**Corso di Laurea in Matematica - Anno Accademico 2017-2018**

# IN490 - Linguaggi di Programmazione

Docente: Marco Pedicini

Studente: Daniele Salierno

## 1 Introduzione

Lo scopo finale di questo progetto è la costruzione e l'attacco di un crittosistema di Elgamal e di un RSA costruiti a partire da primi di Sophie Germain.

Il documento si articola descrivendo gli algoritmi necessari al funzionamento del programma finale, seguendo il flusso presentato nel procedimento 1, che sintetizza al massimo il lavoro da svolgere.

**Algoritmo 1:** Workflow

1. Generazione di due primi di Sophie Germain $p < q$ sfruttando il Crivello ed il test di Solovay Strassen.
2. Creazione di un crittosistema di Elgamal con il primo $p$ trovato.
3. Creazione di un crittosistema RSA con i primi $p, q$ trovati.
4. Attacchi a Elgamal (Shanks e Pohlig Hellmann) e all'RSA (Wiener e $p-1$ o $\rho$ di Pollard).

Una prima parte del programma si occuperà di generare un primo di Sophie Germain, sfruttando tecniche di crivello per scartare a priori alcuni numeri che non soddisfano certe restrizioni modulari (come vedremo nella sezione 2); i numeri che superano il crivello sono sottoposti al test di primalità di Solovay Strassen.

Con i primi restituiti dall'algoritmo 2 vengono costruiti e attaccati i due crittosistemi. Sappiamo che i primi di Sophie Germain sono numeri molto buoni per la costruzione di crittosistemi, quindi ci aspettiamo che gli attacchi falliscano o che impieghino comunque molto tempo per terminare con successo.

## 2 Generazione di Primi

La prima fase del programma procede con due modalità: in un primo momento vengono generati i primi inferiori ad un bound B preso in input con il metodo del crivello di Eratostene; successivamente il programma passa alla modalità generazione di Sophie Germain, che è il cuore di questa sezione.

Il counter salta diversi ordini di grandezza e comincia a restituire numeri molto grandi di cui verificare la primalità. Ciascun candidato deve superare i normali test di divisibilità per i primi piccoli trovati dal crivello e di test definiti in una nuova classe (estensione del vecchio filter di erathostenes): la ragion d'essere dei nuovi test è che non stiamo

semplicemente cercando un numero primo, ma un primo di Sophie Germain, quindi sfruttiamo una condizione necessaria ma non sufficiente per scartarne alcuni e risparmiare tempo.

**Proposizione 1.** *Se* $p \equiv \frac{n-1}{2} \mod n$ *allora* $p$ *non è di Sophie Germain.*

*Dimostrazione.* Se $p$ fosse di Sophie Germain, allora $q = 2p + 1$ sarebbe primo, ma

$$2p + 1 = 2\frac{n-1}{2} + 1 = n \equiv 0 \mod n$$

che contraddice l'ipotesi di primalità di $q$.† □

**Algoritmo 2:** Generazione di Primi di Sophie Germain

1. Crivello di Eratostene con bound B.
2. Ricerca di un numero $p$ maggiore di un bound M che soddisfi le restrizioni modulari e di divisibilità $p \neq \frac{f-1}{2} \mod f \ \forall f$ filtro.
3. Test di Solovay Strassen per determinare se $p$ è primo.
4. Test di Solovay Strassen per determinare se $2p + 1$ è primo, dunque se $p$ è di Sophie Germain.

Quando un numero $p$ supera tutti i filtri è un candidato primo di Sophie Germain. Per prima cosa applichiamo il test di Solovay Strassen (algoritmo 3) a $p$ effettuando 30 iterazioni: se $p$ passa i test è composito con probabilità (che consideriamo trascurabile) di $\frac{1}{2^{30}}$.

Se $p$ passa il test è (molto probabilmente) primo: dobbiamo solo verificare se $q = 2p + 1$ lo è. Ripetiamo quindi il test di Solovay Strassen per $q$: se il risultato è nuovamente $q$ probabilmente primo, allora la generazione è finita e termina restituendo la coppia $p, q$.

Tutto questo procedimento è riassunto nell'algoritmo 2.

**Algoritmo 3:** Test di Solovay Strassen

**Input** : $n$ numero da testare, $k$ numero di iterazioni
**Output** : *false* se $n$ è composito, *true* se è primo

```
1  isPrime = true
2  for i = 1 → k do
3  |  Scegli a ∈ [2, n − 1]
4  |  x = (a/n)
5  |  if x = 0 ∨ a^((n−1)/2) ≢ x mod n then
6  |  |  isPrime = false
7  |  end
8  end
9  return (isPrime);
```

Nel test di Solovay Strassen sono implicitamente utilizzati un algoritmo per calcolare il simbolo di Legendre-Jacobi (alla riga 4) e uno per l'esponenziazione modulare nella condizione a riga 5 dell'algoritmo 3.

Per rendere il calcolo dell'esponenziazione modulare veloce si può usare l'algoritmo Square&Multiply.

# 3 Crittosistema di Elgamal

Per la costruzione del crittosistema è necessario trovare un generatore in $\mathcal{U}(\mathbb{Z}_q)$. Usiamo il seguente teorema.

**Proposizione 2.** *Se $p, q = 2p + 1$ sono una coppia di Sophie Germain, $x \in \mathbb{Z}_q$ è un generatore se e solo se $x^p \equiv -1 \mod q$.*

Osserviamo che poiché $p, q$ sono primi, vale $x^p \mod q = \left(\frac{x}{q}\right)$.

Simuliamo una comunicazione e un attacco tramite l'algoritmo di Shanks: una classe costruirà il crittosistema vero e proprio, cercando un generatore modulo $p$ e calcolando il resto della chiave; poi un'altra classe cifrerà un messaggio e lo invierà tramite il crittosistema; infine un'ultima classe intercetterà il messaggio e cercherà di decifrarlo senza conoscere la chiave privata.

**Algoritmo 4:** Algoritmo di Shanks

**Input** : $q$ primo, $g$ generatore di $\mathbb{Z}_q$, $y \in \mathbb{Z}_q$ messaggio da decifrare
**Output** : $x = \log_g y$

1 *Leggere $q$ e $g$*
2 $m = \lceil \sqrt{q-1} \rceil$
3 **for** $j = 0 \to m - 1$ **do**
4 | *Calcolare le coppie $(j, g^{mj})$ e ordinarle per la seconda componente*
5 **end**
6 *Leggere in input il messaggio $y$*
7 **for** $i = 0 \to m - 1$ **do**
8 | $z = y(g^i)^{-1}$
9 | **if** $\exists j : z = g^{mj}$ **then**
10 | | $x = mj + i$
11 | | $i = m - 1$
12 | **end**
13 **end**
14 **return** $x$