(HackIt'00 – 16/17/18 Giugno 2000 – Centro Sociale Forte Prenestino Roma)

Seminario

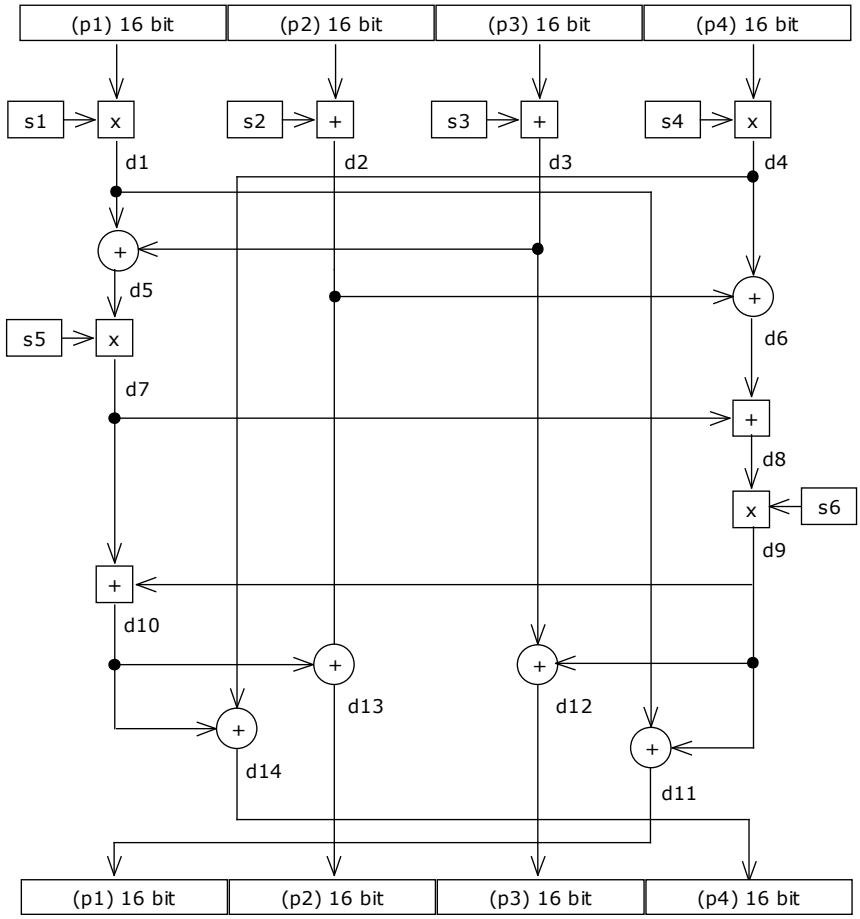
Introduzione agli aspetti matematici della crittografia asimmetrica del PGP e GNUPG

di Enrico Zimuel
(aka cerin0)
cerin0@olografix.org

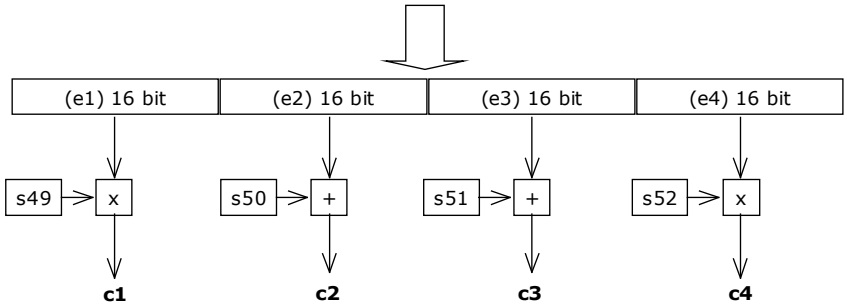
Sommario:

l'algoritmo IDEA, schema di funzionamento a blocchi del PGP aritmetiche circolari, le funzioni unidirezionali, il logaritmo finito, il teorema di Fermat-Eulero, generazione di numeri primi elevati, il simbolo di Jacobi, l'algoritmo RSA.

Algoritmo IDEA



la procedura viene iterata altre 7 volte considerando le altre 6 sub-key a partire da Sn dove n=7,8,9...52



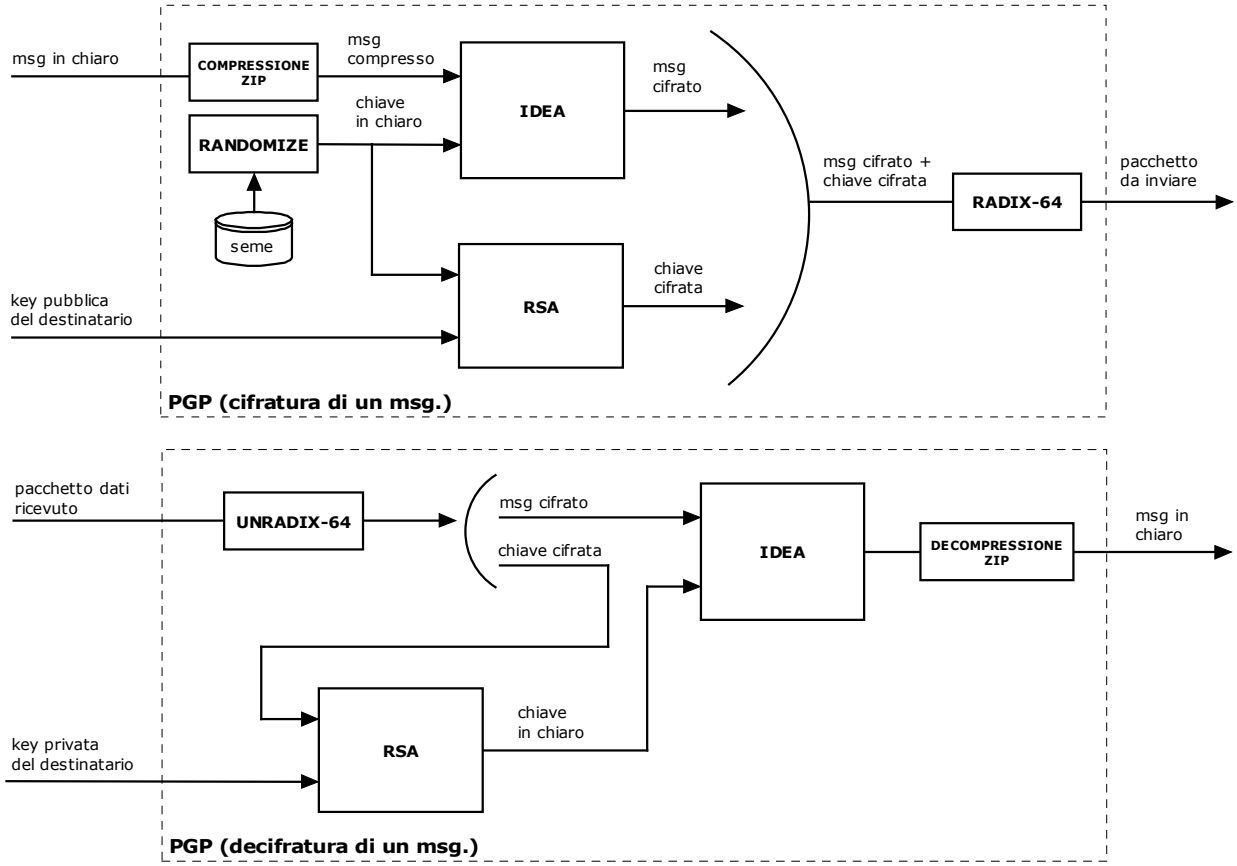
LEGENDA:

x **prodotto mod ((2^16)+1)**

\+ **somma mod (2^16)**

\+ **xor**

Principio di funzionamento del PGP

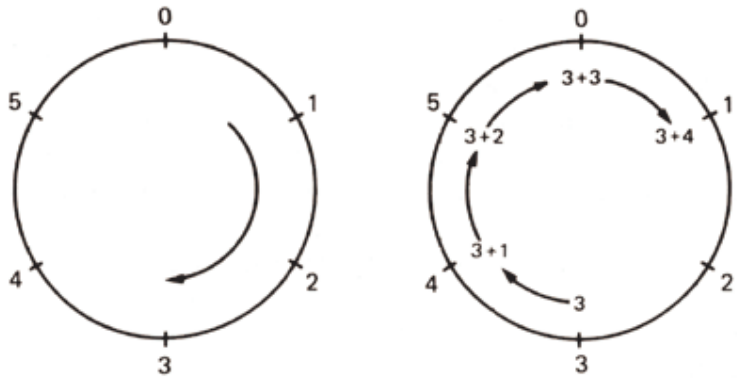


Aritmetiche circolari

Considerando un insieme numerico di m cifre è possibile costruire una particolare aritmetica chiamata **modulo m** (*o aritmetica circolare*).
Per m=6, l'operazione 3+3 da come risultato 0, di solito si usa scrivere:

$$(3 + 3) \bmod 6 = 0$$

In pratica avendo solo un numero limitato di m cifre le operazioni di somma e prodotto devono essere considerate in maniera "circolare":



A livello operativo la somma ed il prodotto modulo m vengono effettuati considerando il resto della divisione per m dell'operazione somma o prodotto eseguita in maniera classica. Considerando l'esempio precedente:

$$(3 + 3) \bmod 6 = 6 \bmod 6 = \text{resto di } 6/6 = 0$$

Lo stesso vale per il prodotto:

$$(3 * 2) \bmod 6 = 0$$
$$(4 * 2) \bmod 6 = 2$$
$$\ldots$$

L'insieme dei numeri di m cifre viene indicato in letteratura con il simbolo $\mathbf{Z_6}$, la coppia ($\mathbf{Z_6}$, +) costituisce *un gruppo finito commutativo*.

Considerando l'operazione prodotto modulo m si notato alcuni casi "particolari", ad esempio:

$$(4 * 3) \bmod 6 = 0$$

Esiste una coppia di numeri diversi da zero (nel nostro caso 4 e 3) che da come risultato zero!
Questo risultato invalida la proprietà algebrica dell'annullamento del prodotto: *un prodotto è nullo se e solo se lo è almeno uno dei due fattori.*
L'operazione di prodotto modulo m non può quindi essere utilizzata per costruire strutture algebriche (che abbiano quindi proprietà interessanti per un'applicazione crittografica) a meno che non si limiti la scelta dei moduli m a numeri primi.

Definizione
Un numero intero p>1 è **primo** se i soli divisori positivi di p sono p e 1.

Considerando quindi insiemi numerici modulo p (con p primo) ed operazioni di somma e prodotto si possono costruire strutture algebriche ricche di proprietà ed interessanti dal punto di vista crittografico.
In particolare la terna ($\mathbf{Z_p}$, +, *), con p primo, costituisce un campo finito o **campo di Galois**.

Le funzioni unidirezionali

Nella crittografia asimmetrica rivestono un'importanza fondamentale le funzioni unidirezionali.
Esse sono delle particolari funzioni matematiche nelle quali *il calcolo dell'inversa della funzione è di difficile attuazione*.

Consideriamo una funzione banale f(x)=2x essa avrà come funzione inversa la funzione $f^{-1}(x)=(1/2)x$, ad esempio:

$$f(3)= 2 * 3 = 6;\ f^{-1}(6)= (1/2)*6 = 3$$

In particolare $f^{-1}(f(x))=x$ e $f(f^{-1}(x))=x$. In questo caso il calcolo della funzione inversa è banale ed anche la sua attuazione pratica risulta di facile implementazione, basta moltiplicare x per 0.5.

Esistono delle funzioni particolari nelle quali il calcolo di $f^{-1}(x)$ è computazionalmente complesso e richiede quindi "molto" tempo.

Consideriamo ad esempio il campo di Galois Z_7 e consideriamo le potenze dei numeri mod 7. Ad esempio $3^0=1$, $3^1=3$, $3^2=2$, $3^3=6$, $3^4=4$, $3^5=5$, $3^6=1=3^0$. Il numero 3 con tutte le sue 6 potenze genera tutti gli elementi non nulli di Z_7, un tale numero viene chiamato in letteratura ***elemento primitivo***. Un elemento primitivo (b) su di un campo di Galois può essere sfruttato per costruire una funzione unidirezionale: *la funzione esponenziale* $f(x)=b^x$.
Se prendiamo Z_7 con b=3 elemento primitivo, il calcolo di f(x) per 0,1,2,3,4,5 può essere sviluppato facilmente, si ottengono i seguenti valori:

f(0)= 1
f(1)= 3
f(2)= 2
f(3)= 6
f(4)= 4
f(5)= 5

La funzione inversa risulta essere $f^{-1}(x)= \log_b(x)$, per calcolarla in termini pratici basta leggere l'elenco precedente al contrario (da destra a sinistra, ad esempio $f^{-1}(6)=3$). Nel caso di campi di Galois

piccoli la funzione $f^{-1}(x)= \log_b(x)$ è facilmente calcolabile, nel caso di numeri primi enormi, ad esempio dell'ordine di grandezza di 10^{50}, il calcolo del logaritmo diventa molto complesso dal punto di vista computazionale.
La situazione è fortemente asimmetrica: per il calcolo degli esponenti esistono algoritmi semplici, per il calcolo dei logaritmi non esistono algoritmi efficienti; l'algoritmo più veloce che si conosca per il calcolo del logaritmo finito richiede un numero di passi dell'ordine di:

$$e^{\sqrt{\ln(p)*\ln(\ln(p))}}$$

In pratica per $p\sim10^{50}$ il numero di passi risulta essere di circa $1.4*10^{10}$, ossia 14'000'000'000 passi, con un Pentium III a 1 Ghz (~3280 MIPS) basterebbero circa 4,5 secondi.
Proviamo a stimare per $p\sim10^{200}$ il numero di passi è dell'ordine di $1.2*10^{23}$, con lo stesso Pentium III occorrerebbero circa 3.6*10^13 secondi, ossia 10 miliardi di ore circa 1 milione di anni!

Per il tempo di calcolo dell'algoritmo relativo all'esponente la complessità è di circa ln(p)/ln(2) per cui per $p\sim10^{200}$ si hanno dei tempi di calcolo inferiori al secondo.

Di seguito è proposto l'algoritmo in C per il calcolo di b^x mod n.

```
long esponente(long b, long x, int n)
{
  long b1=b;
  long x1=x;
  long y=1;

  while (x1 != 0){
    while (x1 % 2 == 0){
      x1= x1 \ 2;
      b1= (b1*b1) % n;
    }
    x1-=1;
    y=(y*b1) % n;
  }

  return y;
}
```

Il teorema di Fermat-Eulero

Il teorema di Fermat-Eulero è uno dei teoremi di teoria dei numeri più importanti poiché lega le operazioni in modulo al concetto di numero primo.
Incontreremo questo teorema nel cifrario RSA ed in generale in tutti gli algoritmi asimmetrici basati sui numeri primi elevati.

Prima di introdurre il teorema è necessario definire la *funzione di Eulero* **$\varphi(n)$**, in questo modo:

$\varphi(1)=1$
$\varphi(n)$= numero degli interi minori di n e primi con n (per n>1)

Ad esempio $\varphi(4)=2$ poiché 1 e 3 sono primi con 4, $\varphi(5)=4$ poiché 1,2,3,4 sono primi con 5.
Se p è primo allora tutti i numeri che lo precedono saranno primi con p, per cui $\varphi(p)=p-1$.
In generale $\varphi(n)$ può essere determinato utilizzando la formula:

$$\varphi(n)=n * (1-1/n_1) * (1-1/n_2) * \dots * (1-1/n_m)$$

dove n_1, n_2, … n_m sono i fattori primi distinti che compaiono nella decomposizione di n.
Ad esempio:

$\varphi(6)= 6 * (1-1/2) * (1-1/3) = 2$ poiché $6=3*2$

$\varphi(63)=63 * (1-1/3) * (1-1/7)= 36$ poiché $63=3^2*7$

Teorema (Eulero-Fermat)

Se **n** è un intero positivo e **a** è primo con n, allora $\mathbf{a^{\varphi(n)}=1 \bmod n}$

Generazione di numeri primi elevati: l'algoritmo di Solovay e Strassen

Denifiamo il *simbolo di Jacobi* **J(a,b)** in questo modo:

- a e b sono due interi primi fra loro, a<b, b=2k+1 (k=0,1,2…)
- J(1,b)=1
- se a è pari $J(a,b)= J(a/2,b) * (-1)^{(b*b-1)/8}$
- se a è dispari $J(a,b)= J(b \bmod a,a) * (-1)^{(a-1)(b-1)/4}$

Da questa definizione si deduce che il simbolo di Jacobi J(a,b) è una funzione che ha solo due valori 1 e –1.
Calcoliamo ad esempio J(6,13):

$J(6,13)= J(3,13) * (-1)^{21}= -J(3,13) = -J(13 \bmod 3,3) * (-1)^{6}=$
$= -J(1,3)= -1$

Si può dimostrare che se b è un numero primo le uguaglianze

$$MCD(a,b)= 1 \text{ e } J(a,b)=a^{(b-1)/2} \bmod b$$

valgono per tutti i numeri a compresi fra 1 e b-1; se b non è primo si dimostra invece che esse cadono per almeno la metà degli a compresi fra 1 e b-1. Questo significa che se noi scegliamo *a caso* un numero a compreso fra 1 e b-1 la probabilità che le uguaglianze precedenti siano false è rigorosamente nulla se b è primo ed è almeno ½ se b non lo è. Invece di scegliere un unico numero a compreso fra 1 e b-1 ne scegliamo un quantitativo elevato (considerando che stiamo parlando di numeri elevati). Se le uguaglianze tornano per tutte le scelte casuali di a, ad esempio 100, la probabilità che b sia primo è dell'ordine di $(1-2^{-100})$ ossia circa 1, la certezza; se le scelte hanno esito negativo anche una sola volta vorrà dire che b è composto.

Con questo algoritmo ed un calcolatore veloce (come quelli in attualmente in commercio) si possono verificare con estrema facilità l'eventuale primalità di un numero primo b preassegnato dell'ordine di grandezza di un centinaio di cifre decimali.

Il cifrario RSA

RSA= R.Rivest, A.Shamir, L.Adleman

La funzione unidirezionale che sta alla base dell'algoritmo viene costruita sfruttando il fatto che è facile calcolare il prodotto di due numeri primi "molto grandi", ma dato solo il prodotto è oltremodo difficile risalire ai due fattori.

- Considero p,q due numeri primi molto elevati
- Calcolo il prodotto n= p * q
- La funzione di Eulero $\varphi(n)=(p-1)*(q-1)$
- Scelgo un intero x primo con $\varphi(n)$, calcolo l'inverso y mod $\varphi(n)$ (in pratica $xy= 1 \bmod \varphi(n)$)

La coppia **(y,n)** costituisce la chiave pubblica

Considero il messaggio da cifrare composto da numeri $m_i<n$, la funzione di cifratura è rappresentata da:

$$c= m_i^y \bmod n$$

Per decifrare il messaggio c è necessario conoscere la chiave segreta **(x,n)**:

$$m_i= c^x \bmod n$$

Dimostriamo quest'ultima relazione nel caso che né p né q dividano m_i e dunque m_i sia primo con n. Dal teorema di Fermat-Eulero si ha $m_i^{\varphi(n)}= 1 \bmod n$, dal momento che $xy= 1 \bmod \varphi(n)$ si ha che $xy=Q*\varphi(n)+1$, dove Q è il quoziente nella divisione xy per $\varphi(n)$, 1 è il resto. Dunque

$$m_i^{xy}= m_i^{Q\varphi(n)+1}= (m_i^{\varphi(n)})^Q m_i = (1)^Q\, m_i \bmod n$$

ossia $m_i^{xy}= m \bmod n$ e tenendo presente che il crittogramma è $c= m_i^y \bmod n$ si ha

$$m_i^{xy}= (m_i^y)^x= c^x = m_i \bmod n$$

come volevasi dimostrare.

Riferimenti bibliografici/links

Libri:

- **"Segreti Spie e Codici Cifrati" C.Giustozzi, A.Monti, E.Zimuel - edizioni Apogeo**
- **"Codici & Segreti" Simon Singh - edizioni Rizzoli**
- **"Crittografia" Andrea Sgarro - edizioni Franco Muzzio**
- **"Crittologia - come progettere le informazioni riservate" L.Berardi, A.Beutelspacher - edizioni FrancoAngeli**
- **"L'enigma di Fermat" Amir D.Aczel - edizioni Est**
- **"Kriptonite" Joe Lametta - edizioni Nautilus**
- **"The Official Pgp User's Guide" Philip R. Zimmermann - edizioni Mit**
- **"PGP : Pretty Good Privacy" Simson Garfinkel - edizioni O'Reilly & Associates**

Web Links:

- **http://www.pgpi.org**
- **http://www.gnupg.org**
- **http://www.codicicifrati.com**
- **http://www.cryptography.com/resources/papers/**
- **http://www.cs.berkeley.edu/~daw/people/crypto.html**
- **http://www.jyu.fi/~paasivir/crypt/**