

Numeri casuali

Avere numeri casuali è importante in molte applicazioni

- i numeri devono apparire casuali
 - dopo aver visto una sequenza di numeri generati
 - non deve essere possibile in tempo polinomiale indovinare un singolo numero (anche in modo probabilistico)

Nota: numeri casuali sono richiesti spesso e la loro generazione può essere costosa

Generazione numeri casuali

- uso di fonti di sistema
- uso di informazione esterne
- generatori pseudocasuali
 - dato un seme iniziale (casuale)
 - generano automaticamente una sequenza

Non deve essere possibile indovinare un numero della sequenza in tempo polinomiale (anche in modo probabilistico)

Informazioni di sistema

si possono utilizzare diverse informazioni

- tempo sistema
- spazio libero su dischi
- numero file su disco
- stato memoria centrale
- task del Sistemaoperativo (code in I/O, informazioni nei buffer)
- informazioni definite da utente (es. dim e numero finestre)
- variabili di sistema (es. tempo di clock)

Informazioni esterne

- tempi di battitura
- movimenti del mouse
- tempo arrivo pacchetti in rete

Informazioni esterne e di sistema

in generale forniscono pochi bit casuali (alcune informazioni possono essere facilmente dedotte)

Esempio

- clock di sistema: ha senso usare solo i digit meno significativi (es. millesimi di secondo);
- grandezze maggiori (giorno, ora, minuti) possono essere facilmente dedotte

Mescolamento di sorgenti casuali con tecniche crittografiche (es. uso funzioni hash)

Alcuni errori gravi

problemi

- uso generatori iniziali piccoli
 - es. 16 bit forniscono solo 65536 possibilità facilmente verificabili per ricerca esaustiva
- granularità dell'ora (in alcuni casi la granularità è molto alta e, quindi, facilmente prevedibile)
- pubblicare valori hash del giorno, ora
 - es : un'applicazione usava hash del giorno e poi mostrava l'ora in un header pubblico!!!

netscape

generazione seme

si calcola x come funzione di

- pid -process identifier
- ppid parent process ID (shift di 12 bit)
- ora del giorno (secondi e microsecondi)

$\text{seme} = \text{MD5}(x)$

MD5 funzione hash considerata sicura

netscape

calcolo stringa casuale

$N = \text{MD5}(\text{seme})$

$\text{seme} = \text{seme} + 1$

return N

il calcolo viene effettuato da netscape per calcolare tutti i parametri casuali necessari (chiavi segrete, nonce)

netscape

problemi

- pid, ppid sono lunghi di solito 15 bit (in molte applicazioni sono prevedibili)
- tenuto conto dello shift di 12 posizioni di ppid otteniamo 27 bit
- microsecondi è lungo 20 bit
- secondi sono facilmente predicibili con piccolo errore

Conclusione: poca casualità- piccolo numero di bit casuali nel seme

Generatore crittograficamente forte

un generatore pseudocasuale è crittograficamente forte se passa uno dei seguenti test (equivalenti)

test prossimo bit

- nessun algoritmo polinomiale dopo aver visto una sequenza di numeri generati è in grado di indovinare con prob. $>1/2$ il prossimo bit

test statistici

- nessun algoritmo polinomiale è in grado di distinguere con prob. $>1/2$ una sequenza casuale da una pseudocasuale

Generatore crittografici

- cifratura di un contatore
seme s , valore iniziale c , funzione crittog. C
 $x_i = \text{Codifica con chiave } s \text{ di } c$
 $c = c + 1$
- generatore con RSA: numeri primi p, q , $n = pq$, e
intero tale che $\text{MCD}(e, (n)) = 1$, $z = 0$
 $z_i = z_{(i-1)}^{*e} \text{ mod } n$
 $i = i + 1$
output (bit meno significativo di z_i)

Generatore crittografici: X9.17

generatore con 3DES (triplo DES)

- input: seme s di 64 bit, intero m , chiave triplo DES, Drappresentazione data e ora
 - output: x_1, x_2, \dots, x_m sequenza di m stringhe da 64 bit
1. calcola $I = 3DES(D)$
 2. for $i = 1$ to m do
 - $x_i = 3DES(I \text{ exor } s)$
 - $s = 3DES(x_i \text{ exor } I)$
 3. return x_1, x_2, \dots, x_m

Esercizio

- Generazione con funzione hash
- seme iniziale (casuale)

$y = \text{seme}$

for $i=1$ to n do

$y = \text{Hash}(y)$

Perché non va bene?