Problemi difficili e assunzioni crittografiche

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

$\text{Fact}_{A, \text{GenMod}}(n)$

1. C esegue $\text{GenMod}(1^n)$ per ottenere (p, q, N)
2. A riceve da C il modulo N e dà $p', q' > 1$
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 $negl(n)$:

$$P_2\left[Factor_{A, GenMod}(n) = 1\right] \leq negl(n)$$

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 soltanto fattori piccoli)

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

Quindi se $B = O((\log N)^c)$ — tempo poly, può succedere basso

se $B = O(\sqrt{N})$ — tempo exp, può succedere alto

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

Idea = cerca di calcolare un x tale che $MCD(x, N) = p$

i fattori piccoli di p-1 servono per calcolare x

- Pollard's Rho (general purpose)

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

$$MCD(x - x', N) = p$$

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

Complessità $O(2^{n/4})$

↓

migliora rispetto alla ricerca esaustiva

- Quadratic Sieve (general purpose)

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

Cerca di trovare x e y tali che $x^2 \equiv y^2 \bmod N$

con $x \neq \pm y$ per calcolare

$$p = MCD(x-y, N)$$

Complessità $O\left(2^{\sqrt{n \log n}}\right)$

(Notate che $c \cdot \sqrt{n \log n}$ cresce meno di dn, sub-esponenziale)

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 e q di n bit

$N = p \cdot q$ e due interi e, d maggiori di zero tali che

$$MCD(e, \varphi(N)) = 1 \qquad ed \equiv 1 \bmod \varphi(N)$$

$RSA\text{-}inv_{A, GenRSA}(n)$

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 $x^e \equiv y \bmod N$, C dà in output 1; altrimenti, 0;

Esempio

$GenMod(1^n) \rightarrow (N, p, q) = (143, 11, 13)$

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

Scegliamo $e = 7$. Usando Extend_Euclid, $d = 103$

$(7 \cdot 103 \equiv 1 \bmod 120)$

GenRSA dà in output $(143, 7, 103)$

$\downarrow \quad \downarrow \quad \downarrow$

N, e, d

Scelta di e popolare: $e = 3$

si scelgono p e q tali che $p, q \neq 1 \bmod 3$

$\Rightarrow \gcd(3, \varphi(n)) = 1$

Questa scelta è soggetta ad attacchi per esp. piccoli

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

10 01

Leggermente più costosa di $e = 3$

Attenzione: d piccolo è una cattiva idea

Possono essere applicati attacchi di forza bruta.

Anche se $d \geq N^{1/4}$ esistono

algoritmi noti per recuperare d da N ed e

Problema discreto di $\mathbb{Z}_N^*$

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

f_d è la permutazione inversa

Consiste nel calcolare la radice e-esima di y scelto a caso in $\mathbb{Z}_N^*$

Conoscendo d è facile. Il problema richiede di farlo senza

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

$$P_2\left[\text{RSA-inv}_{A,\,\text{Gen}_{RSA}}(n) = 1\right] \leq \text{negl}(n)$$

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

Posso fattorizzare? Dati p e q, calcolo $\varphi(N)$, calcolo $d = e^{-1} \bmod \varphi(N)$, inverto!

Pertanto, indicando con Gen_{Mod} il gen in Gen_{RSA}, affinché il prob RSA sia difficile relativamente a Gen_{RSA}, il prob. della fattorizzazione deve essere difficile relativamente a Gen_{Mod}.

RSA non può essere più difficile di Factoring

Factoring difficile $\Rightarrow$ RSA difficile ?

NON lo sappiamo, grossa questione aperta

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

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

Teorema (Fattorizzazione di N dato d).

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

Idea: $x^2 \equiv 1 \bmod N$ ha 4 radici quadrate

Due sono quelle banali ± 1

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

$$MCD(x \pm 1, N)$$

Problema del logaritmo discreto

Sia $Gen_G()$ un alg ppt per la generazione di gruppi ciclici

$$(G, g, q)$$

- G: insieme elementi
- g: generatore
- q: ordine del gruppo

L'operazione $\oplus$ è efficientemente calcolabile

L'appartenenza di un el a G è efficientemente calcolabile

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

$$g^x = h$$

$$x = \log_g h$$

$Dlog_{A, GenG}(n)$

1. C esegue $GenG(1^n)$ e ottiene (G, g, q)
2. C sceglie in modo uniforme $h \in G$
3. A riceve (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 uniformemente a caso

Esempi

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

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

Def. Relativamente a $Gen_G()$, il problema DL è difficile se, $\forall$ A ppt, esiste una funzione trascurabile, tale che

$$P_2\left[Dlog_{A, Gen_G}(n) = 1\right] \leq negl(n)$$

Assunzione DL. Esiste un $Gen_G()$ 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

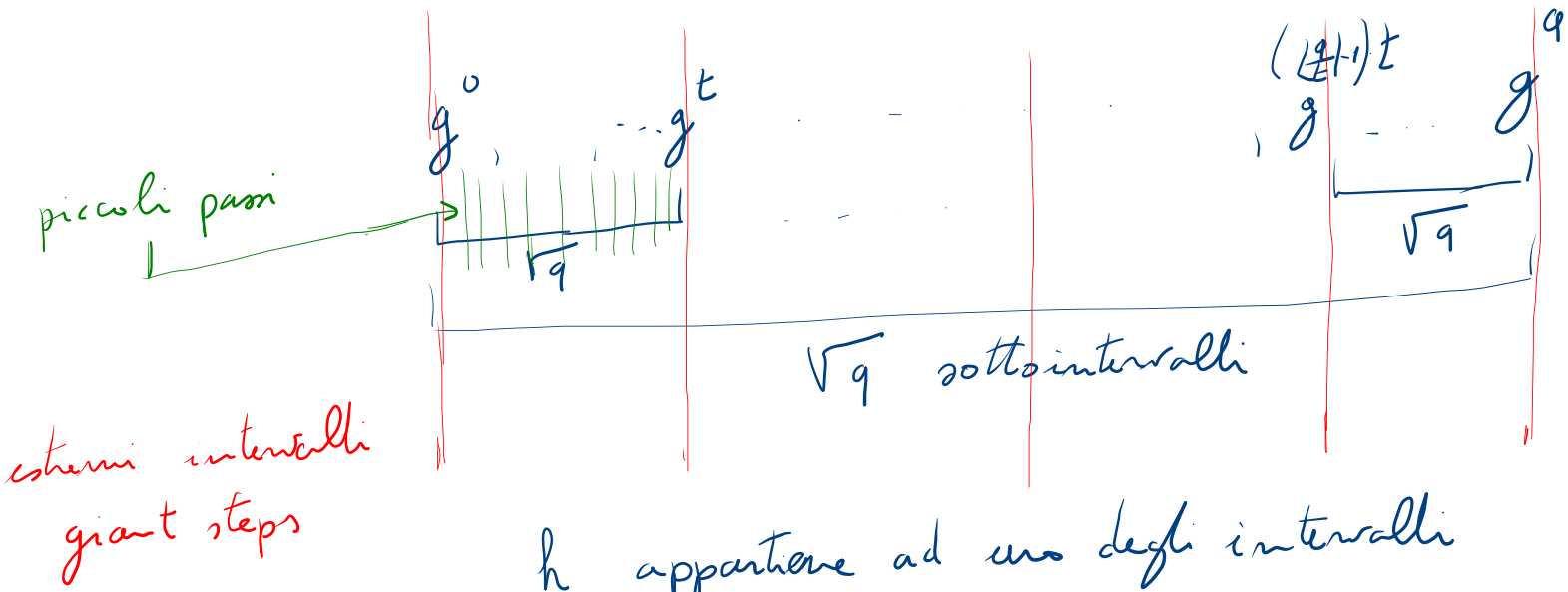
Complessità $\leq$ complessità di risoluzione nel sott. di ordine q' con q' max tra gli ordini dei sottogruppi.

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

- Baby step / giant step di Shanks

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

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



h appartiene ad uno degli intervalli

Calcolando $h \cdot g^1, h \cdot g^2, \ldots h \cdot g^t$ "shiftiamo" h fino a raggiungere l'esterno. Per cui, per qualche i

$$h \cdot g^i = g^{Kt} \Rightarrow x = Kt - i \bmod q$$

- Pollard's Rho

"Trade-off tempo/spazio"
rispetto all'alg. 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

Problemi Diffie-Hellman

$Gen_G()$ alg. ppt (G, g, q) Stesse ipotesi DL

$CDH_{A, Gen_G}(n)$

1. C esegue $Gen_G(1^n)$ 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 $h_1 = g^{x_1} \in G$, $h_2 = g^{x_2} \in G$
3. A riceve (G, g, q), h_1, h_2, e dà in output $h_3 \in G$
4. Se $h_3 = g^{x_1 x_2}$, allora C dà 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

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

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

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

$$P_2\left[CDH_{A, Gen_G}(n) = 1\right] \leq negl(n)$$

Assunzione (CDH). Esiste un algoritmo $Gen_G(1^n)$ 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) $\quad$ (h_1, h_2, h_3)

$\downarrow$ $\quad$ $\downarrow$

Diffie-Hellman $\quad$ unif. a caso

$DDH_{A, \mathrm{Gen}_G}(n)$

1. C esegue $\mathrm{Gen}_G(1^n)$ per ottenere (G, g, q)
2. C sceglie uniformemente 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 sceglie un bit b. Se $b = 1$, calcola $h_3 = g^{x_1 x_2}$

   Altrimenti, sceglie x_3 uniformemente a caso e calcola $h_3 = g^{x_3}$

4. A riceve (G, g, q) e (h_1, h_2, h_3) e dà a C b'.

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

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

$$Pr\left[DDH_{A, Gen_G}(n) = 1\right] \leq \frac{1}{2} + negl(n)$$

Assunzione (DDH). Esiste un algoritmo $Gen_G(1^n)$ 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 $x_1 = \log_g h_1$ 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 $h_3' \stackrel{?}{=} h_3$

CDH difficile $\overset{?}{\Rightarrow}$ 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 DDH è facile.

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

DDH si ritiene 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 ove richiesto calcolo 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

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

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

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

$(\mathbb{Q}, +, \cdot)$ razionali — $(\mathbb{R}, +, \cdot)$ reali — $(\mathbb{C}, +, \cdot)$ complessi — infiniti

finito → $(\mathbb{Z}_p, +_p, *_p)$ (p primo) → $(\mathbb{Z}_p, +_p)$ e $(\mathbb{Z}_p^*, *_p)$ e vale prop. distr.

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

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

Grado di f → $\deg(f)$ = esponente più grande termine di f.

Come per gli interi, in generale,

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

$\deg(f) = n$; $\deg(r) < n$

I polinomi $q(x)$ ed $r(x)$ 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 scriveremo

$$g(x) \equiv h(x) \bmod f(x)$$

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

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

$$\mathbb{Z}_p[x]/f(x)$$

(tutti i polinomi di grado al più $n-1$)

$$|\mathbb{Z}_p[x]/f(x)| = p^n$$

$$a_{n-1}x^{n-1} + \dots + a_1 x + a_0$$

$a_{n-1} \to \mathbb{Z}_p$; $a_1 \to \mathbb{Z}_p$; $a_0 \to \mathbb{Z}_p$

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

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

se e solo se $f(x)$ è irriducibile

(Inverso moltiplicativo calcolabile 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 ai coefficienti modulo 2

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

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

Esempio di moltiplicazione

$$(x^2+1)(x^2+x+1) = x^4 + x^3 + x + 1 \qquad \text{(divido per } x^3+x+1 \text{ dà)}$$

$$(x^4 + x^3 + x + 1) = \underbrace{(x+1)}_{q(x)}\underbrace{(x^3+x+1)}_{f(x)} + \underbrace{(x^2+x)}_{r(x)}$$

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 tupla di coefficienti in $\mathbb{Z}_p$, cioè $(a_{m-1}, \dots, a_1, a_0)$

$\downarrow$

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

Usando questa rappresentazione la tabellina della moltiplicazione diventa

	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

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

$\updownarrow$

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

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 costruiti a partire da due polinomi irriducibili diversi risultano isomorfi

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

↙ Galois Field

Sottogruppi di campi finiti

Il problema DL è ritenuto difficile anche nel gruppo moltiplicativo 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 scelta di gruppi di ordine primo in cui DL e CDH sono ritenute valide