

CRITTOGRAFIA

lezione del
29/9/2020



Sorgente binaria casuale

genera una sequenza di bit

1) $P(0) = P(1) = \frac{1}{2}$

(si può indossare richiedendo $P(0) > 0$, $P(1) > 0$,
e immutabili durante il processo)

2) la generazione di un bit è indipendente da quella degli altri bit.

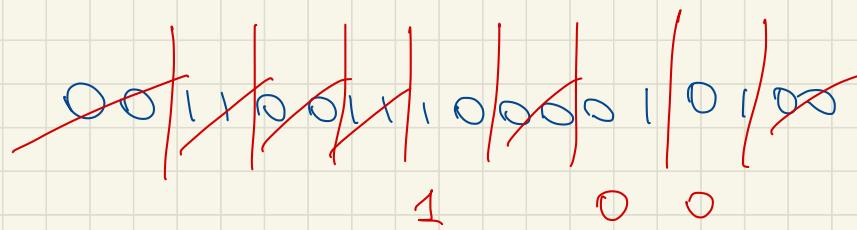
→ non si può prevedere il valore di un bit
osservando quelli già generati

Esempio

$$P(0) > P(1)$$

Si genera la sequenza:

$$\begin{array}{l} 01 \rightarrow 0 \\ 10 \rightarrow 1 \end{array}$$



Generazione di sequenze brevi

1) fenomeni casuali presenti in natura

↳ sorgenti di casualità

2) processi software

3) si genera la casualità mediante un algoritmo, cercando
all'interno di processi matematici

↳ Generatori di numeri
pseudo-casuali

es. rand() del linguaggio C

genera interi da 0 e $2^{16}-1$

Sono maggiormente brevi

Generatore \rightarrow "Ampificatore di casualità"

INPUT: Semse (sequenza o valore breve)

OUTPUT: flusso di bit arbitrariamente lungo, periodico

il suo interno contiene una ~~sottosequenza~~ sottosequenza che

si ripete \rightarrow Periodo

viene usata come sequenza casuale

un generatore è tanto migliore quanto + lungo è il periodo

S : #bit nel semse

n : lunghezza delle sequenze ottenuta col generatore

$$n \gg S$$

sequenze diverse : $2^S \ll 2^n$

\rightarrow Una sequenza \neq per ogni frame.
 2^S possibili semi

ESEMPIO

Generatore lineare

$$x_i = (a x_{i-1} + b) \bmod m$$

a, b, m interi positivi

seme: valore intero iniziale x_0 (casuale)

$$x_0, x_1 = (ax_0 + b) \bmod m, x_2 = (ax_1 + b) \bmod m, \dots$$

quando $x_i = x_0$, la sequenza si ripete

il periodo massimo è m

$$\text{MCD}(b, m) = 1$$

($a-1$) deve essere divisibile per ogni fattore primo di m

($a-1$) deve essere un multiplo di 4 se Ora m lo è

il generatore
produce una
permutazione
degli interi
 $[0, m-1]$

$$\alpha = 3141592653$$

π

$$a = 7$$

$$b = 7$$

$$m = 9$$

$$x_0 = 3$$

$$b = 2718281829$$

e

$$M = 2^{32}$$

$$x_i = (7x_{i-1} + 7) \bmod 9$$

~~Se~~ $x_0, 3, 1, 5, 6, 4, 8, \dots \rightarrow \underline{3}$

$x_0 \quad x_1 \quad x_2 \quad x_3$

→ trasformarlo in generatore binario:

$$\frac{x_i}{m} \rightarrow \underbrace{\text{punto}}_{\downarrow} \text{ del primo decimale}$$

PARITA'

Test statistici

valutazione delle sequenze prodotte da un generatore
pseudocasuale.

si voluto se la sequenza presenta le proprietà tipiche di una
sequenza casuale

- test di frequenza
 - poker test
 - test di autocorrelazione
 - run test
-

Per le applicazioni crittografiche si riduce a solo il

TEST di PROSSIMO BIT (implica i 4 test statistici preceduti)

↳ impossibile fare previsioni sugli elementi della
sequenza prima che siano generati

Test di prossimo bit

Un generatore binario supera il test di prossimo bit se
 \nexists algoritmo polinomiale in grado di precedere l' $(i+1)$ -esimo
bit della sequenza ~~a partire~~ a partire dalla conoscenza
degli i -bit precedentemente generati, con probabilità $> \frac{1}{2}$



→ Un generatore è CRUTICAMENTE SICURO se
supera il test di prossimo bit

Generatore polinomiale

(non è crittograficamente sicuro)

$$x_i = (Q_1 x_{i-1}^t + Q_2 x_{i-1}^{t+1} + \dots + Q_t x_{i-1} + Q_{t+1}) \bmod m$$

$$r_i = \frac{x_i}{m}$$

prima cifra | poi $\rightarrow 0$
prima cifra disponibile $\rightarrow 1$
↓
decimale

Costruzione di generazioni ortogonali sicuri con
funzioni one-way

→ computazionalmente facile calcolare $y = f(x)$

$$x \rightarrow y = f(x) \quad \text{tempo polinomiale}$$

→ computazionalmente difficile da invertire

$$y \rightarrow x = f^{-1}(y) \quad \begin{array}{l} \text{conosciamo solo algoritmi di caccia} \\ \text{in tempo esponenziale} \end{array}$$

$x : y = f(x)$

→ IDEA f : funzione one-way

$$\begin{array}{ccccccc} x & f(x) & f(f(x)) = f^2(x) & f^{(2)}(x) & f^{(i)}(x) = f(x_{i-1}) \\ x_0 & x_1 & x_2 & \dots & \dots & - & x_i \end{array}$$

si itera l'applicazione della funzione one-way un numero
arbitrario di volte

→ ogni elemento della sequenza S si può calcolare facilmente dal precedente, ma non dai valori successivi perché f è one-way

$$x_i \rightarrow x_{i+1}$$

$$\begin{aligned} x_{i+1} &= f(x_i) && \text{tempo polinomiale} \\ &= f(f^{(i)}(x)) = f^{(i+1)}(x) \end{aligned}$$

$$x_{i+1} \rightarrow x_i$$

dificile

$$x_i = f^{-1}(x_{i+1})$$



si calcola la sequenza S per un certo numero di passi, senza sapere il risultato, e si comunicano gli elementi in ordine inverso

→ ogni elemento non è prevedibile in tempo polinomiale, per conoscendo quelli comunicati in precedenza

Generatori BINARI crittograficamente sicuri: si usano

"Predicati HARD CORE" delle funzioni one-way

$b(x)$ è un predicato H.C. di una funzione one-way $f(x)$

se:

$b(x)$ è facile da calcolare conoscendo x

$b(x)$ è difficile da prevedere con probabilità $> \frac{1}{2}$
conoscendo solo $f(x)$

ESEMPIO

funzione one-way

$$x \rightarrow f(x) = x^2 \bmod n$$

n non
primo

$$n = 77 \quad x = 10 \quad f(x) \rightarrow y = 10^2 \bmod 77 = 23$$

$b(x) = "x \text{ è disponibile}" \rightarrow \text{predicato hard core}$

Generatore BBS (crittograficamente sicuro)

↓
Manuel Blum Blum Shub 1986
↳ Premio Turing 1995

$n = p \times q$ p e q primi (grandi)

$$p \bmod 4 = 3$$

$$q \bmod 4 = 3$$

$$2 \lfloor \frac{p}{4} \rfloor + 1 \quad e \quad 2 \lfloor \frac{q}{4} \rfloor + 1$$

primi ha lato

→ y coprimo con n

si calcola

$$\overbrace{x_0 = y^2 \bmod n}^{\downarrow}$$

si usa come seme

→ si calcola una successione di $m \leq n$ intui



$$\bullet \quad x_i = (x_{i-1})^2 \bmod n$$

$$\bullet \quad b_i = 1 \iff x_{m-i} \text{ est disponi}$$

$$b_1 = 1 \iff x_{m-1} \text{ est disponi}$$

$$b_0 = 1 \iff x_m \text{ est disponi}$$

$$x_0 \rightarrow b_m$$

$$\begin{array}{ccccccccc} x_0 & \rightarrow & x_1 & \dots & x_2 & - & \dots & x_{m-1} & x_m \\ b_m & & \downarrow & & b_{m-1} & & & b_1 & b_0 \end{array}$$

si rechweise la sequence in ordine inverso:

b₀ b₁ b₂ . . . -