S-vLab

Numeri primi,
esponenziazione modulare
e scambio di Diffie-Hellman

Anna Riccioni
anna.riccioni@unibo.it

Sicurezza dell'Informazione M — Esercitazione del 20 aprile 2010

Teoremi
sull'esponenziazione
modulare

Teoremi sull'esponenziazione modulare

- Piccolo teorema di Fermat
  - $\forall$ primo p, $\forall x \in Z^*_p$ si ha:
  - $x^{p-1} \bmod p = 1$, cioè $x^{p-1} \equiv 1 \pmod p$
- Corollario 1 del piccolo teorema di Fermat
  - $\forall x \in Z^*_p$ si ha:
  - $s \equiv r \pmod{p-1} \Rightarrow x^s \equiv x^r \pmod p$
- Corollario 2 del piccolo teorema di Fermat
  - $x^{(p-1)/2} \bmod p$ è uguale a 1 o a -1

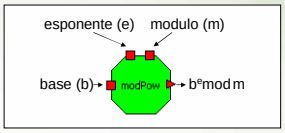
Corollario 2

T4.2: $x^{(p-1)/2} \bmod p$ è uguale a 1 o a -1.

	e									
b	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2	2	4	8	5	10	9	7	3	6	1
3	3	9	5	4	1	3	9	5	4	1
4	4	5	9	3	1	4	5	9	3	1
5	5	3	4	9	1	5	3	4	9	1
6	6	3	7	9	10	5	8	4	2	1
7	7	5	2	3	10	4	6	9	8	1
8	8	9	6	4	10	3	2	5	7	1
9	9	4	3	5	1	9	4	3	5	1
10	10	1	10	1	10	1	10	1	10	1

S-vLab: componenti per la verifica dei teoremi (1/2)

- modPow



- In Java:
  - BigInteger modPow(BigInteger exp, BigInteger m)
    Returns a BigInteger whose value is ($this^{exp}$ mod m).

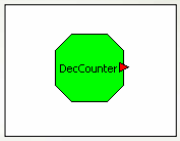
```
BigInteger base = new BigInteger("23", 16);
BigInteger exp= new BigInteger("11", 16);
BigInteger m = new BigInteger("37", 16);
BigInteger res = base.modPow(exp, m);
```

Java: BigInteger

- java.math.BigInteger
- BigInteger estende Number (come Integer)
  - La classe BigInteger offre metodi che svolgono operazioni analoghe a quelle svolte dagli operatori primitivi tra interi
    Es.: `add`, `subtract`, `multiply`, `divide`, `and`, `or`, `xor`, …
  - In più, sono disponibili metodi per il calcolo del MCD, operazioni di aritmetica modulare, generazione di primi, …
    Es.: `cgd`, `mod`, `modPow`, …
- Gestisce numeri "grandi"
  - `int`: 32 bit; `long`: 64 bit; `BigInteger`: dimensione variabile

S-vLab: componenti per la verifica dei teoremi (2/2)

- DecCounter: contatore decimale (in base 10)



	Counter	DecCounter
output	byte[], HEX	BigInteger, 10
input (stato iniziale)	byte[], HEX	BigInteger, 10
parametri	output size	-

modPow e DecCounter in Java

- Operazioni su BigInteger
  - modPow: esponenziazione modulare
  - DecCounter: incremento di un'unità (somma)
- Gli altri componenti di S-vLab corrispondono a classi
  - Cipher -> `Cipher`
  - Hash -> `MessageDigest`
  - PRNG -> `SecureRandom`
- modPow e DecCounter corrispondono invece a metodi di una classe (BigInteger)
  - modPow -> `BigInteger.modPow(BigInteger exp, BigInteger m);`
  - DecCounter -> `BigInteger.add(BigInteger.ONE);`

Algoritmi per l'esponenziazione modulare

Repeated square and multiply – A1

INPUT: $b \in Z_m$, $0 \le e < m$, $e = (e_{t-1}..e_0)_2$

OUTPUT: $b^e \bmod m$

1. Set y ←1. If e = 0 then return (y)
2. Set A ← b
3. If e_0 =1 then set y ← b
4. For i from 1 to t-1 do the following:
   4.1 Set A ←A^2 mod m
   4.2 If e_i =1 then set y ← A y mod m
5. Return (y)

m = 11, b = 7, e = 5 = $(101)_2$
1. **y = 1**
2. **A = 7**
3. **y = 7**
4. **i = 1**
   4.1 **A = 7 x 7 mod 11 = 5**
   i = 2
   4.1 **A = 5 x 5 mod 11 = 3**
   4.2 **y = 3 x 7 mod 11 = 10**
5. **7^5 mod 11 = 10**

Repeated square and multiply – A2

INPUT: $b \in Z_m$, $0 \le e < m$, $e = (e_{t-1}..e_0)_2$

OUTPUT: $b^e \bmod m$

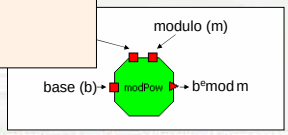
1. y ← 1
2. For i from t-1 down to 1 do the following
   2.1 y ←y^2 mod n
   2.2 If e_i =1 then y ← y b mod m
3. Return (y)

1. **y = 1**
2. **i = 2**
   2.1 **y = 1 x 1 mod 11 = 1**
   2.2 **y = 1 x 7 mod 11 = 7**
   i = 1
   2.1 **y = 7 x 7 mod 11 = 5**
   i = 0
   2.1 **y = 5 x 5 mod 11 = 3**
   2.2 **y = 3 x 7 mod 11 = 10**
5. **7^5 mod 11 = 10**

Repeated square and multiply – Tempi di esecuzione

Proprietà notevole di A1 e di A2:
Sia **t** il numero di bit della rappresentazione binaria dell'esponente **e**
Sia **n** il numero di bit dell'esponente con valore **1**
TEMPO DI ESECUZIONE → **t+n** moltiplicazioni
TEMPO MEDIO → **3/2 t** moltiplicazioni

serie di esponenti (e) da 256 bit con:
- 1 solo "1"
- 64 "1"
- 128 "1"
- 192 "1"
- 256 "1"



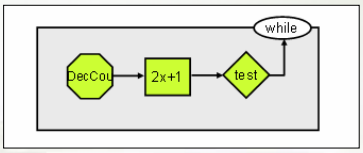
Numeri primi

Safe prime e serie di Sophie Germain

- Safe prime -> un numero primo che:
  - può essere espresso nella forma **p = 2q+1** con **q** primo (di Sophie-Germain)
  - è congruente a 5 mod 6

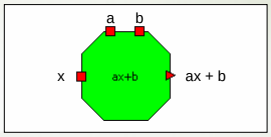
Generazione di numeri primi

- Possibile schema di ricerca di un primo sulla base di una progressione:



S-vLab: componenti per la generazione di numeri primi

- ax + b



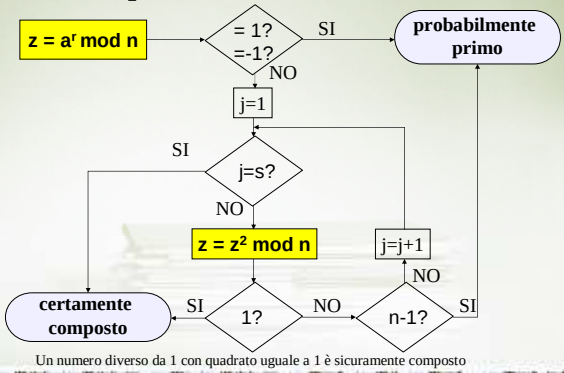
- generazione di progressioni
- individuazione di safe prime
- individuazione di primi di Sophie-Germain
- …

Test di primalità: Miller e Rabin

- **n** (il numero da sottoporre al test) deve essere dispari
- si pone **n-1 = r.2s** (con **r** dispari, **s ≥ 1)**
- si sceglie a caso un intero **a** e si calcola:

$$a^{n-1} \bmod n = (a^r)^{2^s} \bmod n$$

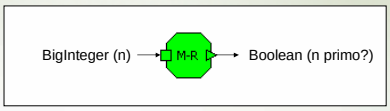
Test di primalità: Miller e Rabin



Un numero diverso da 1 con quadrato uguale a 1 è sicuramente composto

S-vLab: test di Miller e Rabin

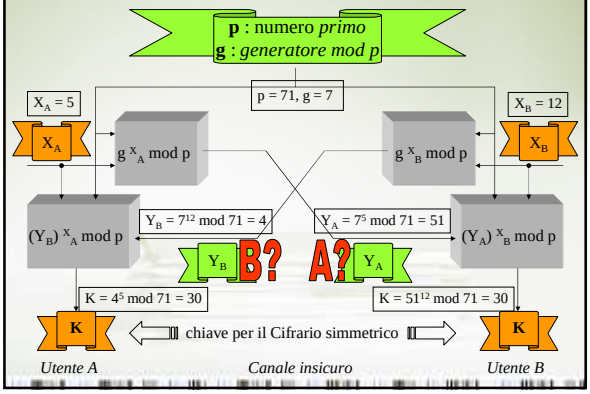
- M-R: test di Miller-Rabin



	M-R
input	BigInteger, 10
output	Boolean (n è primo con probabilità $1 - 1/2^{certainty}$)
parametri	certainty (`int`): contribuisce a definire l'affidabilità del risultato; influisce sul tempo di esecuzione

Scambio di Diffie-Hellman

Lo scambio di Diffie-Hellman



Scambio di Diffie-Hellman