ALGORITHM 9.34
Generating a random prime

Input: Length n
Output: A uniform n-bit prime

for $i = 1$ to $3n^2$:
  $p' \leftarrow \{0,1\}^{n-1}$
  $p := 1 \| p'$
  run the Miller–Rabin test on input p and parameter 1^n
  if the output is "prime," **return** p
return fail

THEOREM 9.32 *For any $n > 1$, the fraction of n-bit integers that are prime is at least $1/3n$.*

$$\left(1 - \frac{1}{3n}\right)^t = \left(\left(1 - \frac{1}{3n}\right)^{3n}\right)^n \leq \left(e^{-1}\right)^n = e^{-n}$$

Assunzione della Fattorizzazione $\Longrightarrow$ Una funzione one-way

$Gen(1^n) \longrightarrow (N, p, q)$, $N = p \cdot q$

p, q: primi di n bit (esatto con prob trasc.)

Gen: algoritmo ppt

↳ usa un numero polinomiale di bit casuali

Idea: possiamo definire una funzione f_{Gen} che usa il suo input x come "random bits" per l'esecuzione di $Gen(1^n)$. Modifichiamo allora $Gen(1^n)$ portando la randomness fuori, i.e., $Gen(1^n, x)$

Algoritmo per calcolare f_{Gen}

Input: stringa x di n bit

Output: intero N di n bit

Calcola $(N, p, q) \leftarrow Gen(1^n, x)$

random bit fissati forniti come input a Gen

return N

f_{Gen} è facile da calcolare poly time

$f_{Gen} : \{0,1\}^n \to \{0,1\}^n$ risulta one-way osservando che

- i moduli N restituiti da $f_{Gen}(x)$ con $x \in \{0,1\}^n$ uniforme
- i moduli N restituiti da $Gen(1^n)$

sono distribuiti identicamente.

Pertanto ...

Se i moduli generati da $Gen(1^n)$ sono difficili da fattorizzare, così risultano quelli generati da $f_{Gen}(x)$.

Discende che, trovare una qualsiasi preimmagine x' di N rispetto a f_{Gen}, cioè tale che

$$f_{Gen}(x') = f_{Gen}(x) = N,$$

deve essere difficile. Altrimenti, eseguendo

$$\underset{\substack{\uparrow \\ \text{poly time}}}{Gen(1^n, x')} \longrightarrow (N, \underbrace{p, q}_{\text{fattorizzazione di } N})$$

si otterrebbe la fattorizzazione di N.

L'assunzione BSA $\Rightarrow$ Una famiglia di permutazioni one-way

$\Pi = (Gen, Samp, f)$

1. $Gen(1^n) \to I : |I| \geq n$

   $f_I : D_I \to D_I$

2. Samp : $x \leftarrow_R D_I$

3. L'alg di valutazione di f, su input I ed x, dà

   $y = f_I(x)$

1. Gen : su input 1^n, esegui

   $GenBSA(1^n) \to (N, e, d)$

   e dai in output

   $I = (N, e)$, $D_I = \mathbb{Z}_N^*$

2. Samp : su input $I = (N, e)$ scegli unif $x \in \mathbb{Z}_N^*$ ($x \in \mathbb{Z}_N \setminus \{0\}$ fino a quando $\gcd(x, N) = 1$)

3. f : su input $I = (N, e)$ ed $x \in \mathbb{Z}_N^*$ dai in output

   $x^e \bmod N$

È immediato constatare che, se il problema RSA è difficile relativamente a GenRSA, allora la famiglia di permutazioni proposta è one-way

Procedendo in modo simile si può costruire una famiglia di permutazioni one-way sulla difficoltà del problema DL in $\mathbb{Z}_p^*$

primo

$Gen(1^n) \rightarrow (p, g, p-1)$

$Samp \rightarrow x \in_R \mathbb{Z}_p^*$ $\qquad \{1, \ldots, p-1\}$

$f : g^x \bmod p$ $\qquad \{g^1 \ldots g^{p-1}\} \equiv \{1, \ldots, p-1\}$

Funzioni hash resistenti a collisioni

$$\mathcal{G}(1^n) \longrightarrow (G, q, g)$$

$\mathcal{G}$: algoritmo ppt generazione gruppi

G: gruppo ciclico

q: $|q| = n$ primo (ordine di G)

g: generatore

(Gen, H) funzione hash a lunghezza fissata

Gen: su input 1^n, esegui $\mathcal{G}(1^n) \Rightarrow (G, q, g)$, seleziona $h \in G$ in modo uniforme e dà in output $s = (G, q, g, h)$

H: data una chiave $s = (G, q, g, h)$ e un input $(x_1, x_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$ dà in output

$$H^s(x_1, x_2) = g^{x_1} \cdot h^{x_2} \in G$$

Note

G ed H possono essere calcolate in tempo polinomiale

H^s prende in input una stringa di $2 \cdot (n-1)$ bit

Se gli elementi del gruppo G possono essere rappresentati con meno di $2 \cdot (n-1)$ bit, allora H^s comprime

$$H^s : \{0,1\}^{2\cdot(n-1)} \longrightarrow \{0,1\}^{\ell} \qquad \ell < 2(n-1$$

Th: Se il problema DL è difficile relativamente a $\mathcal{G}$, allora H è una funzione hash resistente a collisioni di lunghezza fissata.

Dim. Sia A ppt e sia

$$\Pr[\text{Hash-coll}_{A,\Pi}(n) = 1] = \varepsilon(n)$$

Possiamo usare A per costruire A' di risolvere il problema DL con prob di successo $\varepsilon(n)$.

Algoritmo A'

Riceve in input $G, q, g, \underline{\underline{h}}$ ← (target per il DLP)

1. Sia $s = \langle G, q, g, h \rangle$ ← (seme per H)

Esegui $A(s)$ ed ottieni x ed x' (collisione)

2. Se $x \neq x'$ e $H^s(x) = H^s(x')$ ←

(a) se $h = 1$, return 0

(b) altrimenti ($h \neq 1$), interpreta

x come (x_1, x_2), $x_1, x_2 \in \mathbb{Z}_q$

x' come (x'_1, x'_2) $x'_1, x'_2 \in \mathbb{Z}_q$

e return

$$[(x_1 - x'_1)(x'_2 - x_2)^{-1} \bmod q] \leftarrow \log_g h$$

Osservazioni: A' esegue in tempo polinomiale

Il valore s dato ad A è distribuito esattamente come nell'esperimento $\text{Hash-coll}_{A,\Pi}(n)$, per lo stesso valore del parametro di sicurezza n

$\Rightarrow$ con prob. $\varepsilon(n)$ viene prodotta una collisione

$$H^s(x_1, x_2) = H^s(x_1', x_2') \iff g^{x_1} h^{x_2} = g^{x_1'} h^{x_2'}$$

$$\iff \left(g^{x_1} \cdot h^{x_2}\right) \cdot g^{-x_1'} \cdot h^{-x_2} = \left(g^{x_1'} h^{x_2'}\right) \cdot g^{-x_1'} h^{-x_2}$$

$$\iff \qquad g^{x_1 - x_1'} = h^{x_2' - x_2}$$

Nota che $x_2' - x_2 \neq 0 \bmod q$, altrimenti sarebbe

$$g^{x_1 - x_1'} = h^0 = 1 = g^0 \implies x_1 = x_1' \text{ e } x_2 = x_2'$$

Poiché q è primo, esiste $(x_2' - x_2)^{-1} \bmod q$

$$\implies g^{(x_1 - x_1')(x_2' - x_2)^{-1}} = h^{(x_2' - x_2)(x_2' - x_2)^{-1}} = h^1 = h$$

$$\implies \log_g h = (x_1 - x_1')(x_2' - x_2)^{-1}$$

Pertanto, A' risolve il problema DL

con probabilità esattamente $\varepsilon(n)$

Poiché, per assunzione, il problema è difficile

relativamente a $\mathcal{G}(1^n)$, allora $\varepsilon(n)$ è trascurabile

$$\Downarrow$$

H é collision resistant!

Cifratura a chiave privata

Codici per l'autenticazione

Permutazioni Pseudocasuali

Cifrari a blocchi

Generatori Pseudocasuali

Nella pratica ...

Funzioni one-way

In teoria ...

Fattorizzazione, RSA

Logaritmo discreto

Funzioni e famiglie di permutazioni one-way