

Problemi difficili e assunzioni critografiche

GenMod(). algoritmo ppt che genera due primi
 $p + q$ di n bit , $N = p \cdot q$

Factor
 A, GenMod (n)

1. C esegue GenMod(1^n) per ottenere (p, q, N)

2. A riceve da C il modulo N e da $p', q' > 1$

3. Se $p' \cdot q' = N$, allora C dà in output 1, altrimenti 0

Def. Relativamente a GenMod(1^n) il problema della fattorizzazione è difficile se, per ogni A ppt, esiste una funzione trascurabile $\text{negl}(n)$:

$$\Pr_{A, \text{GenMod}}[\text{Factor}_{A, \text{GenMod}}(n) = 1] \leq \text{negl}(n)$$

Assunzione (Fattorizzazione) Esiste un algoritmo GenMod(1^n) relativamente al quale il problema della fattorizzazione è difficile

Algoritmi per la fattorizzazione: stato dell'arte

- Pollard's $p-1$ (applicabile se $p-1$ ha molti fattori piccoli)

$$O(B \log B \cdot (\log n)^2 + (\log n)^3) \leq \text{tutti} \leq B$$

Quindi $x B = O((\log N)^c)$

tempo poly
prob success bassa

$$x B = O(\sqrt{N})$$

tempo exp
prob success alte

Quando i fattori di $(p-1) \times (q-1)$ sono grandi (safe primes)

Idea = cerca di calcolare un x tale che $\text{MCD}(x, N) = p$
i fattori piccoli di $p-1$ servono per calcolare x

. Pollard's rho (general purpose)

Cerca di trovare due interi $x < x'$ equivalenti
modulo p (senza conoscere p) in modo tale che

$$\text{MCD}(x - x', N) = p$$

La ricerca della coppia (x, x') viene effettuata
in modo efficiente

Complexità $O(2^{n/4})$

migliora rispetto alla ricerca esaurienta

• Quadratic Sieve (general purpose)

Sfrutta un'altra idea comune agli algoritmi di fact.
sviluppati a partire dagli anni '70.

Cerca di trovare $x < y$ tali che $x^2 \equiv y^2 \pmod{N}$
con $|x| \neq \pm y$ per calcolare

$$p = \text{MCD}(x-y, N)$$

Complexità $O(2^{\sqrt{n \log n}})$

(Nota che $c \cdot \sqrt{n \log n}$ cresce meno di \sqrt{n} , sub-exponential)

Più efficienti al momento

- Number field sieve
 - Elliptic curve factoring algorithm
- Per valori di n molto grandi il primo ha
il miglior tempo di esecuzione

Problema RSA

GenRSA ppt genera due primi $p \neq q$ di n bit
 $N = p \cdot q$ e due interi e, d maggiori di zero tali che
 $\text{MCD}(e, \varphi(N)) = 1$ ed $\equiv 1 \pmod{\varphi(N)}$

RSA-inv (n)
A, GenRSA

1. C esegue GenRSA(1^n) per ottenere (N, e, d)
2. C sceglie in modo uniforme $y \in \mathbb{Z}_N^*$
3. A riceve da C (N, e, y) e dà a C $x \in \mathbb{Z}_N^*$
4. Se $x^e \equiv y \pmod{N}$, C dà un output 1; altrimenti, 0;

Esempio

$$\text{GenMod}(1^*) \rightarrow (N, p, q) = (153, 11, 13)$$

$$\varphi(N) = (11-1)(13-1) = 10 \cdot 12 = 120$$

Scegiamo $e = 7$. Usando Extended-Euclid, $d = 103$
 $(7 \cdot 103 \equiv 1 \pmod{120})$

GenRSA dà in output $(153, 7, 103)$

\downarrow
 N , e , d

Scelte di e popolari: $e = 3$

di scelgo $p \neq q$ tali che $p, q \not\equiv 1 \pmod{3}$
 $\Rightarrow \gcd(3, \varphi(n)) = 1$

Questa scelta è soggetta ad attacchi per esp. piccoli

$$e = 2^{16} + 1 = 65537 \quad (\text{peso di Hamming } d \text{ e basso})$$

1 0 . . . 0 1

Leggermente più costosa di $e = 3$

Attenzione: d piccolo è una cattiva idea

Potrebbero essere applicati attacchi di forza bruta.

Anche se $d \approx N^{\frac{1}{2}}$ esistono

algoritmi noti per recuperare d da N e e

Problema descritto di \mathbb{Z}_N^*

$f_e : \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$ sappiamo è una permutazione

f_d è la permutazione inversa

Conosciuto il calcolare la radice ℓ -esima di y solto

a costo in \mathbb{Z}_N^*

Conoscendo d è facile. Il problema riduce a fare senza

Def. Relativamente a $\text{Gen RSA}(1^n)$, il problema dell'inversione della permutazione RSA è difficile, $\forall A \text{ ppt}$, esiste una funzione trascurabile $\text{negl}(n)$:

$$\Pr_{A, \text{Gen RSA}}[\text{RSA-inr}_{A, \text{Gen RSA}}(n) = 1] \leq \text{negl}(n)$$

Assunzione RSA Esiste un algoritmo $\text{Gen RSA}(1^n)$ relativamente al quale il problema dell'inversione della permutazione RSA è difficile.

Fattorizzazione ed RSA

Penso fattorizzare? Dati p, q , calcolo $\varphi(N)$,
calcolo $d = e^{-1} \pmod{\varphi(N)}$, inserito!

Pertanto, indicando con tenMod il gen in tenRSA,
affinché il prob RSA sia difficile relativamente a tenRSA,
il prob. della fattorizzazione deve essere difficile relativamente
a tenMod .

RSA non può essere più difficile di Factoring

Factoring difficile \Rightarrow RSA difficile ?

NON lo sappiamo, grossa questione aperta

Ma sappiamo di calcolare d a partire
da (N, e) è tanto difficile quanto fattorizzare

Pertanto, se l'UNICO modo per inventare RSA fosse
tramite il calcolo preventivo di d , allora si
potrebbe concludere che le due assunzioni sono
equivalenti.

Teorema (Fattorizzazione di N dato d) .

Ese,te un A ppt che, ricevendo in input $N = p \cdot q$ con p, q primi, e gli interi e, d : $ed \equiv 1 \pmod{\phi(n)}$, dà in output la fattorizzazione di N , eccetto con probabilità trascurabile

Idea: $x^2 \equiv 1 \pmod{N}$ ha \leq radici quadrate

Due sono quelle banali ± 1

Ogni radice q- non banale può essere usata per calcolare eff. un fattore d di N tramite

$$\text{MCD}(x^{\pm 1}, N)$$

Problema del logaritmo discreto

Sia $\text{Gen}(G)$ un alg. ppt per la generazione di gruppi ciclici

$$(G, g, q)$$

insieme elementi ↑ ↗
generatore ordine del gruppo

L'operazione \oplus è efficientemente calcolabile

L'appartenenza di un el a G è efficientemente calcolabile

$$G = \{g^0, g^1, \dots, g^{q-1}\}, \quad \forall h \in G, \exists! x \in \mathbb{Z}_q :$$

$$g^x = h$$

log_g h

$D \log_{A, \text{ten} \mathcal{G}}(u)$

1. C esegue $\text{GenG}(1^n)$ e ottiene (\mathcal{G}, g, q)
 2. C sceglie in modo uniforme $h \in \mathcal{G}$
 3. A riceve (\mathcal{G}, g, q) ed h e dà $x \in \mathbb{Z}_q$
 4. Se $g^x = h$, allora C dà in output 1; altrimenti, 0.
-

Problema del log discreto: calcolare $x = \log_g h$, per h scelto unif. a caso

Esemp:

$$(\mathbb{Z}_p^*, g, p^{-1}) \quad h \in \mathbb{Z}_p^*, \quad x \in \mathbb{Z}_{p-1} : \quad g^x \equiv h \pmod{p}$$

$$(\mathcal{E}, P, q) \quad Q \in \mathcal{E}, \quad x \in \mathbb{Z}_q : \quad Q = x^P$$

Def. Relativamente a $\text{Gen} \mathcal{G}()$, il problema DL è difficile se, $\forall A$ ppt, esiste una funzione trascurabile, tale che

$$P_2 [D\log_{A, \text{Gen} \mathcal{G}}(n) = 1] \leq \text{negl}(n)$$

Assunzione DL. Esiste un $\text{Gen} \mathcal{G}()$ relativamente al quale il problema del logaritmo discreto è difficile.

Algoritmi per il calcolo del logaritmo discreto

- Pohling - Hellman, ordine q composto, fattorizzazione nota

Idea : il problema viene ridotto al calcolo del logaritmo discreto in sottogruppi

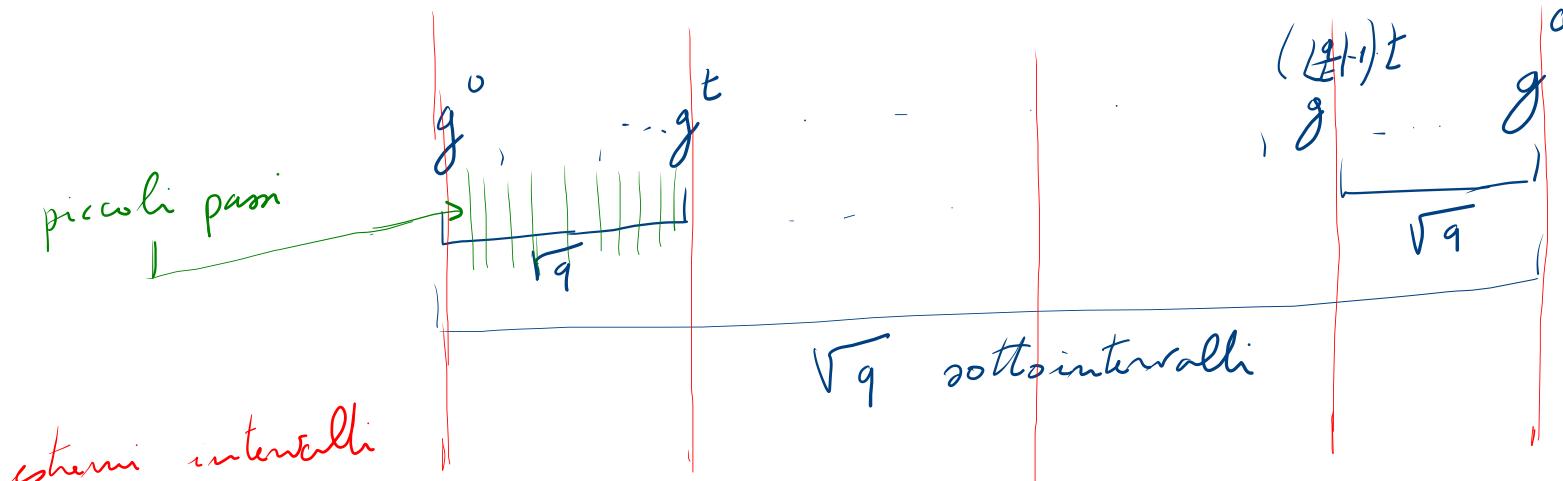
Complessità \leq complessità di risoluzione
nel n. t. di ordine q' con
 q' max tra gli ordini
dei sottogruppi.

(i.e., q' max divisore di q)

• Baby step / giant step di Shanks

Calcola in tempo $O(\sqrt{q} \text{ poly}(\log(q))) = O(2^{kt} \text{ poly}(n))$

memorizza $O(\sqrt{q})$ valori



h appartiene ad uno degli intervalli

Calcolando $h \cdot g^1, h \cdot g^2, \dots, h \cdot g^t$ "shiftano" h

fino a raggiungere l'estremo. Per cui, per qualche i

$$h \cdot g^i = g^{kt} \Rightarrow x = kt - i \pmod{q}$$

ALGORITHM 10.6

The baby-step/giant-step algorithm

Input: Elements $g, h \in \mathbb{G}$; the order q of \mathbb{G}

Output: $\log_g h$

$t := \lfloor \sqrt{q} \rfloor$

for $i = 0$ to $\lfloor q/t \rfloor$:

compute $g_i := g^{i \cdot t}$

sort the pairs (i, g_i) by their second component

for $i = 1$ to t :

compute $h_i := h \cdot g^i$

if $h_i = g_k$ for some k , **return** $[(kt - i) \bmod q]$

Pollard's Rho

"Trade-off tempo / spazio"
rispetto all'algor. di Shanks

Nota: Shanks e Pollard's Rho sono algoritmi ottimali relativamente agli algoritmi generici per il calcolo del logaritmo discreto

Ogni algoritmo generico non può risolvere il problema con complessità meno di esponenziale

Tuttavia, algoritmi specifici che sfruttano la rappresentazione del gruppo possono fare meglio

Problema Diffie - Hellman

$\text{Gen}_\mathsf{B}(\cdot)$ alg. ppt (G, g, q) Sotto ipotesi DL

$\text{CDH}_{A, \text{Gen}_\mathsf{B}}(n)$

1. C esegue $\text{Gen}_\mathsf{B}(1^n)$ per ottenere (G, g, q)

2. C sceglie in modo uniforme $x_1 \in \mathbb{Z}_q, x_2 \in \mathbb{Z}_q$ e

calcola

$$h_1 = g^{x_1} \in \mathsf{G}, \quad h_2 = g^{x_2} \in \mathsf{G}$$

3. A riceve $(\mathsf{G}, g, q), h_1, h_2, c$ dà in output $h_3 \in \mathsf{G}$

4. Se $h_3 = g^{x_1 x_2}$, allora C di 1. Altrimenti, 0;

Il problema Diffie-Hellman computazionale consiste, quindi, nel calcolare $h_3 = g^{x_1 x_2}$ per elementi scelti a caso h_1, h_2

$$(\mathbb{Z}_p^*, g, p^{-1}) \quad h_1 = g^{x_1} \text{ mod } p, \quad h_2 = g^{x_2} \text{ mod } p \Rightarrow h_3 = g^{x_1 x_2} \text{ mod } p$$

$$(E, P, g) \quad h_1 = x_1 P, \quad h_2 = x_2 P \Rightarrow h_3 = x_1 x_2 P$$

Def. Relativamente a $\text{Gen} \in \text{F}(1^n)$, il problema Diffie-Hellman computazionale è difficile se, $\forall A \text{ ppt}$, esiste una funzione trascurabile $\text{negl}(n)$, tale che

$$\Pr_2 [CDH_{A, \text{negl}}(n) = 1] \leq \text{negl}(n)$$

Assunzione (CDH) . Esiste un algoritmo ben & (1^m) relativamente al quale il problema Diffie-Hellman computazionale è difficile

Esiste anche una variante decisionale del problema, Diffie-Hellman decisionale. Richiede di distinguere tra

$$(h_1, h_2, h_3)$$


Diffie-Hellman

$$(h_1, h_2, h_3)$$


inf. a caso

DDH_{A, BnG}(n)

1. C'è un g_b bit (1ⁿ) per ottenere (t, g, q)

2. C'è scegli unif. a caso $x_1 \in \mathbb{Z}_q$, $x_2 \in \mathbb{Z}_q$ e calcola

$$h_1 = g^{x_1}, \quad h_2 = g^{x_2}$$

3. C'è scegli un bit b. Se $b = 1$, calcola $h_3 = g^{x_1 x_2}$

Altrimenti, scegli x_3 unif. a caso e calcola $h_3 = g^{x_3}$

4. A riceve $(t, g, q) \leftarrow (h_1, h_2, h_3)$ e dà a C b'

5. Se $b' = b$, allora C dà 1. Altrimenti, 0.

Def. Relativamente a $\text{Gen}^t(1^n)$, il problema Diffie-Hellman decisionale è difficile se, $\forall A \text{ ppt}$, esiste una funzione trascurabile $\text{negl}(n)$, tale che

$$\Pr_{A, \text{Gen}^t} [\text{DDH}_{A, \text{Gen}^t}(n) = 1] \leq \frac{1}{2} + \text{negl}(n)$$

Assunzione (DDH). Esiste un algoritmo $\text{Gen}^t(1^n)$ relativamente al quale il problema Diffie-Hellman decisionale è difficile

Relazioni tra i problemi

DL, CDH, DDH

DL facile \Rightarrow CDH facile

Dati h_1 e h_2 , calcolo $x_1 = \log_{\frac{h_1}{h_2}} h_1$ e poi $h_2^{x_1} = h_3$

Non sappiamo se DL difficile \Rightarrow CDH difficile

CDH facile \Rightarrow DDH facile

Data la tripla (h_1, h_2, h_3) , calcolo h_3' da h_1 e h_2
e controllo se $h_3' = h_3$

(DH difficile $\stackrel{?}{\Rightarrow}$ DDH difficile)

Non sembra essere vero: ci sono dei gruppi in cui,
allo stato attuale delle nostre conoscenze, DL e
CDH sembrano essere difficili mentre DH
è facile.

\hookrightarrow (e.g. non è vera in \mathbb{Z}_p^* , p primo)

DDH non è vera nel sottogruppo di ordine q di \mathbb{Z}_p^*
dove $p = 2q + 1$.

Gruppi ciclici di ordine primo

Preferiti perché

- DL più difficile da risolvere con gli alg. noti
- DDH sembra valere
- Trovare un generatore è immediato
- Semplificano le prove di sicurezza ovvero richiedono calcoli inversi moltiplicativi. Nei gruppi di ordine primo ogni esponente è invertibile
- Usando DDH, se $|G| = q$, q primo, la distribuzione dei valori $g^{x_1 x_2}$, $x_1, x_2 \in \mathbb{Z}_q$ unif. a caso, è quasi uniforme

Effective Key Length	RSA	Discrete Logarithm	
	Modulus N	Order-q Subgroup of \mathbb{Z}_p^*	Elliptic-Curve Group Order q
112	2048	$p: 2048, q: 224$	224
128	3072	$p: 3072, q: 256$	256
192	7680	$p: 7680, q: 384$	384
256	15360	$p: 15360, q: 512$	512

FIGURE 10.1: All values are in bits, e.g., for a 112-bit effective key length in the RSA setting, a 2048-bit modulus N should be used.

Campi finiti

Un campo (S, \oplus, \odot) è una struttura algebrica costituita da un insieme di elementi S e due operazioni binarie, \oplus e \odot , definite su di esso, che godono delle seguenti proprietà:

1. (S, \oplus) è un gruppo abeliano
2. $(S \setminus \{e\}, \odot)$, dove e è l'identità in S rispetto ad \oplus , è un gruppo abeliano
3. Vale la proprietà distributività. Per ogni $a, b, c \in S$ risulta

$$(a \oplus b) \odot c = a \odot c \oplus b \odot c$$

Se risulta $|S| < \infty$, ovvero S ha un numero finito di elementi, allora il campo si dice *finito*.

$$(Q, +, \cdot)$$

razionali

$$(R, +, \cdot)$$

reali

$$(C, +, \cdot)$$

complessi

infiniti

finito \rightarrow

$$(\mathbb{Z}_p, +_p, \cdot_p)$$

primo

$$\begin{cases} (\mathbb{Z}_p, +_p) \\ (\mathbb{Z}_p^*, \cdot_p) \end{cases}$$

e vale prop. dist.

Esistono campi finiti se e solo se $n = p^e$, p primo, $e \geq 1$
caratteristica del campo

Come si costruiscono? Idea...

p primo, $\mathbb{Z}_p[x]$ insieme tutti i polinomi in x
con coefficienti in \mathbb{Z}_p
sommati e moltiplicati (con op. in \mathbb{Z}_p)

$f(x), g(x) \in \mathbb{Z}_p[x]$, diremo che $f(x)$ divide $g(x)$

$f(x) | g(x)$ se esiste $q(x) \in \mathbb{Z}_p[x]$ tale che

$$g(x) = q(x) f(x)$$

grado di f $\rightarrow \deg(f)$ = esponente più grande term. d' f .

Come per gli interi, in generale,

$$g(x) = q(x) \cdot f(x) + r(x)$$

\uparrow \downarrow
 $\deg(f) = n$ $\deg(r) < n$

I polinomi $q(x)$ e $r(x)$ sono unici

Indicheremo $r(x)$ con $g(x) \pmod{f(x)}$

Dati $f(x), g(x), h(x) \in \mathbb{Z}_p[x]$, $\deg(f) = n \geq 1$.

$g(x)$ è congruente ad $h(x)$ modulo $f(x)$ e scriviamo

$$g(x) \equiv h(x) \pmod{f(x)}$$

$$x \quad g(x) \pmod{f(x)} = h(x) \pmod{f(x)}$$

Ogni polinomio in $\mathbb{Z}_p[x]$ è congruente ad un unico polinomio di grado al più $n-1$

Come per gli interi in \mathbb{Z} , partizionati in classi di \mathbb{Z}_n stiamo partizionando $\mathbb{Z}_p[x]$ in classi di congruenza rispetto al polinomio $f(x)$ prescelto.

Indicheremo l'insieme delle classi di congruenza con

$$\mathbb{Z}_p[x]/f(x) \quad \begin{pmatrix} \text{tutti i polinomi} \\ \text{di grado al più } n-1 \end{pmatrix}$$

$$|\mathbb{Z}_p[x]/f(x)| = p^n, \quad a_{n-1}x^{n-1} + \dots + a_1x + a_0$$
$$\downarrow \qquad \qquad \qquad \downarrow \qquad \qquad \downarrow$$
$$\mathbb{Z}_p \qquad \mathbb{Z}_p$$

Un polinomio $f(x)$ si dice irriducibile se non esistono due polinomi $f_1(x)$ ed $f_2(x)$ tali che

$$f(x) = f_1(x) \cdot f_2(x)$$

dove $\deg(f_1) > 0$, $\deg(f_2) > 0$.

$(\mathbb{Z}_p[x]/f(x), +_{\text{poly}}, *_{\text{poly}})$ è un campo

se e solo se $f(x)$ è irriducibile

(Inversi moltiplicativi calcolabili con una modifica
dell'algoritmo di Euclide - Esteso per gli interi)

"irriducibilità gioca il ruolo della primalità"

Proviamo a costruire il campo con 2^3 elementi
 $\mathbb{Z}_2 = \{0, 1\}$, operazioni sui coefficienti modulo?

$$f(x) = x^3 + x + 1 \quad (\text{irriducibile})$$

Polinomi del campo : $0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1$

Esempio di moltiplicazione

$$(x^2+1)(x^2+x+1) = x^4 + x^3 + x + 1 \quad (\text{diviso per } x^3+x+1 \text{ da})$$

$$(x^4 + x^3 + x + 1) = (x+1)(x^3+x+1) + \frac{(x^2+x)}{x(x)}$$

Quindi in $\mathbb{Z}_2[x]/(x^3+x+1)$, $(x^2+1)(x^2+x+1) = x^2+x$

Elementi del campo rappresentabili come tuple di coefficienti in \mathbb{Z}_p , cioè $(a_{n-1}, \dots, a_1, a_0)$

\downarrow
 \mathbb{Z}_2 \longrightarrow sequenza binaria

Usando questa rappresentazione la tabellina
della moltiplicazione diretta

	001	010	011	100	101	110	111
001	001	010	011	100	101	110	111
010	010	100	110	011	001	111	101
011	011	110	101	111	100	001	010
100	100	011	111	110	010	101	001
101	101	001	100	010	111	011	110
110	110	111	001	101	011	010	100
111	111	101	010	001	110	100	011

$$(x^2+1)(x^2+x+1)$$



$$(101) \cdot (111) = 110$$

Similmente costruiamo
la tabellina per la somma

Si può dimostrare che \exists almeno un polinomio
irriducibile per ogni grado n in $\mathbb{Z}_p[x]$.

$\Rightarrow \exists$ un campo finito con p^n elementi
per tutti i primi p , $n \geq 1$

(solitamente, per ogni grado, esistono diversi pol. irriducibili)

Si può dimostrare che i campi costituiti a partire da
quei polinomi irriducibili diversi risultano isomorfi

$\Rightarrow \exists !$ campo con p^n elementi $(F_{p^n} \circ FF(p^n))$

Talors Field

Sottogruppi di campi finiti
Il problema DL è inteso difficile anche nel gruppo
multiplicativo di un campo finito con caratteristica
grande.

$F_{p^n}^*$ è un gruppo ciclico di ordine $p^n - 1$

Se q è un fattore primo grande di $p^n - 1$
allora $F_{p^n}^*$ ha un sottogruppo ciclico di
ordine q .

... altra salta di gruppi di ordine primo in
cui DL e CDT sono intente valide