

DIFFIE HELLMAN - DH

E' un ALGORITMO ASIMMETRICO per lo SCAMBIO DI CHIAVI.

PROBLEMA: distribuzione
(CIFRARIO SIMMETRICO) della chiave

PREMESSA: Questo algoritmo si fonda sull'**ARITMETICA MODULARE**... per cui dobbiamo imparare a ragionare su di essa:

MODULO: $a \text{ mod } M$ oppure $a \% M$

Resto della divisione di a per M .

Siamo in ambito intero.

A noi interesserà un caso particolare: M PRIMO

PROPRIETA':

- E' una **ARITMETICA FINITA** per cui non si rischia, in ottica informatica, di andare in overflow. I numeri infatti partono da 0 e arrivano fino a $M - 1$... dopodiché ci si ripete.

$$\bullet (ab) \text{ mod } M = (a \text{ mod } M)(b \text{ mod } M) \text{ mod } M$$

↑
X Dim.
vedi slide

ESEMPIO: $(527 \cdot 38) \text{ mod } 5 =$

$$(527 \text{ mod } 5)(38 \text{ mod } 5) \text{ mod } 5 =$$

$$(2)(3) \text{ mod } 5 = 1$$

$$\bullet (a^b) \text{ mod } 5 = (a \text{ mod } M)^b$$

↑
NO Dim.

DIVISORE ESATTO:

$$X|Y \quad \text{se} \quad Y \bmod X = 0$$

X divide Y .

Detto in altre parole: $\exists k$ t.c. $Y = kX$



ESEMPIO: $3|15$. 3 divide 15
poiché $15 \bmod 3 = 0$
analogamente $15 = 5 \cdot 3$

PROPRIETA':

- $\forall x \quad x|0$ \leftarrow Infatti: $0 = 0 \cdot x$
- $\forall x \quad x|x$ \leftarrow Infatti: $x = 1 \cdot x$
- $\forall y \quad 1|y$ \leftarrow Infatti: $y = y \cdot 1$
- Se: $X|Y, Y|Z \Rightarrow X|Z$ \leftarrow x Dim. vedi Slide
- Se: $X|Y, X|Z \Rightarrow X|(iY + jZ)$
- Se: $n|XY \quad \& \quad MCD(X, n) = 1 \Rightarrow n|Y$

↑
NO Dim.

CONGRUO:

$$x \equiv y \pmod{M}$$

se

$$x \pmod{M} = y \pmod{M}$$

ESEMPIO: $17 \equiv 12 \pmod{5}$

Poiché: $17 \pmod{5} = 12 \pmod{5} = 2$

PROPRIETÀ:

$$x \equiv 1 \pmod{M}$$

Significa che x diviso M da resto 1.

ESEMPIO: $17 \equiv 1 \pmod{8}$

DIFFIE HELLMAN

NOTI: q NUMERO PRIMO

α RADICE PRIMITIVA o GENERATORE

A | CHIAVE PRIVATA: $S_a < q$

A | CHIAVE PUBBLICA: $P_a = \alpha^{S_a} \text{ mod } q$

B | CHIAVE PRIVATA: $S_b < q$

B | CHIAVE PUBBLICA: $P_b = \alpha^{S_b} \text{ mod } q$

| CHIAVE CONDIVISA (DI SESSIONE):

$$K = P_b^{S_a} \text{ mod } q = P_a^{S_b} \text{ mod } q$$

A ↗ B ↘

$$(\alpha^{S_b})^{S_a} \text{ mod } q = (\alpha^{S_a})^{S_b} \text{ mod } q$$

PREMESSA: Affinche' l'Algoritmo sia forte rispetto ad un ATTACCO DI FORZA BRUTA, si deve avere un q NUMERO PRIMO, molto grande ...

A e B generano ciascuno le proprie chiavi:

una CHIAVE PRIVATA che rimane segreta
e a partire da essa una CHIAVE PUBBLICA

Dopodiche' A e B fanno due calcoli diversi, A parte da sx e B parte da dx... per mostrare che sono uguali.
Questo e' possibile grazie alla PROPRIETA' COMMUTATIVA dell'exp.

Solo A e B possono fare questo calcolo in tempi ragionevoli, poiche' conosciamo tutti gli "ingredimenti".

ESEMPIO:

DIFFIE HELLMAN

NOTI: q NUMERO PRIMO
 α RADICE PRIMITIVA

A | CHIAVE PRIVATA: $S_a < q$
 CHIAVE PUBBLICA: $P_a = \alpha^{S_a} \text{ mod } q$

B | CHIAVE PRIVATA: $S_b < q$
 CHIAVE PUBBLICA: $P_b = \alpha^{S_b} \text{ mod } q$

$$q = 7 \quad \alpha = 3$$

A | $S_a = 2$
 $P_a = 3^2 \text{ mod } 7 = 2$

B | $S_b = 6$
 $P_b = 3^6 \text{ mod } 7 = 1$

$$K = (1)^2 \text{ mod } 7 = 1 = (2)^6 \text{ mod } 7$$

CHIAVE CONDIVISA:

K = $P_b^{S_a} \text{ mod } q = P_a^{S_b} \text{ mod } q$



Per realizzare DIFFIE HELLMAN occorre:

- ALGORITMO per GENERARE CHIAVE PUBBLICA
- ALGORITMO per GENERARE q NUMERO PRIMO
- ALGORITMO per GENERARE α RADICE PRIMITIVA
 - GENERATORE

I primi due algoritmi, come vedremo, sono anche necessari per cifratura e firma con RSA

GENERARE RADICE PRIMITIVA

α

Partiamo subito da questo punto poiche' non lo approfondiremo. Tuttavia, e' bene sapere che dietro un GENERATORE c'e' un calcolo particolare.

Questo perche' ci sono certi numeri che sono inefficaci e facilitano l'avversario.

RADICE PRIMITIVA
o GENERATORE



numero $\alpha < q$ per cui qualunque numero tra 1 e $q-1$ viene espresso come potenza di $\alpha \pmod{q}$

OBIETTIVO: α moltiplicato per se stesso, in ottica, modulare genera tutti i numeri tra 1 e $q-1$

ESEMPI:

- $\alpha = 2 \pmod{7}$ NON va bene!

Poiche':

$$2^0 = 1$$

$$2^1 = 2$$

$$2^2 = 4$$

$$2^3 = 8 = 1$$

mi ripeto

Mi ripeto all'infinito.

Mi perdo il 3, 5, 6. 😞

- $\alpha = 3 \pmod{7}$

Va bene!

Poiche', si generano tutti i numeri da 1 a 6... 😊

GENERARE CHIAVE PUBBLICA

23/03/2023

Detto in altre parole, cerchiamo un ALGORITMO per calcolare:

$$a^b \text{ mod } q$$

L'operazione di generazione della CHIAVE PUBBLICA prende il nome di:

ESPOENTE MODULARE



COMPLESSITÀ
POLINOMIALE 😊

per

NOI

Trovare P_a conoscendo
gli ingredienti α, q, S_a

LOGARITMO DISCRETO



COMPLESSITÀ
ESPOENZIALE 😊

per

AVVERSARIO

Risalire ad S_a a partire
da α, q, P_a

La ragione risiede nel fatto
che questi numeri sono
imprevedibili per l'avversario.

Detto in altre parole: calcolare in avanti (NO) è facile,
calcolare indietro (AVVERSARIO) è difficile.
Cioè, c'è una osimmetria.

DISCRETO poiché siamo in ambito di numeri interi.

EXTRA: C'è una corrispondenza tra i numeri di n cifre decimali e K bit
... che è di 1:3 (il triplo!).

ESEMPIO: $81 = 1010001$ 2 cifre → 7 cifre (~ il triplo!)

Questo ci serve per comprendere che stiamo lavorando con grandi numeri ... per cui dovremmo chiederci la fattibilità dei calcoli da un punto di vista pratico.

PROBLEMA

$$a^b \text{ mod } q$$

SOLUZIONI:

//Metodo RICORSIVO

```
int expmod (int a, int b, int q) {
    if (b == 0) return 1; // a^0 mod q = 0
    if (b%2 == 0)// b pari
        return sq(expmod(a,b/2,q))%q;
    else// b dispari
        return (a*expmod(a,b-1,q))%q;
}
```

COMPLESSITÀ
LOGARITMICA



$$\log(b)$$

^b
PARI

^b
DISPARI

$$2 \cdot \log(b)$$

OBIETTIVO: Avvicinarmi allo 0

Nel caso di b DISPARI ci avviciniamo di 1.

Nel caso di b PARI ci avviciniamo moltissimo:
facciamo un salto di metà

CASO MIGLIORE: b sempre PARI $\Rightarrow \log(b)$

CASO PEGGIORE: b sempre DISPARI $\Rightarrow 2 \cdot \log(b)$

↑
se b DISPARI ↫
al caso successivo
avrei b PARI ☺

//Metodo ITERATIVO

// $b = b_k \dots b_2 b_1 b_0$; vedo il numero espresso in bit

```
d=1;
for (i=k; i>=0; i--){
    d = (d*d)%q;
    if (bi == 1)
        d = (d*a)%q;
}return d;
```

NO Dim.

Questo algoritmo e' ancora più efficiente del precedente!

COMPLESSITÀ
LOGARITMICA ☺

$$\log(b)$$

GENERARE NUMERO PRIMO

q

E' cruciale in questo punto il **TEST PRIMALITÀ** scelto.

**TEST PRIMALITÀ
INGENUO**

Gemero q , Divido per tutti i numeri più piccoli e verifico che tutti i resti delle divisioni siamo $\neq 0$.

MIGLIORIA

Contro: **COMPLESSITÀ
ESPOENZIALE** 😞

ESEMPIO: 7 PRIMO?
Divido per 6, 5, ..., 1

**TEST PRIMALITÀ
- INGENUO**

Mi posso fermare alla radice quadrata; non mi serve andare oltre nella ricerca di un divisore.

dire $i < \sqrt{M}$ →
e' equivalentemente
a dire $i^2 <= M$

1. Genera M di k bit a caso
2. for ($i=2; i^2 <= M; i++$)
if ($M \% i = 0$)
 goto 1 // M non primo
3. return M // M primo

Contro:

E' ancora di
**COMPLESSITÀ
ESPOENZIALE** 😞

Pro:

Seppur possa sembrare il contrario, ho la certezza di trovare un numero primo.

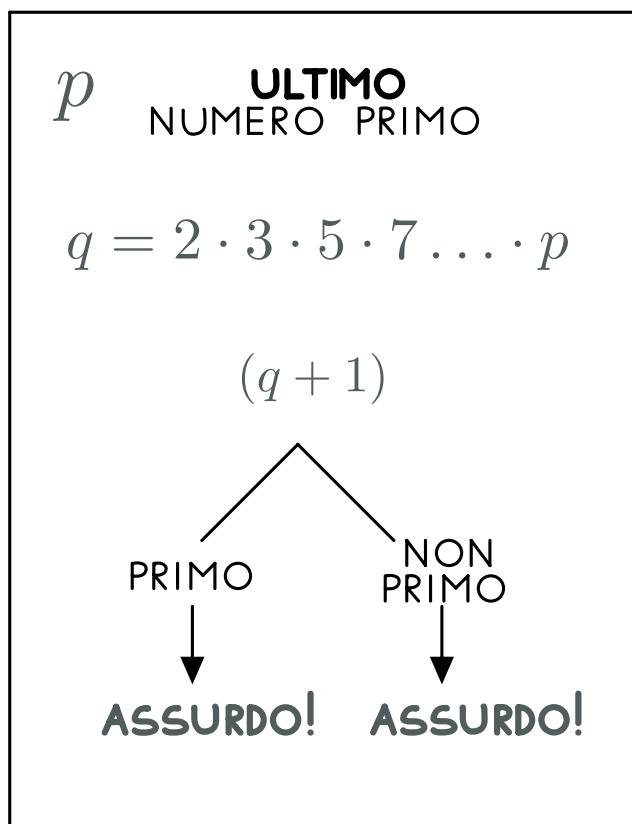
Cioè è possibile grazie al:

- **TEOREMA EUCLIDE**
- **TEOREMA NUMERI PRIMI**

TEOREMA EUCLIDE

Esistono INFINITI NUMERI PRIMI

DIMOSTRAZIONE (X ASSURDO!):



Supponiamo che p sia l'ultimo numero primo.

Sia q il prodotto dei numeri primi fino a p .

Distinguo due casi:

• $(q + 1)$ PRIMO → ASSURDO!

poiché avevo considerato p come ULTIMO PRIMO.

• $(q + 1)$ PRIMO → ASSURDO!

poiché questo implica che è divisibile per un certo numero $r > p \dots$ con r PRIMO.

poiché non è divisibile per alcun primo fino a p

TEOREMA NUMERI PRIMI (NO Dim.)

Grazie a questo teorema possiamo sapere quanti numeri sono presenti in un range (intervallo):

Sia $\pi(x)$ il numero di NUMERI PRIMI minori di x .

Allora: $\pi(x) \cong x/\ln(x)$ \Rightarrow equiventemente $\lim_{x \rightarrow \infty} [\pi(x)/(x/\ln(x))] = 1$

ESEMPIO: $\pi(10) = 4$ $10/\ln(10) \simeq 4,34$

2, 3, 5, 7

In crittografia (es: DH, RSA) useremo numeri primi molto grandi di almeno 100 cifre decimali (~ 300 bit).

Ci chiediamo: è facile che un numero casuale M di 100 cifre sia primo?

è facile generare un casuale M (molto grande)

Grazie a questo teorema sappiamo che:

$$\bullet \pi(10^{100}) - \pi(10^{99})$$


eliminiamo i numeri superflui

Numeri di esattamente 100 cifre

$$\bullet 10^{100}/\ln(10^{100}) - 10^{99}/\ln(10^{99}) \cong 3,9 \cdot 10^{97}$$

E quindi: $\underbrace{100^{100}}_{\text{casi favorevoli}} / \underbrace{3,9 \cdot 10^{97}}_{\text{casi possibili}} \simeq \boxed{256}$ tentativi.

Questo calcolo si può ottimizzare ad esempio non generando i multipli di 2, 3, 5, ... In ogni caso comunque si tratta di un numero costante.

L'idea e' avere un **TEST PRIMALITA'** di complessita' **POLINOMIALE**
... dopodiché lo ripeto per un certo numero costante di volte e
quindi sempre di complessità polinomiale.

Conclusione: il TEST PRIMALITA' **INGENUO** e' efficace.

Tuttavia richiede COMPLESSITA' ESPONENZIALE

TEST PRIMALITA'
PROBABILISTICI: Non sempre dammo la risposta corretta.
Tuttavia questi TEST sono detti
"ASIMMETRICI" nel senso che:



I TEST sono **INDIPENDENTI**:

PROBABILITA' TEST **DI SBAGLIARE**: $< \frac{1}{2}$ ← **e il minimo sindacale**

PROBABILITA' TEST **TOTALE** **DI SBAGLIARE**: $< \frac{1}{2^T} = 2^{-T}$

$$\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots$$

\downarrow
2 volta: $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2^2}$

PROBABILITA' TEST **TOTALE CORRETTA**: $> 1 - 2^{-T}$

1. Genera M di k bit a caso
2. for ($i=0$; $i < T$; $i++$) // Ripeti T volte
 - genero $a < M$ a caso
 - if(**Test(a,M)**) goto 1 // M non primo
3. return M // M primo con Probabilità $> 1 - 2^{-T}$

La PROBABILITÀ aumenta esponenzialmente: mi basta ripetere un certo numero di volte per avere una probabilità di errore estremamente bassa:

ESEMPIO: $T = 100$

$$P(M \text{ PRIMO }) > 1 - 2^{-T}$$

Noi useremo il **TEST DI MILLER-RABIN** com
che si basa su:

$$P(M \text{ PRIMO }) > \frac{1}{4}$$

• PICCOLO TEOREMA DI FERMAT

Se M **PRIMO** e $a = M \Rightarrow a^{M-1} \bmod M = 1$

• PRINCIPIO FONDAMENTALE

Se M PRIMO e $x^2 \bmod M = 1 \Rightarrow x = 1 \text{ o } \underbrace{x = M - 1}_{\substack{\uparrow \\ \text{im } x = -1}}$

```

//Test di Miller-Rabin
// M-1 = bk...b2b1b0
d = 1;
for (i=k; i>=0; i--) {
    x = d;
    d = (d*d)%M; // d ≡ x^2 (mod M)

    if (d == 1 && x != 1 && x != M-1)
        return(TRUE); // 1 ≡ x^2 (mod M) - Conseguenza del Principio Fondamentale
    if (bi == 1)
        d = (d*a)%M;
}// d = aM-1 mod M = 1 - Piccolo Teorema di Fermat

if (d != 1)
    return TRUE;
else
    return (FALSE);

```

Conseguenza del Principio Fondamentale

Se M è PRIMO e $x^2 \text{ mod } M = 1 \Rightarrow x = 1 \text{ o } x = -1$



Principio Fondamentale:

Se $\exists x, y \text{ t. c. } x^2 \equiv y^2 \pmod{n} \text{ e } \neg(x \equiv y \pmod{n}) \Rightarrow n \text{ NON primo}$



Dimostrazione (lez.23-03-2023):

Sia $d = MCD(x - y, n)$

Ci sono due casi:

- $d = n \Rightarrow x \equiv y \pmod{n} \Rightarrow d \neq n$
- $d = 1 \Rightarrow d \neq 1$

Sappiamo che $n \mid x^2 \equiv y^2 \Rightarrow n \mid (x - y)(x + y)$

Quindi n non può dividere $x - y$...quindi deve dividere $x + y$

Questo però è in contraddizione con $\neg(x \equiv y \pmod{n}) \Rightarrow d \neq 1$

Quindi: $d = MCD(x - y, n) \neq n$ e $d \neq 1 \Rightarrow n \text{ NON primo}$

$$\text{Se } M \text{ è PRIMO e } a < M \implies a^{M-1} \bmod M = 1$$

Si dimostra generalizzando il Teorema di Eulero; Il Piccolo Teo di Fermat, ne diventa un corollario.

Dimostrazione (lez.28-03-2023):

Definisco: $\Phi(n) = \text{numeri interi } < n \mid \text{numeri interi relativamente primi con } n$
 $a, b \text{ relativamente primi} = \text{numeri con } MCD = 1$ Esempio: 4,9 non primi ma coprimi

Teorema di Eulero: Se M, a relativamente primi a $a^{\Phi(M)} \bmod M = 1$

Conseguenza (utile per DH e RSA):

Se M, a relativamente primi $\Rightarrow ax \equiv ay \pmod{M} \implies x \equiv y \pmod{M}$

//Posso dividere per a e semplificare.

//Dimostrazione: vedi RSA

Dimostrazione del Teo. di Eulero:

Sia $R = \{x_1, x_2, \dots, x_{\phi(M)}\}$ l'insieme dei numeri minori di M relativamente primi con M .

Sia $S = \{ax_1 \bmod M, ax_2 \bmod M, \dots, ax_{\phi(M)} \bmod M\}$.

Dimostriamo ora che $S=R$.

1) S incluso in R

$ax_i \bmod M$ è relativamente primo con M //Poiché sia a sia x_i sono relativamente primi con M

2) R incluso in S

Non ci sono ripetizioni in S.

//Poiché, in caso contrario: $ax_i \bmod M = ax_j \bmod M \implies x_i \equiv x_j \pmod{M}$

//Ma, per la Conseguenza sopra vista, posso semplificare ($x_i \equiv x_j \pmod{M}$) ottenendo però una //contraddizione con la definizione di R: elementi tutti diversi... per cui in S non ci sono ripetizioni

La dimostrazione del teorema si conclude con:

//R=S per cui la moltiplicazione è una funzione

$$\begin{aligned} R &= \{x_1, x_2, \dots, x_{\phi(M)}\} = \{ax_1 \bmod M, ax_2 \bmod M, \dots, ax_{\phi(M)} \bmod M\} = S = \\ &= x_1, x_2, \dots, x_{\phi(M)} \bmod M = (ax_1 \bmod M)(ax_2 \bmod M) \dots (ax_{\phi(M)} \bmod M) = \\ &= x_1, x_2, \dots, x_{\phi(M)} \bmod M = (ax_1)(ax_2) \dots (ax_{\phi(M)}) \bmod M = \\ &= x_1, x_2, \dots, x_{\phi(M)} \bmod M = a^{\phi(M)}(x_1, x_2, \dots, x_{\phi(M)}) \bmod M //Porto in evidenza a, $a^{\phi(M)}$ volte \\ &\quad //Siccome x_i sono tutti relativamente primi con M , li posso semplificare: \\ &\quad = 1 \bmod M = a^{\phi(M)} \bmod M \end{aligned}$$

CONCLUSIONI

DIFFIE HELLMAN ci consente di stabilire una **CHIAVE CONDIVISA** e segreta utilizzando un canale di comunicazione insicuro senza la necessità che le due parti si siano scambiate informazioni in precedenza.

Contro: **NON** permette direttamente di **CIFRARE/AUTENTICARE**.

Potrebbe solo da **ATTACCHI PASSIVI**

Lettura non autorizzata
dei dati trasmessi su rete

NON potrebbe da **ATTACCHI ATTIVI**

**ATTACCO
MAN IN THE MIDDLE**

ATTACCO MAN IN THE MIDDLE

ATTACCO: CHIAVE PUBBLICA **FALSA!**

