruppi di ogni gruppo* $< G, \cdot >$*, detti* sottogruppi banali.
2. *L'insieme* P *di tutti i numeri pari è un sottogruppo del gruppo* $< Z, + >$.

Teor 1.1.4. *(teorema di Lagrange).* Sia $< G, \cdot >$ un gruppo finito di ordine n. Un sottogruppo H di G ha ordine che divide n.

In generale non vale l'inverso del teorema di Lagrange, tuttavia sussiste il seguente

Teor 1.1.5. Sia $< G, \cdot >$ un gruppo abeliano finito di ordine n. Per ogni divisore m di n esiste almeno un sottogruppo di G di ordine m.

1.2 Gruppi Ciclici

Def 1.2.1. *(Gruppo Ciclico).* Sia $< G, \cdot >$ un gruppo e $g \in G$ un qualsiasi elemento di G. L'insieme di tutte le potenze di g è un sottogruppo di G detto *sottogruppo ciclico generato da g* ed indicato in seguito con $< g >$.
Un gruppo $< G, \cdot >$ si dice *ciclico* se esiste un $g \in G$ tale che $G =< g >$. L'elemento g si chiama *generatore* di G

Ovviamente ogni gruppo ciclico è abeliano (segue immediatamente dalle proprietà delle potenze).

Es 1.2.2.
1. $< Z, + >$ è un gruppo ciclico ed un suo generatore è 1. Notate che anche -1 genera $< Z, + >$
Altri gruppi ciclici di interesse sono:
$(G, +) = (Z_n, +) =< 1 >= \{0, 1, 2, 3, \ldots, n-1\}$, con n intero; che identifica il gruppo delle classi di resto modulo n, assumendo come rappresentante di ciascuna classe l'elemento numericamente positivo più piccolo. Infine, $(G, \cdot) = (Z_p^*, \cdot) = \{1, 2, 3, \ldots, p-1\}$, con p numero primo, si può dimostrare essere un gruppo ciclico (rimandiamo alla bibliografia o ai testi/corsi di algebra per la dimostrazione [1]).

Teor 1.2.3. Ogni sottogruppo di un gruppo ciclico è ciclico.

Def 1.2.4. *(periodo di un elemento).* Considerati un gruppo $< G, \cdot >$ e un suo elemento g; si dice *periodo (o ordine)* di g, e si indica con $|g|$, il minimo intero positivo m, se esiste, tale che $g^m = e$; in tal caso si dice che l'elemento g è periodico o che ha periodo finito. Se tale m non esiste, cioè se non esiste alcun intero (positivo) n tale che $g^n = e$, si dice che g ha periodo 0 (o infinito)

Sia $< G, \cdot >$ un gruppo e sia g un suo generico elemento:

- Se $|g| = m$, $< g >= \{g,\ g^2, ...,\ g^{m-1},\ g^m = e\}$
- Sia $|G| = n$. G è ciclico sse esiste $g \in G$ tale che $|g| = n$.
- Se $|G| = n$, $\forall g \in G$ $|g|$ divide n. Infatti $|g| = | < g > |$ e per il Teorema di Lagrange $| < g > |$ divide $|G|$.
- Se $|G| = p$ con p primo, allora G è ciclico.
- Se $\exists t > 0 : g^t = e$, $|g|$ divide t. Infatti g ha ordine finito e posto $m = |g|$ si ha (dividendo t per m) $t = qm + r$, dove $0 \leq r < m$; allora $g^r = g^{t-qm} = g^t \cdot (g^m)^{-q} = e$, da cui $r = 0$.
- In un gruppo ciclico $G =< g >$ di ordine $n = |g|$, dato $h \in \mathbb{N}$ si ha:

$$|g^h| = \frac{n}{MCD(n, h)}$$

Dim. Posto $r = |g^h|$, si ha che $g^{hr} = (g^h)^r = e \Rightarrow n|hr$; questo vuol dire che esiste un intero m tale che $rh = mn$; dividendo ambo i membri per $MCD(n,h)$ si ottiene:

$$r\,\frac{h}{MCD(n,h)} = m\,\frac{n}{MCD(n,h)}$$

$\frac{h}{MCD(n,h)}$ e $\frac{n}{MCD(n,h)}$ sono co-primi per costruzione, quindi si può concludere:

$$\frac{n}{MCD(n,h)} \mid r$$

Viceversa si può osservare che

$$(g^h)^{\frac{n}{MCD(n,h)}} = (g^n)^{\frac{h}{MCD(n,h)}} = e \Rightarrow r \mid \frac{n}{MCD(n,h)}$$

da cui la tesi.

□

- Se $G =< g >$ e $|G| = n$, tutti gli elementi g^h con h primo con n generano G.

Prop 1.2.5. Un gruppo ciclico G di ordine n ha uno ed un solo sottogruppo di ordine m per ogni divisore m di n

Es 1.2.6. Consideriamo il gruppo delle classi di resto modulo p, con p numero primo, avente come operazione interna la moltiplicazione.
(N.B.: Lo zero è escluso dall'insieme perché non ammette inverso).
$(G,\cdot) = (Z_p,\cdot)$=$\{1,2,3,\ldots,p-1\}$, $|G| = p-1$.
$p = 7$, $(Z_7^*,\cdot) = \{1,2,3,4,5,6\}$; $n = |Z_7^*| = 6 = 2\cdot 3$;
$g = 3$ è un generatore per Z_7^* (Infatti, $Z_7^* = \langle 3\rangle = \{3^0 \equiv 1,\ 3^1 \equiv 3,\ 3^2 \equiv 2,\ 3^3 \equiv 6,\ 3^4 \equiv 4,\ 3^5 \equiv 5\}$. Essendo Z_7^* un gruppo ciclico finito, il teorema di Lagrange si inverte e sapendo che $n = 6 = 2\cdot 3$, sappiamo che esiste un sottogruppo di ordine 2 e un sottogruppo di ordine 3. Ci limitiamo a verificarlo senza dare un criterio costruttivo per esibire i generatori di questi sottogruppi. $H_1 = \{1,2,4\}$, $H_2 = \{1,6\}$.

(N.B.: il modulo e la cardinalità del gruppo sono due quantità distinte: le operazioni sono effettuate modulo 7, mentre la cardinalità di Z_7^* è $n = 6$.)

1.3 Algoritmo Esteso di Euclide

Senza pretesa di rigore o completezza in senso matematico, ci limitiamo a ricordare che: dati $a, b \in Z$ se $d \in Z$ è il loro massimo comun divisore (MCD), allora d divide anche una loro qualunque combinazione lineare ($d \mid$

$\xi \cdot a + \eta \cdot b$ con $\xi, \eta \in Z$). Si può dimostrare, assieme all'esistenza di d anche l'esistenza di almeno una coppia di elementi $\xi_a, \eta_b \in Z$ tali che

$$d = \xi_a a + \eta_b b$$

Il precedente risultato permetterà di calcolare l'inverso moltiplicativo in un campo finito.

Lemma 1.3.1. *Dati $a, b \in Z$; $a > b$; $q = \lfloor a/b \rfloor$; $r = a \bmod b = a - qb \in \{0, 1, 2 \ldots, b-1\}$, vale il seguente:*

$$MCD(a, b) = MCD(b, a \bmod b)$$

Usando il lemma precedente si possono scrivere le seguenti uguaglianze:

$a > b > 0$	$d = MCD(a, b)$
$r_0 = a$	
$r_1 = b$	$d = MCD(r_0, r_1)$
$r_2 = r_0 \bmod r_1 = r_0 - \lfloor r_0/r_1 \rfloor r_1;\ \ 0 \leq r_2 < r_1$	$d = MCD(r_1, r_2)$
$r_3 = r_1 \bmod r_2 = r_1 - \lfloor r_1/r_2 \rfloor r_2;\ \ 0 \leq r_3 < r_2$	$d = MCD(r_2, r_3)$
$r_4 = r_2 \bmod r_3 = r_2 - \lfloor r_2/r_3 \rfloor r_3;\ \ 0 \leq r_4 < r_3$	$d = MCD(r_3, r_4)$
$\ldots$	$\ldots$
$r_n = r_{n-2} \bmod r_{n-1} = r_{n-2} - \lfloor r_{n-1}/r_{n-2} \rfloor r_{n-2};\ \ r_n = 0$	$d = MCD(r_{n-1}, 0)$

per un certo intero n si ha:

$$d = MCD(a, b) = r_{n-1}$$

Pensiamo ora di riscrivere tutti i passaggi precedenti come combinazioni lineari di a, b:

$$
\begin{aligned}
&a > b > 0 \\
&r_0 = 1 \cdot a + 0 \cdot b \\
&r_1 = 0 \cdot a + 1 \cdot b \\
&r_2 = r_0 \, mod \, r_1 = (1a + 0b) - \lfloor \tfrac{r_0}{r_1} \rfloor (0a + 1b) = (\xi_2 a + \eta_2 b);\ 0 \leq r_2 < r_1 \\
&r_3 = r_1 \, mod \, r_2 = (0a + 1b) - \lfloor \tfrac{r_1}{r_2} \rfloor (\xi_2 a + \eta_2 b) = (\xi_3 a + \eta_3 b);\ 0 \leq r_3 < r_2 \\
&r_4 = r_2 \, mod \, r_3 = (\xi_2 a + \eta_2 b) - \lfloor \tfrac{r_2}{r_3} \rfloor (\xi_2 a + \eta_2 b) = (\xi_4 a + \eta_4 b);\ 0 \leq r_4 < r_3 \\
&\dots \\
&r_{n-1} = r_{n-3} \, mod \, r_{n-2} = (\xi_{n-3} a + \eta_{n-3} b) - \lfloor \tfrac{r_{n-3}}{r_{n-2}} \rfloor (\xi_{n-2} a + \eta_{n-2} b) = \\
&\qquad = (\xi_{n-1} a + \eta_{n-1} b);\ 0 \leq r_{n-1} < r_{r-2} \\
&r_n = r_{n-2} \, mod \, r_{n-1} = (\xi_{n-2} a + \eta_{n-2} b) - \lfloor \tfrac{r_{n-3}}{r_{n-2}} \rfloor (\xi_{n-1} a + \eta_{n-1} b) = \\
&\quad = (\xi_n a + \eta_n b);\ r_n = 0
\end{aligned}
$$

per cui:

$$d = r_{n-1} = \xi_{n-1} a + \eta_{n-1} b;\ \ \xi, \eta \in Z$$

Formalizzando la precedente costruzione si ottiene l'algoritmo di Euclide esteso (Extended Euclidean Algorithm, EEA) per il calcolo del massimo comun divisore.

Algorithm 1.3.1: Algoritmo esteso di Euclide

Input: $a, b \in D$
Output: $d = \xi_a \cdot a + \eta_b \cdot b,\ \xi_a, \eta_b$

```
1  begin
2      u ← (a, 1, 0)
3      v ← (b, 0, 1)
4      repeat
5          w ← u − ⌊u[0]/v[0]⌋ · v
6          u ← v
7          v ← w
8      until (w[0] = 0)
9      d ← u[0], ξ_a ← u[1], η_b ← u[2]
10     return (d, ξ_a, η_b)
11 end
```

Esempio 1.3.1. *Sia $D =< Z, +, \cdot >$ il consueto insieme di numeri interi relativi*

$d = MCD(11, 5) = 11\xi + 5\eta$*;*

$\underline{u} \leftarrow (11, 1, 0)$*;*
$\underline{v} \leftarrow (5, 0, 1)$*;*

$q = \lfloor \frac{11}{5} \rfloor = 2$*,* $\underline{w} \leftarrow (11 - 2 \cdot 5, 1 - 0 \cdot 2, 0 - 1 \cdot 2) = (1, 1, -2)$*;*
$\underline{u} \leftarrow (5, 0, 1)$*;*
$\underline{v} \leftarrow (1, 1, -2)$*;*

$q = \lfloor \frac{5}{1} \rfloor = 5$*,* $\underline{w} \leftarrow (5 - 1 \cdot 5, 0 - 1 \cdot 5, 1 - (-2) \cdot 5) = (0, -5, 11)$*;*
$\underline{u} \leftarrow (1, 1, -2)$*;*
$\underline{v} \leftarrow (0, -5, 11)$*;*

$d = 1$*;* $\xi = 1$*;* $\eta = -2$*.*

infatti: $1 = 1 \cdot 11 + (-2) \cdot 5$*.*

1.4 Funzione di Eulero: $n \mapsto \varphi(n)$

Dato un intero n, consideriamo l'insieme:

$$\{x \in Z \ : \ 1 \leq x \leq n-1, \ MCD(x, n) = 1\}$$

la sua cardinalità prende il nome di *Funzione di Eulero di n (Totient Function)*:

$$\varphi(n) \ = \mid \{x \in Z \ : \ 1 \leq x \leq n-1, \ MCD(x, n) = 1\} \mid$$

SE conosco la fattorizzazione di n, cioé: $n = \prod_{i=0}^{s} p_i^{e_i}, \ e_i \geq 1$; allora

$$\varphi(n) = \prod_{i=0}^{s} p_i^{e_i} - p_i^{e_i - 1} = n \cdot \prod_{i=0}^{s} (1 - \frac{1}{p_i})$$

La giustificazione della formula segue dalla immediata applicazione dei due seguenti lemmi, che riportiamo per comodità:

Lemma 1.4.1. *Siano $n, m \in \mathbb{N}^+$ con $n > m$, se $MCD(n, m) = 1$ allora $\varphi(n \cdot m) = \varphi(n) \cdot \varphi(m)$*

Dim. Omessa.

Lemma 1.4.2. *Siano p un numero primo e $k \in \mathbb{N}^+$, allora*

$$\varphi(p^k) = p^k - p^{k-1}$$

Dim. Cerchiamo tutti i numeri interi minori di p^k che abbiano un fattore in comune con p^k. Osserviamo che tali interi saranno tutti multipli di p, pertanto della forma $h \cdot p$ con $h \in \{1, \ldots, p^{k-1}\}$. Il numero di interi minori di p^k che hanno un fattore in comune con p^k è: p^{k-1}. Il numero di interi coprimi con p^k è dunque immediato: $\varphi(p^k) = p^k - p^{k-1}$.

1.5 Gruppi $(Z_n^*, \cdot)$

Per il crittosistema RSA è fondamentale avere chiare le proprietà dei gruppi $(Z_n^*, \cdot)$, che sono insiemi composti dai rappresentati delle classi di resto modulo n (n **qualsiasi**) che ammettono inverso moltiplicativo).
Per esempio si assuma $n = 15$, quindi in $(Z_{15}^*, \cdot)$ devono esserci gli elementi identificati con i rappresentanti delle classi di resto modulo 15 **che ammettono inverso moltiplicativo**.

Affinché un elemento (x mod n) abbia inverso moltiplicativo, deve accadere che $MCD(x, n) = 1$, infatti utilizzando l'algoritmo esteso di Euclide, si ha: $MCD(x, n) = 1 \ \Rightarrow \exists\ \xi, \eta \in Z : \ x \cdot \xi + n \cdot \eta = 1$, prendendo ambo i membri modulo n si ha: $x^{-1} \bmod n \equiv \ \xi \bmod n$.

Quindi tornando al nostro esempio,
$(Z_{15}^*, \cdot) = \{1, 2, 4, 7, 8, 11, 13, 14\}$,
$|Z_{15}^*| = \varphi(15) = \varphi(3 \cdot 5) = (3^1 - 3^0)(5^1 - 5^0) = 2 \cdot 4 = 8$.
Riportiamo per completezza che il gruppo $(Z_n^*, \cdot)$ è ciclico se e soltanto se $n = 2, 4, p^k, 2p^k$, con $k \geq 1$ dispari e p un numero primo.
(In particolare, il gruppo Z_p^* è ciclico con $p - 1$ elementi per ogni primo p. Più in generale, ogni sottogruppo finito del gruppo moltipicativo di un campo finito $GF(p^m)$ è ciclico [1].)

1.6 Calcolo degli inversi in un gruppo $(Z_n^*, \cdot)$

Il calcolo degli inversi in gruppi del tipo $(Z_n^*, \cdot)$, con n numero primo o composto, può essere computato in due modi distinti: *i*) tramite l'algoritmo esteso di Euclide, *ii*) utilizzando le nozioni di teoria dei gruppi.

i) Dato un $x \in (Z_n^*, \cdot)$, si ha che $MCD(x, n) = 1$ da cui, utilizzando l'algoritmo esteso di Euclide si trova che: $\exists\ \xi, \eta \in Z : \ x \cdot \xi + n \cdot \eta = 1$, quindi, prendendo ambo i membri modulo n si ha: $x^{-1} \bmod n \equiv \ \xi \bmod n$.

ii) Essendo $(Z_n^*, \cdot)$, $|Z_n^*| = \varphi(n)$, un gruppo finito di ordine $\varphi(n)$, ciascuno dei suoi elementi avrà periodo che divide $\varphi(n)$ e quindi vale:

$$x \in Z_n^*,\ x^{\varphi(n)} \equiv 1 \bmod n, \quad \text{(teorema di Eulero)}$$

da cui:

$$x \in Z_n^*,\ x^{\varphi(n)} \equiv 1 \bmod n \ \rightarrow \ x^{-1} \equiv x^{\varphi(n)-1} \bmod n$$

Capitolo 2

Crittosistemi a Chiave Publica

Daremo qui una breve e informale descrizione dei crittosistemi asimmetrici o crittosistemi a chiave publica, strumentale all'introduzione di alcune notazioni e terminologie utilizzate nelle parti restanti di questo documento, si rimanda comunque alla bigliografia e al materiale del corso per un'esposizione più completa ed esaustiva.

Prima della pubblicazione dell'articolo di Diffie ed Hellman [4], nel 1976, si assumeva che uno degli assiomi fondamentali della crittografia fosse l'assoluta segretezza del metodo impiegato, per non parlare delle chiavi di cifratura e di decifratura. Al contrario il lavoro di Diffie ed Hellman per la prima volta descriveva un sistema crittografico in cui non solo non è necessario che si mantenga tutto segreto, ma, al contrario, è indispensabile che una parte dell'informazione necessaria sia addirittura resa pubblica. L'idea dei crittosistemi a chiave pubblica (o asimmetrici) è la seguente: ciascun utente sceglie due funzioni crittografiche ($\mathbb{E}$nc, $\mathbb{D}$ec)che dipendono da alcuni parametri, ma rende noti solo quelli che permettono di codificare i messaggi a lui diretti (chiave pubblica, k_{pub}), mantenendo segreti quelli necessari alla decodifica (chiave privata, k_{priv}). In questo modo, chiunque può spedire un messaggio m ad un altro interlocutore, $c = \mathbb{E}\text{nc}_{k_{pub}}(m)$, senza che questo, se intercettato da terzi, possa essere compreso. Ogni utente gestisce soltanto 1 segreto (la propria chiave privata) ed è l'unico a poter invertire i testi cifrati a lui diretti, $m = \mathbb{D}\text{ec}_{k_{priv}}(c)$.

Nella pratica, gli algoritmi di calcolo delle funzioni crittografiche a chiave pubblica sono più di due ordini di grandezza meno performanti rispetto ai cifrari a blocchi o a *stream*, quindi in molti protocolli applicativi viene realizzata una soluzione ibrida per la quale, due interlocutori che devono scambiarsi informazioni attraverso un canale insicuro, utilizzano la critttografia a chiave pubblica per comunicarsi reciprocamente un segreto, dopodiché

mettendo assieme i due segreti, entrambi sono in grado di computare un parametro condiviso da utilizzare come chiave di un cifrario simmetrico che verrà utilizzato per trasmettere le informazioni.

Vediamo ora come un crittosistema asimmetrico possa essere realizzato: si prenda una funzione biiettiva $f : S_1 \mapsto S_2$ che sia facile da calcolare, ma di cui sia computazionalmente intrattabile il calcolo della funzione inversa (funzioni *One-way*) Naturalmente, posto esattamente in questi termini, il problema di calcolare l'inversa di f è intrattabile anche per il legittimo destinatario del messaggio: il tipo di funzioni che interessano è quello per cui è sì intrattabile il calcolo di f^{-1}, ma solo per coloro che non dispongano di una qualche informazione supplementare (*Trapdoor)* su f. Per questo motivo, nei crittosistemi a chiave pubblica, i ruoli della chiave di cifratura e di quella di decifratura sono molto diversi fra loro: la chiave di decifratura contiene proprio l'informazione necessaria per il calcolo efficiente di f^{-1}.

Schematicamente, un crittosistema a chiave pubblica può essere definito come una 6-upla:

$$\langle\ \mathbb{A},\ \mathbb{M},\ \mathbb{C},\ \mathbb{K},\ \{\,\mathbb{E}\mathrm{nc}_e,\ e \in \mathbb{K}\,\},\ \{\,\mathbb{D}\mathrm{ec}_d,\ d \in \mathbb{K}\,\}\ \rangle$$

dove: $\mathbb{A}$ è un insieme finito di simboli, ovvero l'alfabeto di definizione e tipicamente $\mathbb{A} = \{0, 1\}$; $\mathbb{M}$, $\mathbb{C}$ sono insiemi di stringhe dell'alfabeto che definiscono gli insiemi dei messaggi in chiaro (*plaintexts*) e dei messaggi cifrati (*ciphertexts*), rispettivamente; $\mathbb{K}$ è l'insieme delle **chiavi**, mentre le famiglie delle funzioni $\mathbb{E}\mathrm{nc} : \mathbb{K} \times \mathbb{M} \mapsto \mathbb{C}$ e $\mathbb{D}\mathrm{ec} : \mathbb{K} \times \mathbb{C} \mapsto \mathbb{M}$ parametriche in un valore dello spazio delle chiavi $e, d \in \mathbb{K}$, descrivono delle funzioni *one-way* con *trapdoor* che devono soddisfare gli ulteriori vincoli:

$$\mathbb{D}\mathrm{ec}_d(\mathbb{E}\mathrm{nc}_e(m)) = m\ \in \mathbb{M}$$

$$\mathbb{E}\mathrm{nc}_e(\mathbb{D}\mathrm{ec}_d(c)) = c\ \in \mathbb{C}$$

L'ideazione delle funzioni di cifratura e decifrazione, così come la costruzione delle chiavi e, d deve essere fatta in maniera congiunta e variano a seconda del problema computazionale alla base dell'intrattabilità computazionale nel calcolo degli inversi delle funzioni. Ogni funzione *one-way* con *trapdoor* è basata su un problema matematico per il quale non esistono algoritmi di calcolo dell'inversa con complessità computazionale polinomiale; quindi, pur di scegliere la dimensioni dello spazio delle chiavi $\mathbb{K}$, di $\mathbb{M}$ e di $\mathbb{C}$, *sufficientemente grandi* si ottiene un cifrario computazionalmente robusto.

Esempi di funzioni *one-way* sono: $m \mapsto m^e \bmod n,\ e, n, m \in \mathbb{N}^+$
$x \mapsto a^x \bmod \mathrm{p}, \mathrm{p}primo, a, x \in \mathbb{N}^+$
$k \mapsto E_k^{simm}(m)$, cifrario simmetrico in cui si conoscono m, c.

Capitolo 3

Crittosistema RSA

Certamente RSA (**R**ivest, **S**hamir e **A**dleman) [5] è il più popolare crittosistema a chiave pubblica. Utilizza una *One-way function with Trapdoor* basata sull'intrattabilità computazionale del problema della fattorizzazione di numeri interi.

Nel consueto scenario in cui due interlocutori A e B desiderano comunicare in maniera confidenziale, ciascun utente genera la sua coppia di chiavi pubblica/privata nel seguente modo:

A
$k_{pub,A} = \langle e_A, n_A \rangle$;
$k_{priv,A} = \langle d_A, p_A, q_A, \varphi(n_A) \rangle$

B
$k_{pub,B} = \langle e_B, n_B \rangle$;
$k_{priv,B} = \langle d_B, p_B, q_B, \varphi(n_B) \rangle$

dove

n: modulo.
p, q: numeri primi *grandi* ($\geq 2^{512}$).
e: esponente di cifratura, $e \in Z^*_{\varphi(n)}$ (oppure, in maniera equivalente $e \overset{random}{\leftarrow} \{1, \ldots, \varphi(n) - 1\}$ tale che $MCD(e, \varphi(n)) = 1$).
d: esponente di decifrazione, $d \in Z^*_{\varphi(n)}$, $d = e^{-1} \mod \varphi(n)$.

La primitiva di cifratura del crittosistema suddivide il plaintext M in blocchi da $\lceil \log_2 n \rceil$ bit, in modo da ottenere un messaggio $m \in Z_n$, $0 \leq m \leq n-1$.

Funzione di Cifratura (One-way with Trapdoor)
$c \leftarrow \mathbb{E}\text{nc}_{\langle e,n \rangle}(m)$;
$c \leftarrow m^e \mod n$.

Funzione di Decifrazione
$m \leftarrow \mathbb{D}\text{ec}_{\langle d,p,q,\varphi(n) \rangle}(c)$;
$m \leftarrow c^d \mod n$.

Vista la simmetria delle funzioni di cifratura e di decifrazione, basta dimostrare che:

$$\mathbb{D}\text{ec}(\mathbb{E}\text{nc}(m)) = m \in Z_n$$

$$(m^e)^d \bmod n \equiv m^{ed} \bmod n \stackrel{?}{\equiv} m \bmod n$$

Dim. Occorre distinguere 2 casi:

i) Se $MCD(m,n)$=1, allora vale il teorema di Eulero per cui $m^{\varphi(n)} \equiv 1 \bmod n$ per cui:

$$m^{ed} \bmod n \equiv m^{ed \bmod \varphi(n)} \bmod n \equiv m^{1+t\varphi(n)} \text{ per un qualche } t \text{ intero}$$

$$m^{1+t\varphi(n)} \equiv m \cdot (m^{\varphi(n)})^t \bmod n \equiv m \bmod n$$

ii) Se $MCD(m,n) \neq 1$, allora $m = r \cdot p$ (ad esempio); $MCD(m,n) = p$.
Si consideri che $m^{\varphi(q)} \bmod q \equiv m^{q-1} \bmod q \equiv 1$, quindi

$$(m^{\varphi(n)})^t \bmod q \equiv (m^{q-1})^{t(p-1)} \bmod q \equiv (m^{\varphi(q)})^{t(p-1)} \bmod q \equiv 1 + s \cdot q$$

per un qualche $s \in Z$; da cui:

$$m^{ed} \bmod n \equiv m^{ed \bmod \varphi(n)} \bmod n \equiv m^{1+t\varphi(n)} \text{ per un qualche } t \text{ intero}$$

$$m \cdot (m^{\varphi(n)})^t \bmod n \equiv m \cdot (1 + s \cdot q) \bmod n \equiv$$

$$\equiv m + m \cdot s \cdot q \bmod n \equiv m + r \cdot s \cdot n \bmod \equiv m \bmod n$$

3.1 Algoritmi di Esponenziazione

In questa sezione descriveremo due varianti dell'algoritmo più conosciuto per effettuare l'operazione di esponenziazione in maniera efficiente. Il problema è formulato nel modo seguente: dati $a, n \in \mathbb{N}$, si vuole calcolare l'intero $c = a^n$ effettuando un numero di moltiplicazioni minore di n.
Indicando con t il numero di cifre binarie necessarie per rappresentare l'intero n, cioé:

$$t = \lceil \lg_2 n \rceil,\ n = \langle n_{t-1}, \ldots, n_1, n_0 \rangle \text{ con } n_i \in \{0,1\},\ i \in \{0,1,\ldots,t-1\}$$

possiamo scrivere:

$$c = a^n = a^{\sum_{j=0}^{t-1} n_j 2^j} = a^{n_{t-1}2^{t-1} + n_{t-2}2^{t-2} + \ldots + n_1 2^1 + n_0} \tag{3.1}$$

A seconda del modo in cui è possibile leggere l'ultimo membro della catena di uguaglianze appena scritte, si esibiscono diverse formalizzazioni dell'algoritmo noto come *Square and Multiply(S&M).*

3.1.1 Square and Multiply - Left to Right

Pensando di scandire l'esponente dell'eq. (3.1) da sinistra verso destra, vale la seguente uguaglianza:

$$c = a^n = ((\cdots((a^{n_{t-1}})^2 \cdot a^{n_{t-2}})^2 \cdots)^2 \cdot a^{n_1})^2 \cdot a^{n_0}$$

Esempio 3.1.1. *Si supponga di voler applicare il metodo precedente al calcolo di $c = 5^6$; allora $a = 5$, $t = 3$, $n = 6 = 110_2 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$, si può scrivere:*

$$c = 5^{110_2} = ((5^1)^2 \cdot 5^1)^2 \cdot 5^0 = (5^2 \cdot 5)^2 = 15625.$$

Il costo computazionale del metodo appena illustrato, espresso in numero di moltiplicazioni e quadrati necessari per portare a termine la computazione è uguale a: $\lceil \lg_2 n \rceil$ quadrati + $\frac{1}{2}\lceil \lg_2 n \rceil$ moltiplicazioni (caso medio).

3.1.2 Square and Multiply - Right to Left

Pensando di scandire l'esponente dell'eq. (3.1) da destra verso sinistra, vale la seguente uguaglianza:

$$c = a^n = (a^{2^0})^{n_0} \cdot (a^{2^1})^{n_1} \cdot (a^{2^2})^{n_2} \cdots (a^{2^{t-1}})^{n_{t-1}}$$

Esempio 3.1.2. *Si supponga di voler applicare il metodo precedente al calcolo di $c = 5^6$; allora $a = 5$, $t = 3$, $n = 6 = 110_2 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$, si può scrivere:*

$$c = 5^{110_2} = (5^{2^0})^0 \cdot (5^{2^1})^1 \cdot (5^{2^2})^1 = (5 \cdot 5^2 \cdot 5^4) = 15625.$$

Si osservi come i fattori tra parentesi possano essere calcolati come quadrati del fattore precedente.

Algorithm 3.1.1: S&M Left to Right

Input: a, n, $t = \lceil \lg_2 n \rceil$, $n = \langle n_{t-1}, \ldots, n_1, n_0 \rangle, n \geq 0$
Output: $c = a^n$

```
1  begin
2      if n = 0 then
3          return 1
4      c ← a
5      for i ← t − 2 down-to 0 do
6          if n_i = 1 then
7              c ← c · a
8      return c
9  end
```

Il costo computazionale del metodo appena illustrato, espresso in numero di moltiplicazioni e quadrati necessari per portare a termine la computazione è uguale a: $\lceil \lg_2 n \rceil$ quadrati + $\frac{1}{2}\lceil \lg_2 n \rceil$ moltiplicazioni (caso medio).

Algorithm 3.1.2: S&M Right to Left

Input: a, n, $t = \lceil \lg_2 n \rceil$, $n = (n_{t-1}, \ldots, n_1, n_0), n \geq 0$
Output: $c = a^n$

```
1  begin
2      if n = 0 then
3          return 1
4      b ← a
5      if n_0 = 1 then
6          c ← a
7      else
8          c ← 1
9      for i ← 1 to t − 1 do
10         b ← b^2
11         if n_i = 1 then
12             c ← c · b
13     return c
14 end
```

3.2 Accelerazione funzioni di cifratura/decifrazione

Il calcolo della funzione di cifratura RSA, alla luce di quanto visto nella sezione precedente, è evidentemente accelerato se il peso di Hamming dell'esponente di cifratura, e, è minimo (perché si riduce il numero di moltiplicazioni). I valori dell'esponente di cifratura usualmente adottati in una chiave pubblica RSA sono pertanto:

$$e = \{\, 3,\ 17 = 2^4 + 1,\ 65537 = 2^{16} + 1 \,\}$$

Tutti questi valori (primi) rispettano certamente la condizione per cui $MCD(e, \varphi(n))$=1, qualsiasi sia il valore di n.

L'accelerazione del calcolo della funzione di decifrazione non può usufruire di un accorgimento analogo a quello della funzione di cifratura perché al variare del modulo n, tenendo e fisso, il peso di Hamming del parametro $d = e^{-1} \bmod \varphi(n)$ non è predicibile. Si ricorre invece all'uso di un algoritmo alternativo basato sul Teorema Cinese dei Resti (CRT, Chinese Remainder Theorem) che consente di ottenere uno *speedup* di un fattore 4 nel computo della funzione.

$$k_{priv} = \langle d, p, q, \varphi(n) \rangle$$

$$m \leftarrow \mathbb{D}\mathrm{ec}_{\langle d,p,q,\varphi(n)\rangle}(c) = c^d \bmod n;$$

$$m = c^d \bmod n \leftrightarrow \begin{cases} m_p = c^{d \bmod \varphi(p)} \bmod p \\ m_q = c^{d \bmod \varphi(q)} \bmod q \end{cases} \overset{CRT}{\rightarrow}$$

$$\overset{CRT}{\rightarrow} m \equiv \Big(q(q^{-1} \bmod p) m_p + p(p^{-1} \bmod q) m_q \Big) \bmod n$$

Tutti i calcoli vengono svolti con precisione (lunghezza degli operandi) dimezzata, pertanto anche se il numero complessivo di operazioni modulari non è cambiato, essendo la complessità computazionale di un'operazione di moltiplicazione modulare (tecnica di Montgomery) a complessità quadratica, lo *speedup* complessivo delle implementazioni è pari a 4.

3.3 Teorema Cinese del Resto (CRT)

Dati un numero X e un intero n di cui è nota la seguente scomposizione:

$$n = \prod_{i=1}^{k} n_i = n_1 \cdot n_2 \cdots n_k,\ MCD(n_i, n_j) = 1,\ \forall\, i, j \in \{1, 2, \ldots, k\},\ i \neq j$$

la seguente corrispondenza

$$X \Leftrightarrow \langle\ x_1,\ x_2\ \ldots,\ x_k\ \rangle, \text{ con } x_i \equiv X \bmod n_i;$$

$$(0 < x_i < n_i),\ \forall\, i \in \{1, 2, \ldots, k\}$$

è biunivoca.

Dim. Dati un numero X e la t-upla di interi $\langle n_1, n_2, \ldots, n_k\rangle$ tale che $MCD(n_i, n_j) = 1\ \forall\, i \neq j$, dimostrare che esiste un'unica t-upla $\langle x_1, \ldots, x_k\rangle$ con $x_i \equiv X \bmod n_i; 0 < x_i < n_i$ è immediato:

$$X \to \langle X \bmod n_1, \ldots, X \bmod n_k\rangle$$

Dimostriamo ora, in modo costruttivo, che data la t-upla $\langle x_1, \ldots, x_k\rangle$ con $x_i \equiv X \bmod n_i; 0 < x_i < n_i$ dove $MCD(n_i, n_j) = 1\ \forall\, i \neq j$, a questa è possibile associare in maniera univoca un intero $X \bmod n$, con $n = \prod_{i=0}^{k} n_i$ cioé: $\langle x_1, \ldots, x_k\rangle \to X$. Definiamo i seguenti: $M_i = n/n_i$; $M_i' = M_i^{-1} \bmod n_i$. Per costruzione $MCD(M_i, n_i) = 1$ e dunque esiste l'inverso mod n_i per il teorema di Eulero.

Osserviamo poi che:

$$M_i \cdot M_i^{'} \equiv 1 \bmod n_i\, \forall i \in \{1, \ldots, k\}$$

$$M_j \cdot M_j^{'} \equiv 1 \bmod n_i\, \forall i \neq j \in \{1, \ldots, k\}$$

Si verifica dunque facilmente come il numero intero X, $0 < X < n$:

$$X = \sum_{i=1}^{k} \left(M_i \cdot M_i^{'} \cdot x_i\right) \bmod n$$

è il più piccolo numero intero a cui è associata la t-upla $\langle x_1, x_2, \ldots, x_k\rangle$, $x_i = X \bmod n_i\ \forall i \in \{1, \ldots, k\}$ (N.B.: l'espressione tra parentesi è da intendersi come somma di prodotti tra numeri interi).

Un modo equivalente di enunciare il Teorema Cinese del resto è il seguente: Date le t-uple di interi $\langle x_1, \ldots, x_k\rangle$ e $\langle n_1, \ldots, n_k\rangle$, con $MCD(n_i, n_j) = 1\ \forall i \neq j \in \{1, \ldots, k\}$; il sistema di congruenze:

$$\begin{cases} X \equiv x_1 \bmod n_1 \\ X \equiv x_2 \bmod n_2 \\ \ldots \\ X \equiv x_k \bmod n_k \end{cases}$$

ammette sempre un'unica soluzione $X = \sum_{i=1}^{k} \left(M_i \cdot M_i^{'} \cdot x_i\right) \bmod n$, con $n = \prod_{i=1}^{k} n_i$, $M_i = N/n_i$, $M_i^{'} = M_i^{-1} \bmod n_i$.

3.4 Attacchi al Crittosistema

Forza Bruta: dato $c = m^e \bmod n$, provare tutte le possibili chiavi $d \in \{1, \ldots, \varphi(n)\}$ per trovare quella tale che $c^d \equiv m \bmod n$; ma... $\varphi(n) \approx n \geq 2^{1024}$.

trovare $\varphi(n)$: Noto $\varphi(n) = (p-1)(q-1)$ è immediato calcolare $d = e^{-1} \bmod \varphi(n)$ con l'algoritmo di Euclide.
Calcolare $\varphi(n)$ senza conoscere la scomposizione in fattori primi $n = p \cdot q$, è un problema difficile quanto la fattorizzazione del modulo.

$$Calc(\varphi(n)) \leq_P Fatt(n)$$

infatti se $n = \prod_{i=1}^{s} p_i^{e_i},\ e_i \geq 1$, allora $\varphi(n) = \prod_{i=1}^{s} \left(p_i^{e_i} - p_i^{e_i - 1}\right)$.

$$Fatt(n) \leq_P Calc(\varphi(n))$$

Infatti, nota $\varphi(n)$ si ha: $\varphi(n) = n - (p+q) + 1$, da cui con una semplice equazione di secondo grado $Z^2 + (\varphi(n) - n - 1)\, Z + n = 0$ si scopre la scomposizione in fattori primi del modulo.

Chosen Ciphertext Attack: Se un attaccante ha la possibilità di ottenere il decifrato di un messaggio scelto da lui, la sequenza di azioni seguente permette di violare la sicurezza del crittosistema:

1. Si copia dal canale di comunicazione il testo cifrato del messaggio m scelto come obiettivo: $c = m^e \bmod n$;
2. Si chiede la decifrazione del seguente messaggio $c \cdot x^e \bmod n \neq m$ con x scelto casualmente;
3. Si ottiene $m' = (c \cdot x^e)^d \bmod n$ e si calcola $m = m' \cdot x^{-1} \bmod n$.

Common Modulus Attack: Se le chiavi pubbliche di più di 1 utente fossero generate mantenendo lo stesso valore del modulo n, per esempio, $k_{pub,1} = (e_1, n)$, $k_{pub,2} = (e_2, n)$; nel caso in cui lo stesso messaggio fosse inviato in multicast a entrambi gli utenti $c_1 \equiv m^{e_1} \bmod n$, $c_2 \equiv m^{e_2} \bmod n$, si risalirebbe facilmente al contenuto del messaggio m, calcolando $MCD(e_1, e_2) = 1 = e_1 \cdot \xi + e_2 \cdot \eta = 1,\ \xi, \eta \in Z$, con l'algoritmo esteso di Euclide, ottenendo $m \equiv c_1^{\xi} \cdot c_2^{\eta} \bmod n$.

Low Exponent Attack: anche nello scenario in cui si voglia inviare lo stesso messaggio in multicast a ≥ 3 utenti aventi chiavi pubbliche ciascuno con un modulo differente; si può facilmente risalire al messaggio in chiaro, applicando il teorema cinese del resto. Per esempio, avendo a disposizione $c_1 = m^3 \bmod n_1$, $c_2 = m^3 \bmod n_2$, $c_3 = m^3 \bmod n_3$, $n_1 \neq n_2 \neq n_3$, applicando il CRT si ricava immediatamente m.
Per ovviare a questo attacco, le chiavi pubbliche RSA sono impostate sempre con un valore $e = 2^{16} + 1$, e ogni messaggio viene *salato* accodandogli una porzione di plaintext casuale di lunghezza fissa.

Algoritmi di Fattorizzazione:

Quadratic Sieve (QS)

$$\mathcal{O}\left(exp((1+o(1))\sqrt{ln\,n\ ln\,ln\,n})\right)$$

Number Filed Sieve:

$$\mathcal{O}\left(L_n(\frac{1}{3}, 1.92)\right), \quad L_n(v,c) = exp\left(c(ln\,n)^v(ln\,ln\,n)^{1-v})\right)$$

Quindi utilizzando un modulo sufficientemente *grande* (oggi $\geq 2^{1024}$) la sicurezza del crittosistema non è minacciata da questi algoritmi.

3.5 Esempio Completo di Calcolo

Supponendo di avere un crittosistema RSA con $k_{pub} = (n, e) = (143, 77)$, sapendo che $n = p \cdot q = 13 \cdot 11$:

(1) Determinare la chiave privata corrispondente, $k_{priv} = \langle p, q, d, \varphi(n)\rangle$.

(2) Simulare la funzione di cifratura del crittosistema applicandola al messaggio $m = 101 \in Z_n$.

(3) Simulare la funzione di decifratura del crittosistema **con** e **senza** l'applicazione del CRT.

Calcolo dell'esponente di decifrazione:
$\varphi(n) = (p-1)(q-1) = 120 = 2^3 \cdot 3 \cdot 5$
$d = e^{-1} \bmod \varphi(n) = e^{\varphi(\varphi(n))-1} \bmod \varphi(n) = 77^{\varphi(120)-1} \bmod \varphi(n) =$
$= 77^{31} \bmod \varphi(n)$.
Applicando lo S&M left-to-right:
$d \equiv_{120} 77^{31} \equiv_{120} 77^{11111_2} \equiv_{120} ((((77^2)\cdot 77)^2 \cdot 77)^2 \cdot 77)^2 \cdot 77 \equiv_{120} \cdots \equiv_{120} 53$.
(Alternativamente si poteva applicare l'algoritmo esteso di Euclide).

Cifratura:
$m = 101 \in Z_n$, $n = 143$, $e = 77 = 1001101_2$, applicando ancora lo S&M si ha: $c = m^e \bmod n \equiv_{143} 101^{1001101_2} = ((((101^2)^2)^2 \cdot 101)^2 \cdot 101)^2)^2 \cdot 101 \equiv_{143}$ $\cdots \equiv_{143} 95$.

Decifratura senza CRT:
$m = c^d \bmod n \equiv_{143} 95^{53} \equiv_{143} 95^{11010_2} \equiv_{143} ((((95^2 \cdot 95)^2)^2 \cdot 95)^2)^2 \cdot 95 \equiv_{143}$ $\cdots \equiv_{143} 101$.

Decifratura con CRT:
$n = p \cdot q = 13 \cdot 11 = 143; \varphi(n) = 120; e \equiv_{120} 77, d \equiv_{120} 53$, $c \equiv_n m^e \equiv_{143} 95$.

$$m = c^d \bmod n \equiv_n c^{d \bmod \varphi(n)} \leftrightarrow \begin{cases} m_p = c^{d \bmod \varphi(p)} \bmod p \\ m_q = c^{d \bmod \varphi(q)} \bmod q \end{cases}$$

$m \equiv_n \left(M_p \cdot M_p^{'} \cdot m_p + M_q \cdot M_q^{'} \cdot m_q\right) \bmod n$,
con $M_p = q = 11$, $M_p^{'} = q^{-1} \bmod p \equiv_{13} 6$,
$M_q = p = 13$, $M_q^{'} = p^{-1} \bmod q \equiv_{11} 6$.
Calcoliamo: $d \bmod \varphi(p) \equiv_{12} 5$, $d \bmod \varphi(q) \equiv_{10} 3$, quindi:

$$\begin{cases} m_p \equiv_p c^{d \bmod \varphi(p)} \\ m_q \equiv_q c^{d \bmod \varphi(q)} \end{cases} \rightarrow \begin{cases} m_p \equiv_{13} 95^{5 \bmod 12} \equiv_{13} 10 \\ m_q \equiv_{11} 95^{3 \bmod 10} \equiv_{11} 2 \end{cases} \rightarrow$$

$$\rightarrow m \equiv_{143} (11 \cdot 6 \cdot 10 + 13 \cdot 6 \cdot 2) \equiv_{143} 101$$

Capitolo 4

Crittosistemi basati sul Logaritmo Discreto

Il problema del logaritmo discreto generalizzato (Generalized Discrete Logarithm Problem, GDLP) è formulato nel modo seguente.

Assegnato un gruppo ciclico $(G, \cdot)$ con elemento generatore g tale che $G = \langle g \rangle$, per ogni $\alpha \in G$ trovare l'*unico* esponente positivo $0 \leq m \leq |G| - 1$ tale che $g^m = \alpha$ in G ($m = log_g^D \, \alpha \bmod |G|$).

In crittografia è importante selezionare gruppi ciclici G in cui il GDLP risulti essere un problema computazionalmente *difficile*.

Es 4.0.1. Dato $G = (Z_{19}, +)$, $|G| = 19$, $G = \{0, 1, 2, \ldots, 18\}$, $g = 2$ generatore (tutti gli elementi in realtà sono generatori perché G =19 è un numero primo). Se $x = 15$, calcolare $m = log_g^D \, x \bmod |G| = log_2^D \, 15 \bmod 19$ equivale a risolvere:

$$2 \cdot m \equiv 15 \bmod 19$$

che è polinomialmente risolvibile applicando l'algoritmo esteso di Euclide:

$$m \equiv 2^{-1} \cdot 15 \bmod 19 \equiv 17 \bmod 19$$

Non si utilizzeranno mai gruppi additivi $G = (Z_p, +)$, ma invece i gruppi moltiplicativi di un campo finito, nei quali il problema di estrazione del logaritmo discreto non ammette algoritmi risolutivi con complessità computazionale polinomiale.

Per semplicità di calcolo, vedremo esempi numerici con soli gruppi del tipo: $G = (Z_p^*, \cdot)$, p primo; che sono comunque i più utilizzati nella pratica.

Es 4.0.2. Dato $G = (Z_{11}^*, \cdot)$, $|G| = \varphi(11) = 10$; conosco la fattorizzazione di $|G| = 10 = 5 \cdot 2$, possiamo trovare il generatore del gruppo con il seguente algoritmo:

Input: $|G| = p_1^{e_1} \cdot \dots \cdot p_s^{e_s},\ \forall e_i \geq 1$
Output: g, generatore di G

```
1 begin
2     while true do
3         g ←(random) {1, ..., |G| − 1}
4         if g^{|G|/p_1} ≠ 1 ∧ ... ∧ g^{|G|/p_s} ≠ 1 then
5             break
6     return g
7 end
```

$|G| = \{1, 2, \dots, 10\}$; $g \overset{random}{\leftarrow} \{1, \dots, 9\} = 2$;
$g^{10/2} \bmod 11 \equiv_{11} 2^5 \equiv_{11} 2 \neq 1$
$g^{10/5} \bmod 11 \equiv_{11} 2^2 \equiv_{11} 4 \neq 1$
quindi $g = 2$ è un generatore.

Crittosistemi a chiave pubblica basati sul GDLP si distinguono in tre sottocategorie. Dato un gruppo ciclico finito $(G, \cdot)$, $g \in G$ generatore.

(i) DLP, dato $h \in G$, trovare x tale che $h = g^x$

(ii) DHP (Diffie-Hellman Problem), dati $a = g^x$, $b = g^y$, trovare $c = g^{x\,y}$

(iii) DDH (Decision Diffie-Hellman), dati $a = g^x$, $b = g^y$, $c = g^z$, stabilire se $z \equiv x\,y \bmod |G|$

Risolto il DLP ci si può ricondurre in tempo polinomiale a risolvere DHP e DDH. Il viceversa non ammette alcun risultato generale ed è largamente accettato considerare tutti e tre i problemi come *computazionalmente difficili*.

4.1 Scambio di Chiavi di Diffie-Hellman

Il problema è quello di stabilire una segreto comune tra due interlocutori A, B, mediante scambio di messaggi su canale insicuro.

I parametri pubblici del protocollo constano nello stabilire un gruppo ciclico finito G, con generatore $g \in G$ e ordine $n = |G|$.

Entrambi gli interlocutori, per ogni istanza del protocollo definiscono:
$k_{priv,A} = a \overset{random}{\leftarrow} \{1, \ldots, n-1\}$, $k_{pub,A} = g^a$

$k_{priv,B} = b \overset{random}{\leftarrow} \{1, \ldots, n-1\}$, $k_{pub,B} = g^b$

A invia a B il suo token pubblico (o chiave pubblica effimera – di sessione);
B invia a A il suo token pubblico (o chiave pubblica effimera – di sessione)

$$A \text{ calcola: } k_{BA}^{sessione} = (k_{pub,B})^{k_{priv,A}} = g^{b\,a}$$
$$B \text{ calcola: } k_{AB}^{sessione} = (k_{pub,A})^{k_{priv,B}} = g^{a\,b}$$

La verifica di correttezza è immediata, essendo valide le proprietà delle potenze in un gruppo.
N.B.: Nel protocollo di Diffie-Hellman, le chiavi pubbliche scambiate sono elementi di G, mentre le chiavi private (gli esponenti) sono sempre elementi in Z_n, $n = |G|$.

Il protocollo è sicuro rispetto ad attaccanti *passivi*, che si limitano ad auscultare tutto quello che viene trasmesso sul canale, grazie alla difficoltà del DLP. Il protocollo non essendo autenticato è tuttavia soggetto ad attacchi del tipo *Man-In-The-Middle* (MITM), in cui un avversario attivo può riuscire a mascherarsi verso ciascuno dei due utenti come se fosse la controparte.

La scelta dei parametri pubblici per il protocollo di Diffie-Hellman, nella quasi totalità delle implementazioni esistenti, viene operata nel modo seguente:

Si considera $(Z_p^*, \cdot)$, $p = 2 \cdot p_1 \cdot p_2 + 1$ primo, con p_1, p_2 primi tali che $p \geq 2^{1024}$ ($p_2 = 1$, spesso), quindi $m = |Z_p^*| = p - 1 = 2 \cdot p_1 \cdot p_2$, con generatore α.

Si assume quindi $(G, \cdot)$ come il sottogruppo di $(Z_p^*, \cdot)$ avente cardinalità p_1, con p_1 scelto a priori sufficientemente grande ($p_1 \geq 2^{1024}$), generatore $g = \alpha^{m/p_1}$, e ordine $n = |G| = p_1 \geq 2^{160}$.

Es 4.1.1.
$p = 1 + 2p_1p_2$, $p_1 = 3$, $p_2 = 5$;
$p = 31$ è primo!
$(Z_p^*, \cdot)$; $m = |Z_p^*| = p - 1 = 30 = p_1p_2$, generatore $\alpha = 3$ (verificare!)

Assumiamo come gruppo $(G, \cdot)$ il sottogruppo di Z_p^* avente cardinalità $n = |G| = p_1 = 5$, quindi $g = \alpha^{m/n} \bmod n = 3^{30/5} \bmod 31 \equiv 16 \bmod 31$.

Quindi se:
$k_{priv,A} = 2$ $k_{pub,A} = g^a = 16^2 \bmod 31 \equiv_{31} 8$
$k_{priv,B} = 4$ $k_{pub,A} = g^a = 16^4 \bmod 31 \equiv_{31} 2$

allora,
$k_{BA} = 2^2 \bmod 31 = k_{AB} = 8^4 \bmod 31 = 4 \bmod 31$.

4.2 Crittosistema di ElGamal [6]

Non adottato rispetto ad RSA, perché produce un testo cifrato lungo il doppio rispetto al testo in chiaro. Tuttavia, l'efficienza computazionale e il livello di sicurezza (a parità di prestazioni) sono equivalenti a quelli dell'RSA.

I parametri pubblici del crittosistema constano nello stabilire un gruppo ciclico finito G, con generatore $g \in G$ e ordine $n = |G|$ (preferibilmente primo, con un meccanismo di scelta identico a quanto visto per il protocollo di Diffie-Hellman) in cui il GDLP sia *computazionalmente difficile*.

Le chiavi pubblica/privata di ciascun utente sono definite come segue:

$$k_{pub} = \langle\ n,\ g,\ g^s\ \rangle \quad k_{priv} = \langle\ s \in Z_n\ \rangle$$

Dati due utenti, A, B, con chiavi rispettivamente:

$$k_{pub,A} = \langle\ n,\ g,\ g^{s_A}\ \rangle,\ k_{priv,A} = \langle\ s_A \in Z_n\ \rangle$$

$$k_{pub,B} = \langle\ n,\ g,\ g^{s_B}\ \rangle,\ k_{priv,B} = \langle\ s_B \in Z_n\ \rangle$$

A riceve la chiave pubblica di B e per cifrare un messaggio $m \in G$, $c = \mathbb{E}\,\mathrm{nc}_{k_{pub,B}}(m)$, effettua le seguenti operazioni:

(1) $l \overset{random}{\leftarrow} \{1, \ldots, n-1\} \in Z_n$

(2) $\gamma \leftarrow g^l \ \in G$, $\delta \leftarrow m \cdot (g^{s_B})^l \ \in G$

(3) invio sul canale $c = \langle\ \gamma,\ \delta\ \rangle$

In fase di ricezione B, per recuperare il messaggio in chiaro, $m = \mathbb{D}\,\mathrm{ec}_{k_{priv,B}}(c)$ computa:

$$\gamma^{n-s_B} \cdot \delta \equiv g^{-l\,s_B} \cdot m \cdot g^{s_B\,l} \equiv m \ \in G$$

Lo schema di ElGamal, risulta diffuso nella pratica per implementazioni che assumono come gruppo G, l'insieme dei punti di una curva ellittica (vedi bibliografia); mentre lo schema di firma digitale di ElGamal è tutt'oggi standardizzato e conosciuto come DSA (Digital Signature Algorithm) e ECDSA (Elliptic Curve DSA).

Sicurezza del Crittosistema

Anche in questo caso non abbiamo introdotto schemi di firma digitale, per l'autenticazione dei messaggi, quindi ci limiteremo ad analizzare la sicurezza dello schema nei confronti di un avversario passivo che si limita a copiare ogni messaggio transitato sul canale.

L'avversario conosce $n, g, g^{s_B}, \gamma = g^l, \delta = m \cdot \left(g^{s_B\,l}\right)$, quindi le sue uniche possibilità rimangono quelle di:

(1) estrarre $s_B = log_g^D\, g^{s_B},\ m = \delta \cdot \gamma^{-s_B} \ \in G$, ma.... il DLP è difficile!

(2) se disponesse di un oracolo in grado di risolvere il DHP, allora $DHP(g^{s_B}, \gamma) = DHP(g^{s_B}, g^l) = g^{s_B\,l}$ da cui $m \equiv \delta \left(g^{s_B\,l}\right)^{-1} \ \in G$, ma... il DHP è difficile!

Bibliografia

[1] R.Lidl, H.Niederreiter, *Introduction to finite fields and their applications*, Cambridge University Press.

[2] Stinson D., *Cryptography Theory and Practice*, CRC Press (2002).

[3] Menezes A.J., Oorschot P.C., Vanstone S.A., - *Handbook of Applied Cryptography*, CRC Press (2001).
Disponibile su web http://www.cacr.math.uwaterloo.ca/hac/

[4] Diffie, Whitfield and Hellman, Martin E., *New Directions in Cryptography*, IEEE Transactions on Information Theory, No.6, pp.644–654, 1976.

[5] Rivest, R. L., Shamir, A., and Adleman, L. 1978. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21, 2 (Feb. 1978), 120–126. DOI= http://doi.acm.org/10.1145/359340.359342

[6] Taher El Gamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transactions on Information Theory, Vol.31, No.4, pp.469-472, 1985.