

Casualità e Pseudocasualità

Corso di Sicurezza
a.a. 2019-20

Alfredo De Santis

Dipartimento di Informatica
Università di Salerno

ads@unisa.it

<http://www.di-srv.unisa.it/~ads>



Marzo 2020

Utilità "casualità"

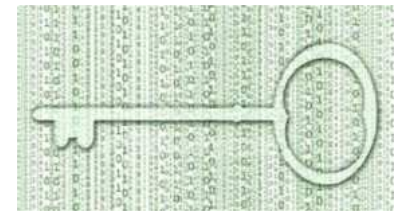
- Simulazione numerica
- Analisi numerica
- Decision making
- Testing computer chips
- Algoritmi probabilistici

➤ ...

➤ Sicurezza dati

➤ Generazione chiavi, nonce, challenge

➤ ...

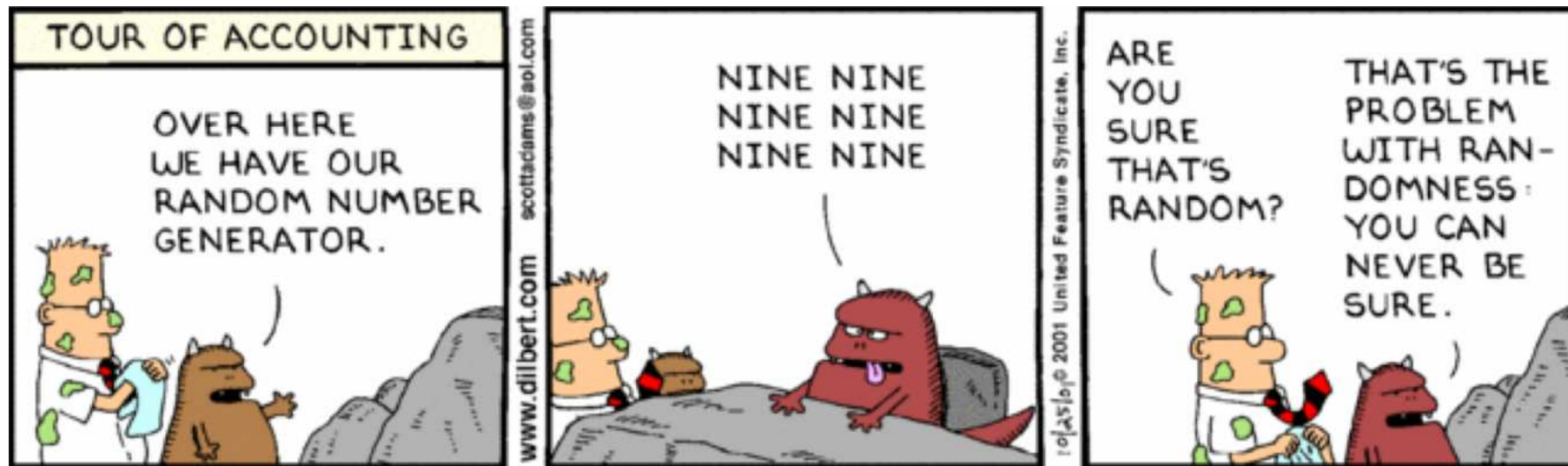


Generazione numeri casuali

```
int getRandomNumber()  
{  
    return 4; // chosen by fair dice roll.  
              // guaranteed to be random.  
}
```

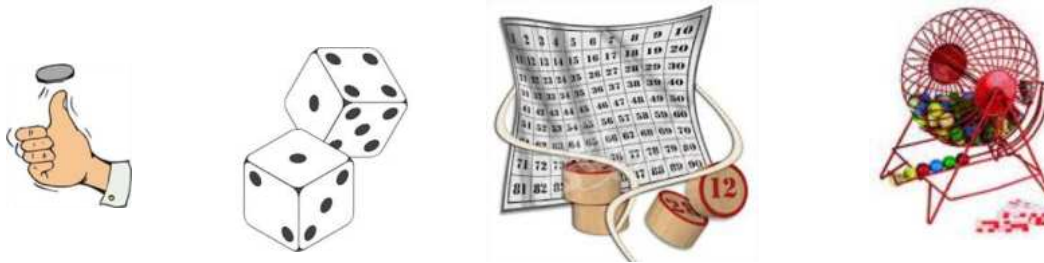
<http://xkcd.com/221/>

Generazione numeri casuali



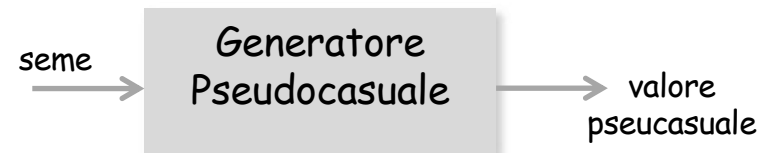
Casualità e pseudocasualità

➤ Casuale




➤ Pseudocasuale

Generazione deterministica da un seme iniziale
Sembra casuale ma non lo è



Randomness by obscurity

- Ian Goldberg e David Wagner, studenti di Berkeley, nel settembre 1995
- reverse-compilation della routine di Netscape 1.1 (implementava SSL)

- 
- Primo web browser di successo
 - Negli anni '90 il browser con utilizzo più elevato
 - Ultima versione (la 9) del 2008

Randomness by obscurity

- Ian Goldberg e David Wagner, studenti di Berkeley, nel settembre 1995
- reverse-compilation della routine di Netscape 1.1 (implementava SSL)
- time, pid, ppid mischiati erano il seme di un generatore per scegliere chiavi e challenge

Rand

- Ian Goldb
- Berkeley,
- reverse-c
- Netscape
- time, pid,
- generator

```
global variable seed;
```

```
RNG_CreateContext()
```

```
(seconds, microseconds) = time of day; /* Time elapsed since 1970 */
```

```
pid = process ID; ppid = parent process ID;
```

```
a = mklcpr(microseconds);
```

```
b = mklcpr(pid + seconds + (ppid << 12));
```

```
seed = MD5(a, b);
```

```
mklcpr(x) /* not cryptographically significant; shown for completeness */
```

```
return ((0xDEECE66D * x + 0x2BBB62DC) >> 1);
```

```
RNG_GenerateRandomBytes()
```

```
x = MD5(seed);
```

```
seed = seed + 1;
```

```
return x;
```

```
global variable challenge, secret_key;
```

```
create_key()
```

```
RNG_CreateContext();
```

```
tmp = RNG_GenerateRandomBytes();
```

```
tmp = RNG_GenerateRandomBytes();
```

```
challenge = RNG_GenerateRandomBytes();
```

```
secret_key = RNG_GenerateRandomBytes();
```

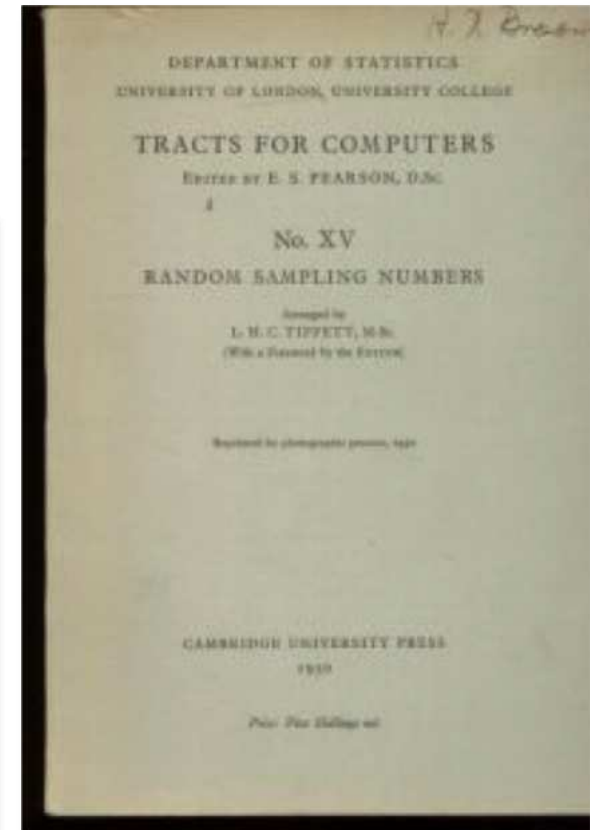
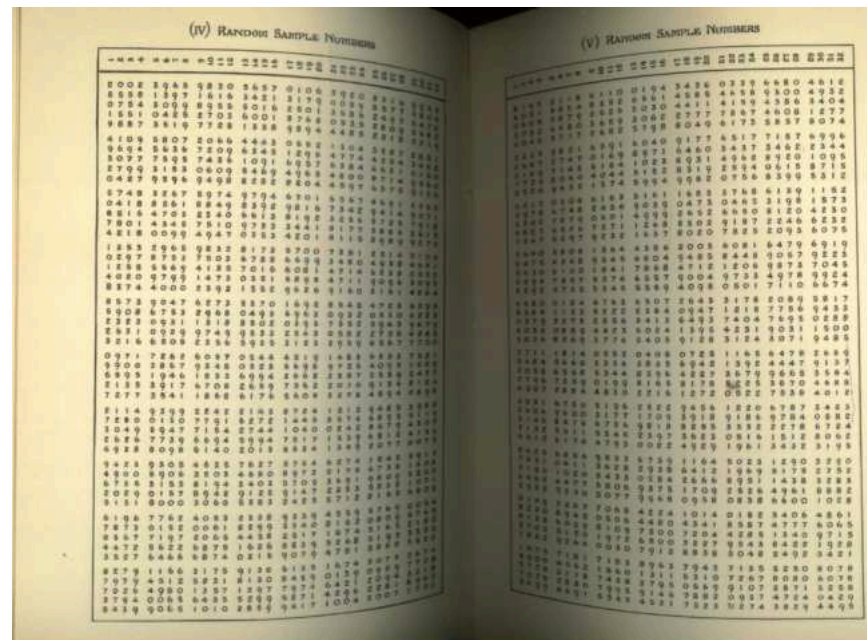

Indice

- Generazione casuale
- Generazione pseudocasuale

Tavole numeri casuali

L.H.C. Tippett, "Random Sampling Numbers", 1927

41.600 numeri casuali ottenuti prendendo le cifre centrali da misure di aree delle chiese inglesi



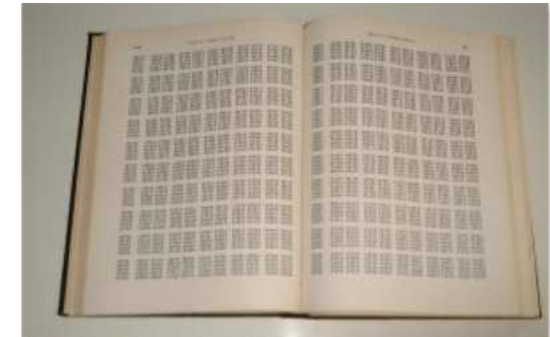
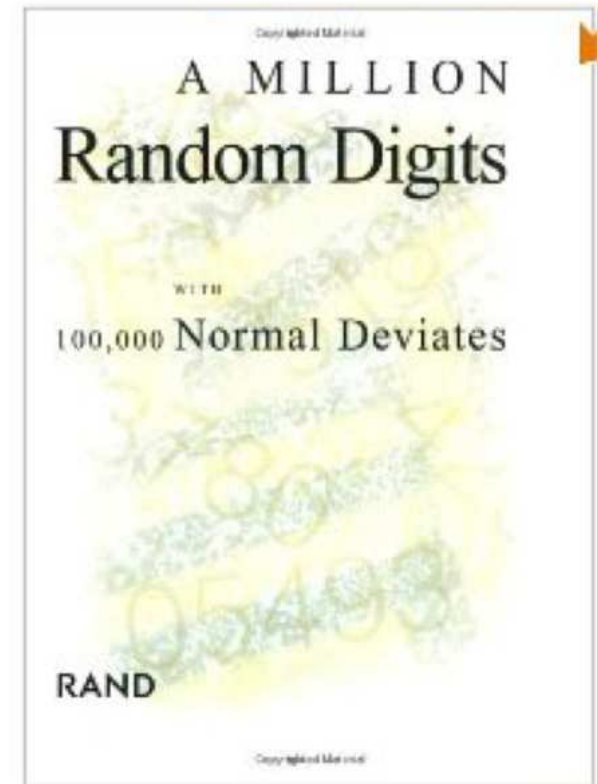
Ristampa 1950

A million random digits

1955, ristampato nel 2002

The random digits in the book were produced by rerandomization of a basic table generated by an electronic roulette wheel. Briefly, a random frequency pulse source, providing on the average about 100,000 pulses per second, was gated about once per second by a constant frequency pulse. Pulse standardization circuits passed the pulses through a 5-place binary counter. In principle the machine was a 32-place roulette wheel which made, on the average, about 3000 revolutions per trial and produced one number per second. A binary-to-decimal converter was used which converted 20 of the 32 numbers (the other twelve were discarded) and retained only the final digit of two-digit numbers; this final digit was fed into an IBM punch to produce finally a punched card table of random digits.

73735	45963	78134	63873
02965	58303	90708	20025
98859	23851	27965	62394
33666	62570	64775	78428
81666	26440	20422	05720
15838	47174	76866	14330
89793	34378	08730	56522
78155	22466	81978	57323
16381	66207	11698	99314
75002	80827	53867	37797
99982	27601	62686	44711
84543	87442	50033	14021
77757	54043	46176	42391
80871	32792	87989	72248
30500	28220	12444	71840

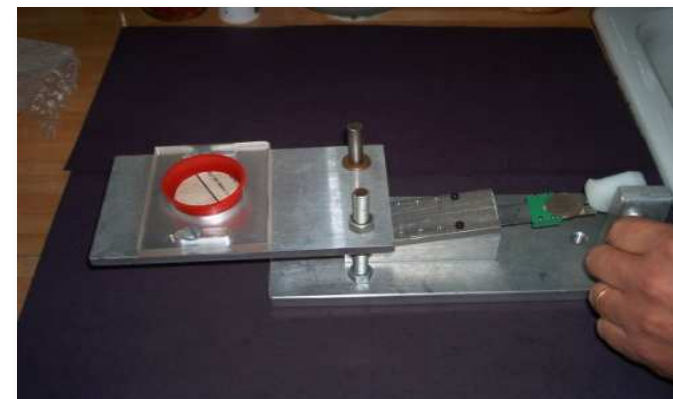


Lancio moneta

Probabilità testa o croce 50%-50%



Persi Diaconis, Susan Holmes , Richard Montgomery,
Dynamical bias in the coin toss,
SIAM Review, Volume 49, Issue 2, April 2007, pag 211-235
<http://statweb.stanford.edu/~susan/papers/headswithJ.pdf>



Lancio moneta

DYNAMICAL BIAS IN THE COIN TOSS

Persi Diaconis

Departments of Mathematics
and Statistics

Stanford University

Susan Holmes

Department of Statistics
Sequoia Hall

Stanford University

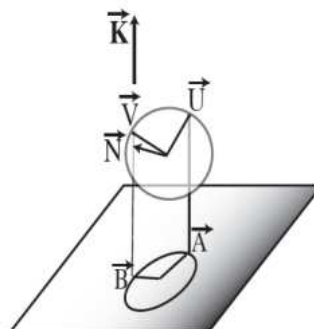
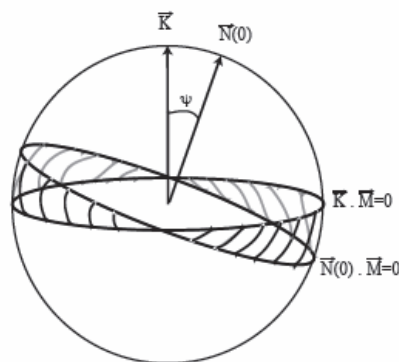
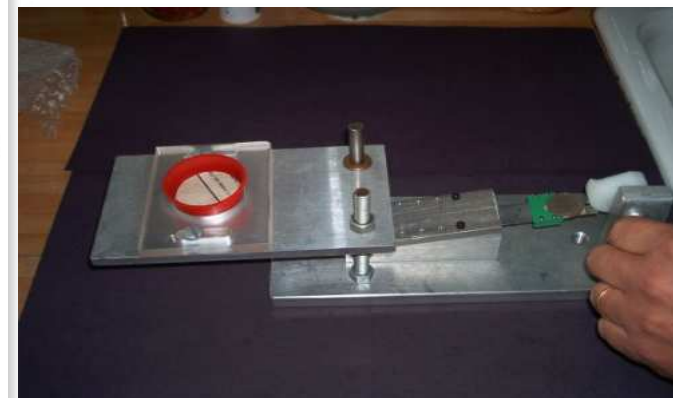
Richard Montgomery

Department of Mathematics
University of California

Santa Cruz

Abstract

We analyze the natural process of flipping a coin which is caught in the hand. We prove that vigorously-flipped coins are biased to come up the same way they started. The amount of bias depends on a single parameter, the angle between the normal to the coin and the angular momentum vector. Measurements of this parameter based on high-speed photography are reported. For natural flips, the chance of coming up as started is about .51.



$$\begin{aligned}
 d_v(X_\lambda Y_\lambda \bmod 2\pi, U) &\leq \int d_v(X_\lambda y \bmod 2\pi, U) P_{y_\lambda}(dy) \\
 &\leq P\{Y_\lambda \leq c\lambda\} + \int_{\{Y_\lambda \geq c\lambda\}} d_v(X_\lambda y \bmod 2\pi, U) P_{Y_\lambda}(dy) \\
 &\leq \frac{\gamma}{\lambda} + \int_{\{Y_\lambda \geq c\lambda\}} \frac{\pi}{4y} \int |p'_{X_\lambda}(x|Y_\lambda = y)| dx P_{Y_\lambda}(dy) \\
 &\leq \frac{\gamma}{\lambda} + \frac{\pi}{4c\lambda} A_\lambda.
 \end{aligned}$$

Lancio moneta

- If the coin is tossed and caught, it has about a 51% chance of landing on the same face it was launched. (If it starts out as heads, there's a 51% chance it will end as heads).
- If the coin is spun, rather than tossed, it can have a much-larger-than-50% chance of ending with the heavier side down. Spun coins can exhibit "huge bias" (some spun coins will fall tails-up 80% of the time).
- If the coin is tossed and allowed to clatter to the floor, this probably adds randomness.
- The same initial coin-flipping conditions produce the same coin flip result. That is, there's a certain amount of determinism to the coin flip.
- ...

Lancio moneta

- Probabilità testa o croce 70%-30%
- Come ottenere evento 50%-50%



Lancio moneta

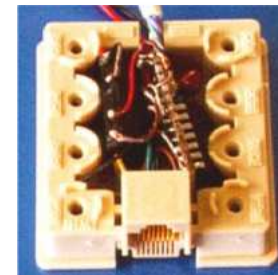
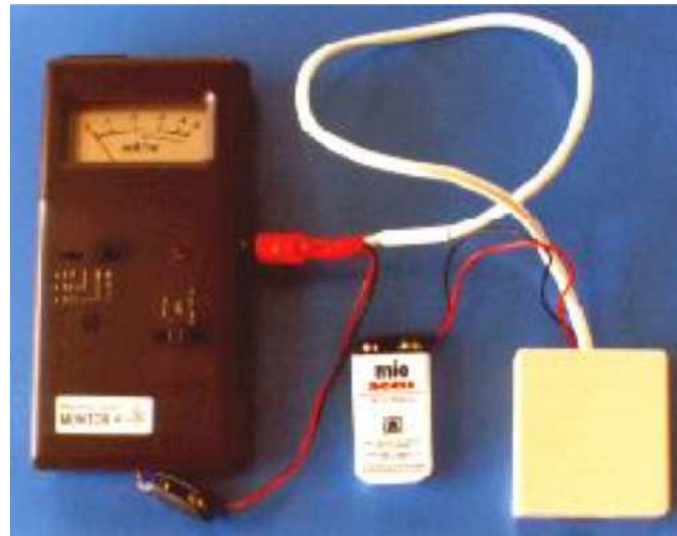
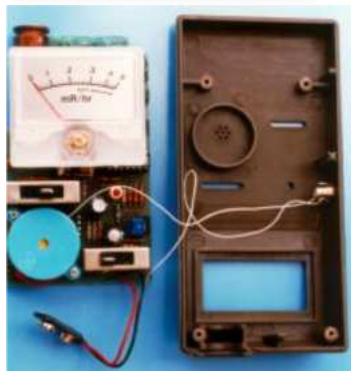
- Probabilità testa o croce 70%-30%
- Come ottenere evento 50%-50%
- Due lanci indipendenti della stessa moneta:
 - 00 ripetere due lanci
 - 11 ripetere due lanci
 - 01 risultato testa
 - 10 risultato croce



Radioactive decay

HotBits

- Numbers are generated by timing successive pairs of radioactive decays detected by a Geiger-Müller tube interfaced to a computer
- Velocità: circa 100 byte al secondo
- <http://www.fourmilab.ch/hotbits/> (1996, update 2006)



Lavarand e Lavarnd

Lavarand

- progettato da Silicon Graphics
- Patent 5.732.138: "Method for seeding a pseudo-random number generator with a cryptographic hash of a digitization of a chaotic system." 1996
- Immagini del movimento nelle lava lamp
 - Poi calcola hash



Lavarnd

- "noise" termico di webcam con lenti oscurate
 - Poi calcola hash
- Progetto open source



Air Turbulence in disk drives

- Velocità rotazione, ventola raffreddamento, spazio disco, causano turbolenze aria
- Tempi accesso lettura singoli settori non sono uguali
- Variazione tempi come sorgente di randomness

D. Davis, R. Ihaka, P.R. Fenstermacher,
Cryptographic Randomness from Air
Turbulence in Disk Drives, CRYPTO '94.

Cryptographic Randomness from Air Turbulence in Disk Drives

Don Davis,¹ Ross Ihaka,² and Philip Fenstermacher³ *

¹ Openvision Technologies, 1 Main St. Cambridge, MA 02142

² University of Auckland, Mathematics Dept, Auckland, NZ

³ 18A Forest St. Cambridge, MA 02138

Abstract. A computer disk drive's motor speed varies slightly but irregularly, principally because of air turbulence inside the disk's enclosure. The unpredictability of turbulence is well-understood mathematically; it reduces not to computational complexity, but to information losses. By timing disk accesses, a program can efficiently extract at least 100 independent, unbiased bits per minute, at no hardware cost. This paper has three parts: a mathematical argument tracing our RNG's randomness to a formal definition of turbulence's unpredictability, a novel use of the FFT as an unbiasing algorithm, and a "sanity check" data analysis.

Dispositivi hardware

"noise" generato da resistenza o da diodo

➤ <http://www.comscire.com/>



➤ <http://www.protego.se>



RANDOM.ORG

- atmospheric "noise"
- Startup fondata nel 1997
- Primo generatore basato su *noise* di una radio acquistata da Radio Shack a \$10
 - Codice C/C++ su Windows NT
- Ora configurazione distribuita
 - Nodi in diverse località geografiche generano randomness, eseguono test statistici e poi inviano ad un server
- <http://www.random.org>

RANDOM.ORG

Search RANDOM.ORG

Google™ Custom Search

Search

True Random Number Service

Do you own an iOS or Android device? [Check out our app!](#)

➤ a-
Bit Tally

➤ S RANDOM.ORG is a true random number service that generates randomness via atmospheric noise. This page shows how many [random bits](#) the service has generated throughout its existence. The figures are updated roughly once per minute.

➤ P
r
c

Service began on:	Thursday, 22 October 1998 at 19:52:27 UTC
Today's date:	Monday, 30 May 2016 at 20:06:15 UTC
Total time operational:	555,552,828 seconds
Total randomness generated:	1,288,466,395,294 bits (approx. 150.0 GiB)
Equivalent to:	236.3 CD-ROMs or 32.6 DVD-ROMs
Average speed:	2,319 bits/second†

➤ O †The average speed is calculated since the service began (i.e., since 1998). The figure does not reflect the generator's maximum speed but rather its effective speed, which is based on demand, over a long period.

➤ Nodi in diverse località geografiche generano randomness, eseguono test statistici e poi inviano ad un server

➤ <http://www.random.org>

True Random Number Generator

Min:

Max:

Result:

37

Powered by [RANDOM.ORG](#)

Altre sorgenti di casualità caratteristiche del sistema



- Variabili di sistema, come tempo di clock, oppure numeri di serie (come indirizzo Ethernet)
- Numero di file su dischi o in particolare directory
- Spazio libero su ogni disco
- Informazioni presenti nei buffer, nello code per l'I/O, nei driver per video
- Numero di task nella coda di schedulazione del sistema operativo, le loro ID, le loro grandezze
- Stato della memoria centrale
- Informazioi definite dall'utente, come grandezza e posizione delle finestre, colori utilizzati, nomi dei file

Altre sorgenti di casualità caratteristiche del sistema



- Variabili di sistema, come tempo di clock, oppure numeri di serie (come indirizzo Ethernet)
- Numero di file su dischi o in particolare directory
- Spazio libero su ogni disco
- Informazioni presenti nei buffer, nello code per l'I/O, nei driver per video
- Numero di task nella coda di schedulazione del sistema operativo, le loro ID, le loro grandezze
- Stato della memoria centrale
- Informazioi definite dall'utente, come grandezza e posizione delle finestre, colori utilizzati, nomi dei file

Pochi bit ... non molto segreti

Altre sorgenti di casualità eventi esterni

- contenuto delle battiture su tastiera
- intervalli di tempo tra la digitazione dei tasti
- misure di tempi e posizione dei movimenti del cursore e/o del mouse

Altre sorgenti di casualità eventi esterni

- contenuto delle battiture su tastiera
- intervalli di tempo tra la digitazione dei tasti
- misure di tempi e posizione dei movimenti del cursore e/o del mouse

informazione raccolta durante una sessione di lavoro
(altrimenti si puo' chiedere di battere tasti a caso
oppure di muovere il mouse per un po')

Altre sorgenti di casualità eventi esterni

- contenuto delle battiture su tastiera
- intervalli di tempo tra la digitazione dei tasti
- misure di tempi e posizione dei movimenti del cursore e/o del mouse

informazione raccolta durante una sessione di lavoro
(altrimenti si puo' chiedere di battere tasti a caso
oppure di muovere il mouse per un po')

- tempo di arrivo dei pacchetti su rete
- intervalli di tempo tra interrupt

Test statistici

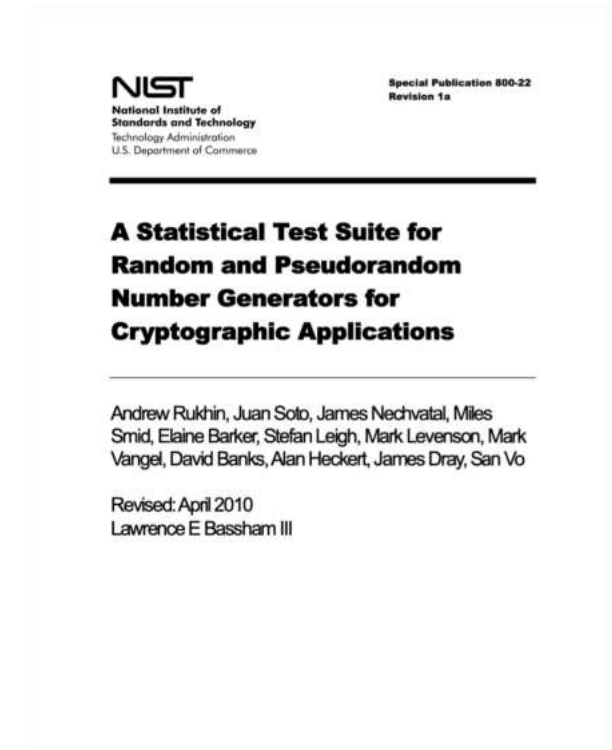
NIST SP 800-22 Rev. 1a.

A Statistical Test Suite for Random and Pseudorandom
Number Generators for Cryptographic Applications

Aprile 2010

16 test:

- Frequency (Monobits) Test
- Frequency Test within a Block
- Runs Test
- Test for the Longest Run of Ones in a Block
- Discrete Fourier Transform (Spectral)
- Approximate Entropy Test
- ...



Combinazione sorgenti casualità

Siano r_1, r_2, \dots, r_k bit/numeri casuali
provenienti da diverse ed indipendenti
sorgenti

$$b = r_1 \oplus r_2 \oplus \dots \oplus r_k$$

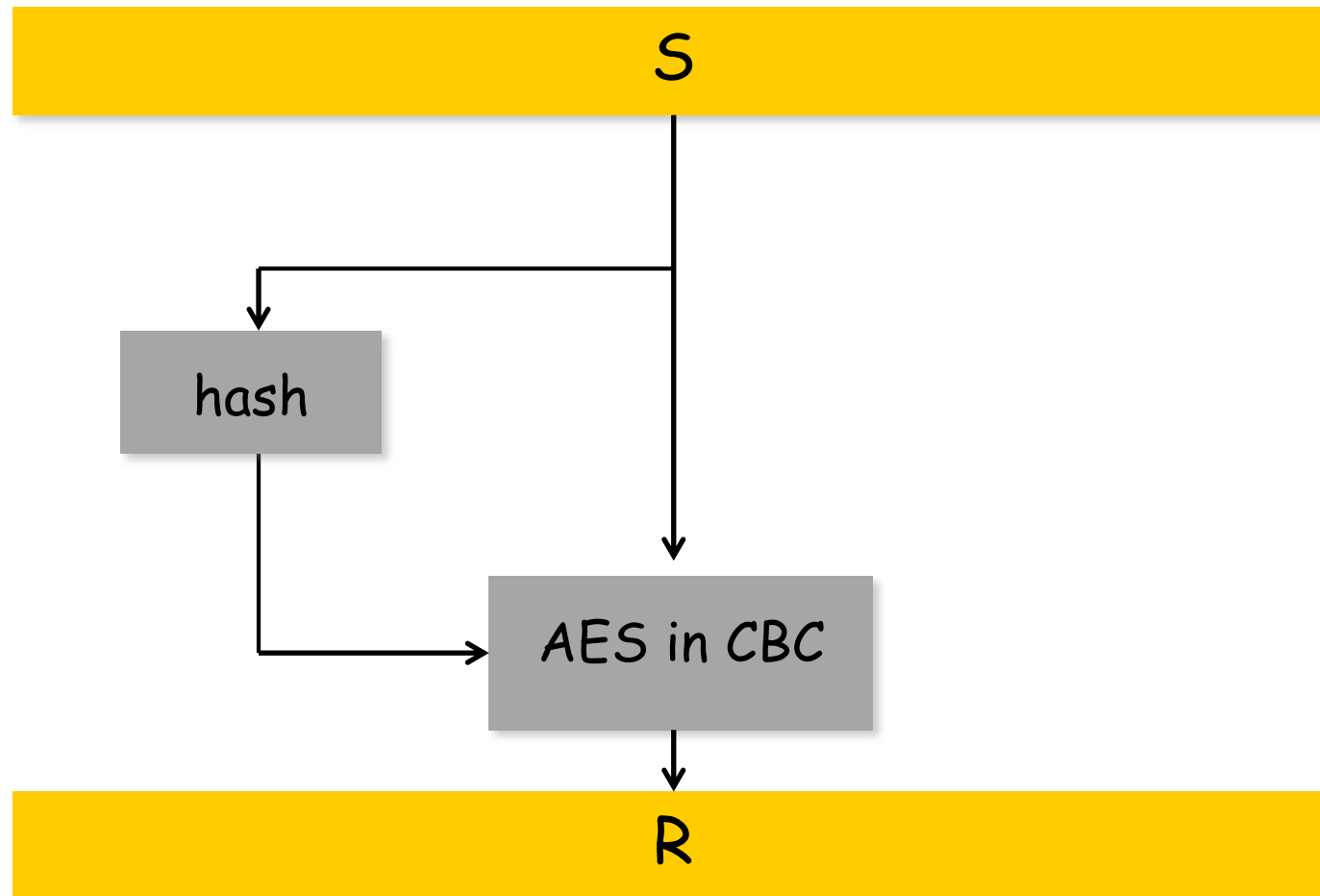
Se anche uno solo tra r_1, r_2, \dots, r_k è
veramente casuale, allora lo è anche b

Mischiare più sorgenti

- In genere si ottengono pochi bit casuali
- Usare più sorgenti, non correlate tra loro
- Mischiare i bit ottenuti tramite una opportuna funzione
 - Evita che ci siano bit facili da indovinare
 - Funzione complessa e non lineare di tutti i bit input

Nessuna funzione deterministica aumenta la casualità !

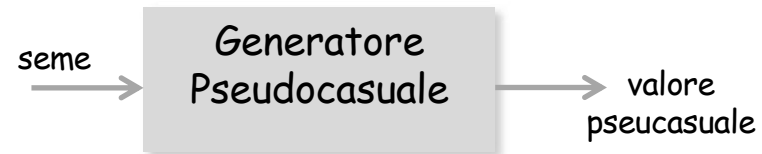
Mischiamiento: un esempio



Indice

- Generazione casuale
- Generazione pseudocasuale

Generazione pseudocasuale



- X_0 valore iniziale o seme
- Generazione deterministica della sequenza
$$X_{i+1} = f(i, X_0 X_1 \dots X_i) \quad i=0,1,2,\dots$$

Generazione pseudocasuale

Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.

John Von Neumann, 1951



Generatore a congruenze lineari

[Lehmer, 1949]

E' la tecnica più diffusa per la generazione pseudocasuale

$$X_{i+1} = (a \bullet X_i + b) \bmod m$$

Generatore a congruenze lineari

Esempio,

$$X_{i+1} = (263 \cdot X_i + 71) \bmod 100$$

$$x_0 = 79$$

$$x_1 = (263 \cdot 79 + 71) \bmod 100 = 20848 \bmod 100 = 48$$

$$x_2 = (263 \cdot 48 + 71) \bmod 100 = 12695 \bmod 100 = 95$$

$$x_3 = (263 \cdot 95 + 71) \bmod 100 = 25056 \bmod 100 = 56$$

$$x_4 = (263 \cdot 56 + 71) \bmod 100 = 14799 \bmod 100 = 99$$

...

Sequenza generata: 79, 48, 95, 56, 99, 8, 75, 96, 68,
36, 39, 28, 35, 76, 59, 88, 15, 16, 79, 48, 95, ...

Generatore a congruenze lineari

Borland C/C++ $X_{i+1} = (22695477 \cdot X_i) \bmod 2^{32}$ bits 30..16 in *rand()*

Newlib $X_{i+1} = (6364136223846793005 \cdot X_i + 1) \bmod 2^{32}$ bits 63..32

Microsoft Visual/Quick C/C++ $X_{i+1} = (214013 \cdot X_i + 2531011) \bmod 2^{32}$ bits 30..16

Microsoft Visual Basic $X_{i+1} = (1140671485 \cdot X_i + 12820163) \bmod 2^{24}$

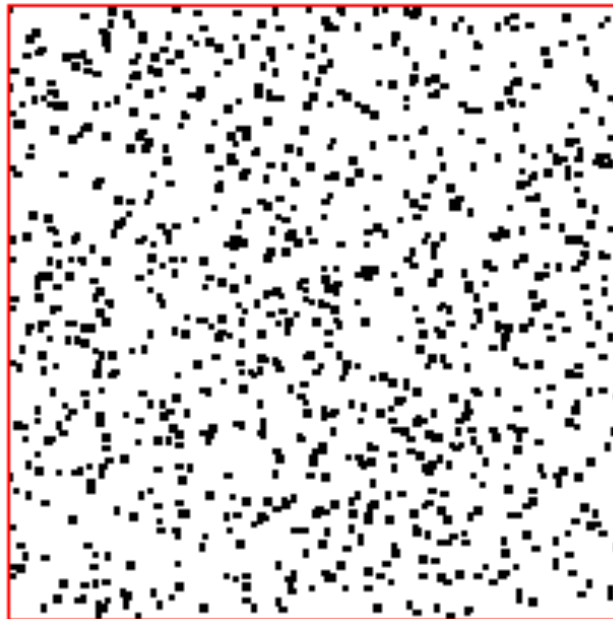
ANSI C $X_{i+1} = (1103515245 \cdot X_i + 12345) \bmod 2^{31}$ bits 30..16

Java $X_{i+1} = (25214903917 \cdot X_i + 11) \bmod 2^{48}$ bits 47..16

...

$2^{31}-1$ è numero primo, conveniente per aritmetica a 32 bit

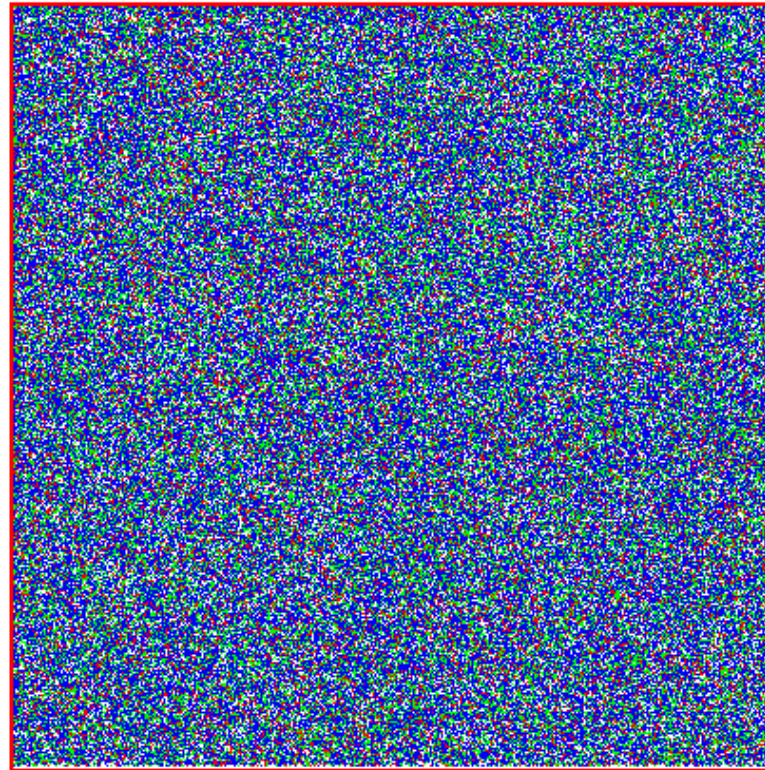
Plot (x_i, x_{i+1})



$$X_{i+1} = (16807 \cdot X_i) \bmod 2^{31}-1 = 2147483647$$

seme=1, 1.000 punti

(x_i, x_{i+1}) (x_i, x_{i+2}) (x_i, x_{i+3})



$$X_{i+1} = (16807 \cdot X_i) \bmod 2^{31}-1 = 2147483647$$

seme=1, 100.000 punti

<http://www.math.utah.edu/~alfeld/Random/Random.html>

Mersenne Twister (MT)

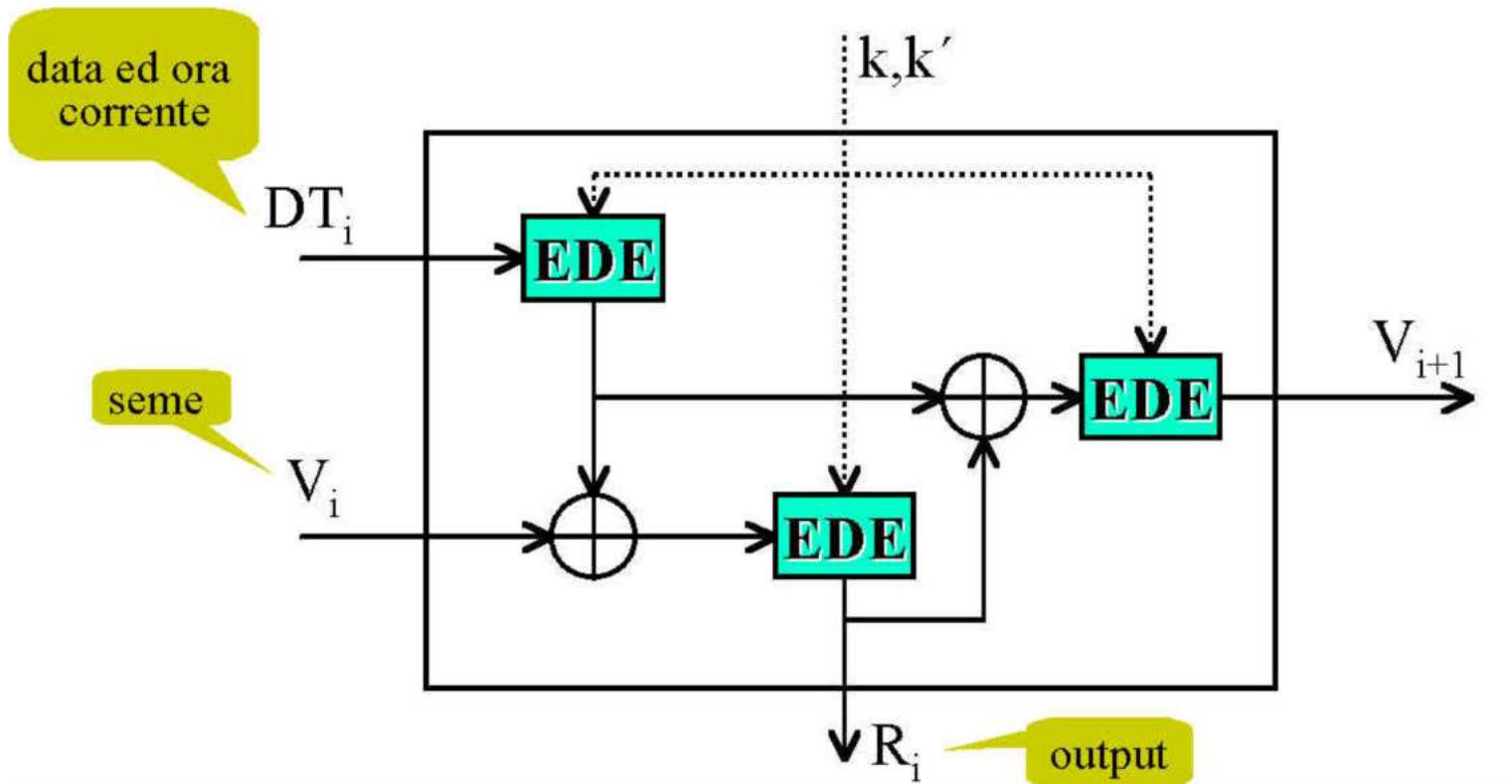
- Sviluppato da Makoto Matsumoto e Takuji Nishimura nel 1996/1997
- Lunghezza ciclo = $2^{19937}-1$
 - 24-esimo numero primo di Mersenne, $M_{19937} = 2^{19937}-1$
 - Il 48-esimo $2^{57.885.161}-1$ scoperto il 25 gen 2013 da GIMPS
- Uso efficiente memoria
 - codice C usa solo 624 word per area lavoro
- <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html>



Great Internet Mersenne
Prime Search

ANSI X9.17

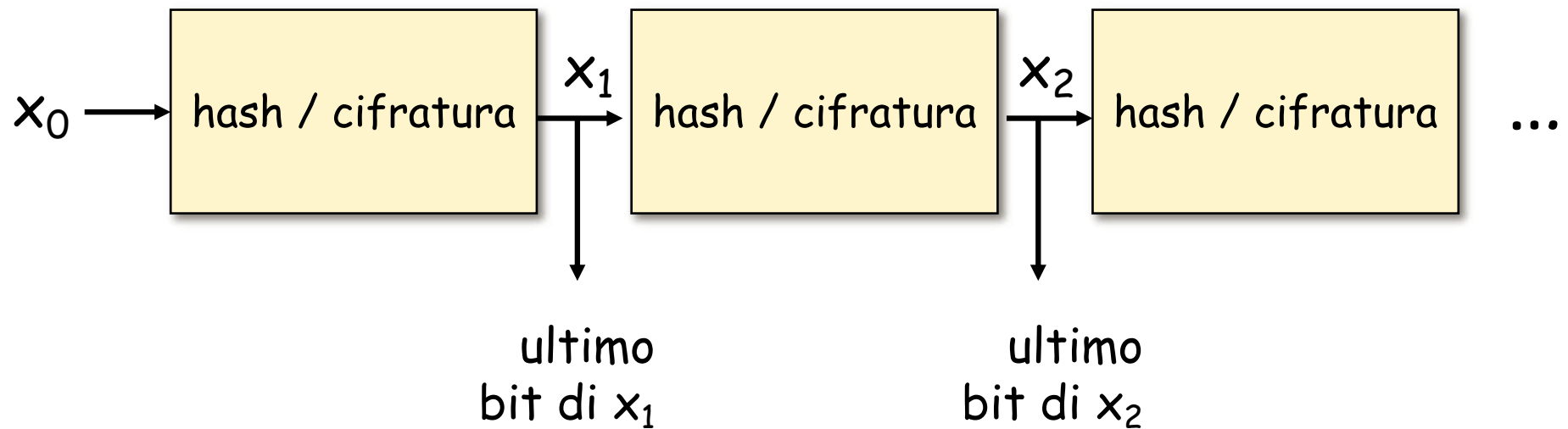
Standard specifica key management per banche, 1992



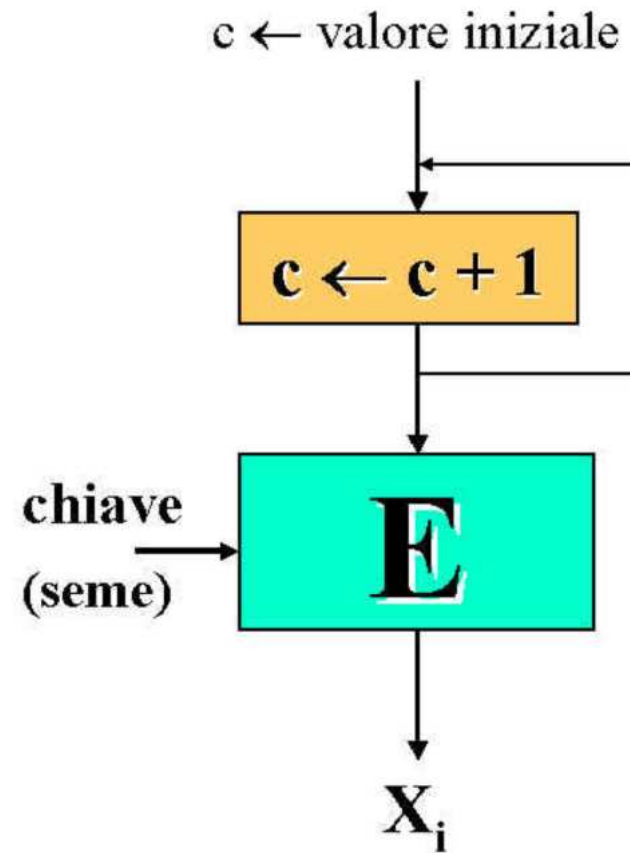
Altri generatori pseudocasuali

- Capstone/Fortezza
- DSA (Digital Signature Specification)
- Yarrow-160
- Fortuna

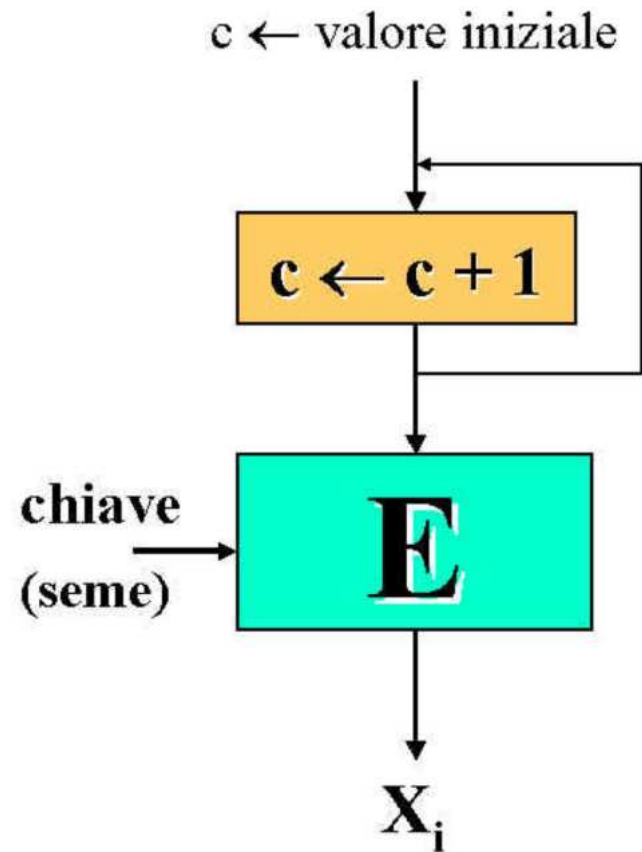
Catena di hash / cifratura



Cifratura di un contatore

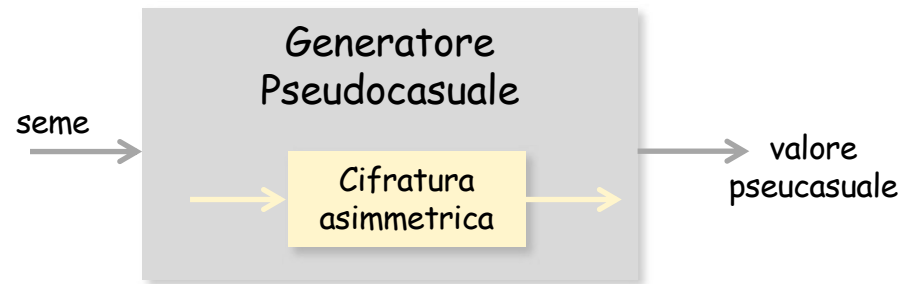


Cifratura di un contatore



Si può cifrare l'output di un generatore (anche a congruenze lineari), anzichè il contatore $c, c+1, c+2, \dots$,

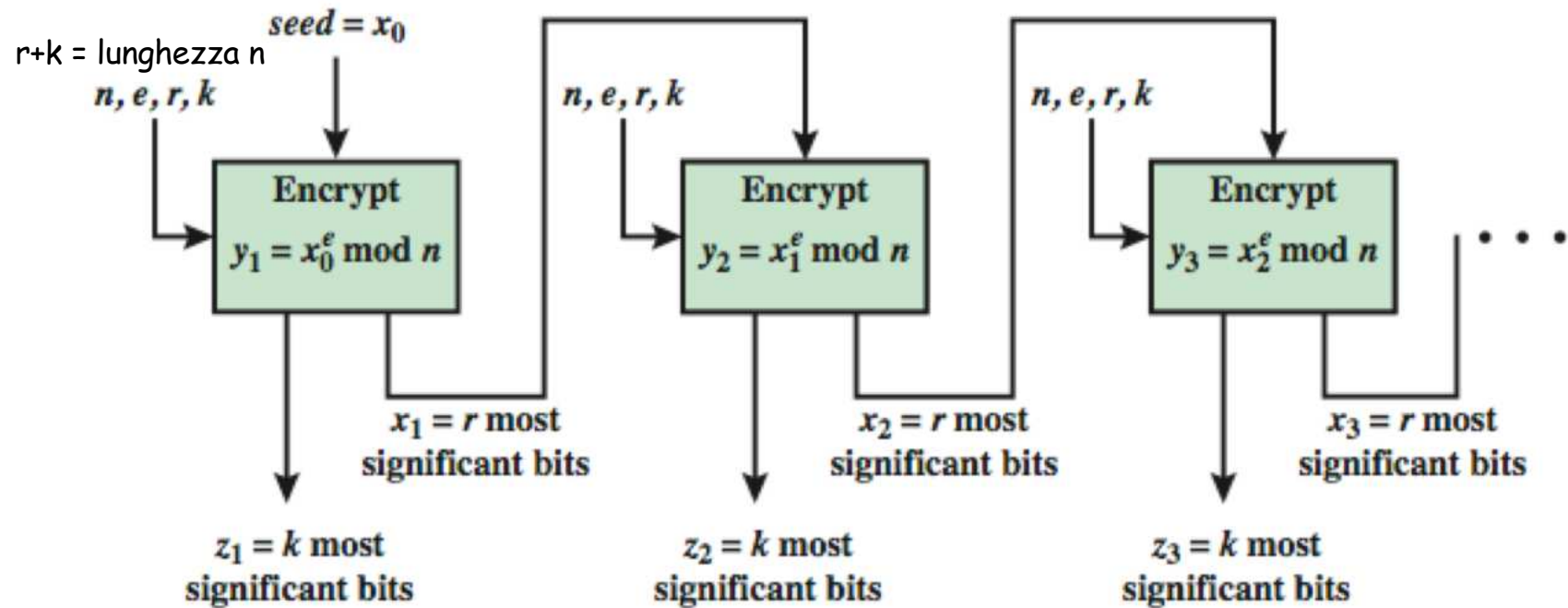
Generatori pseudocasuali basati su cifrari asimmetrici



Cifrari asimmetrici più lenti dei simmetrici

Micali-Schnorr PRNG

ANSI X9.82, ISO 18031



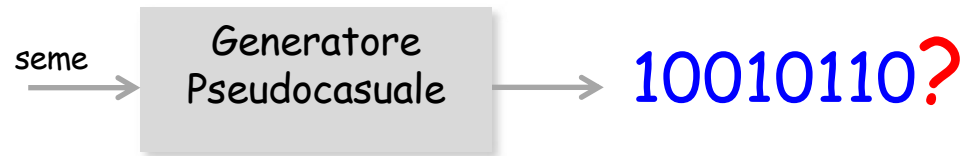
Generatore crittograficamente forte

Deve passare i seguenti test:

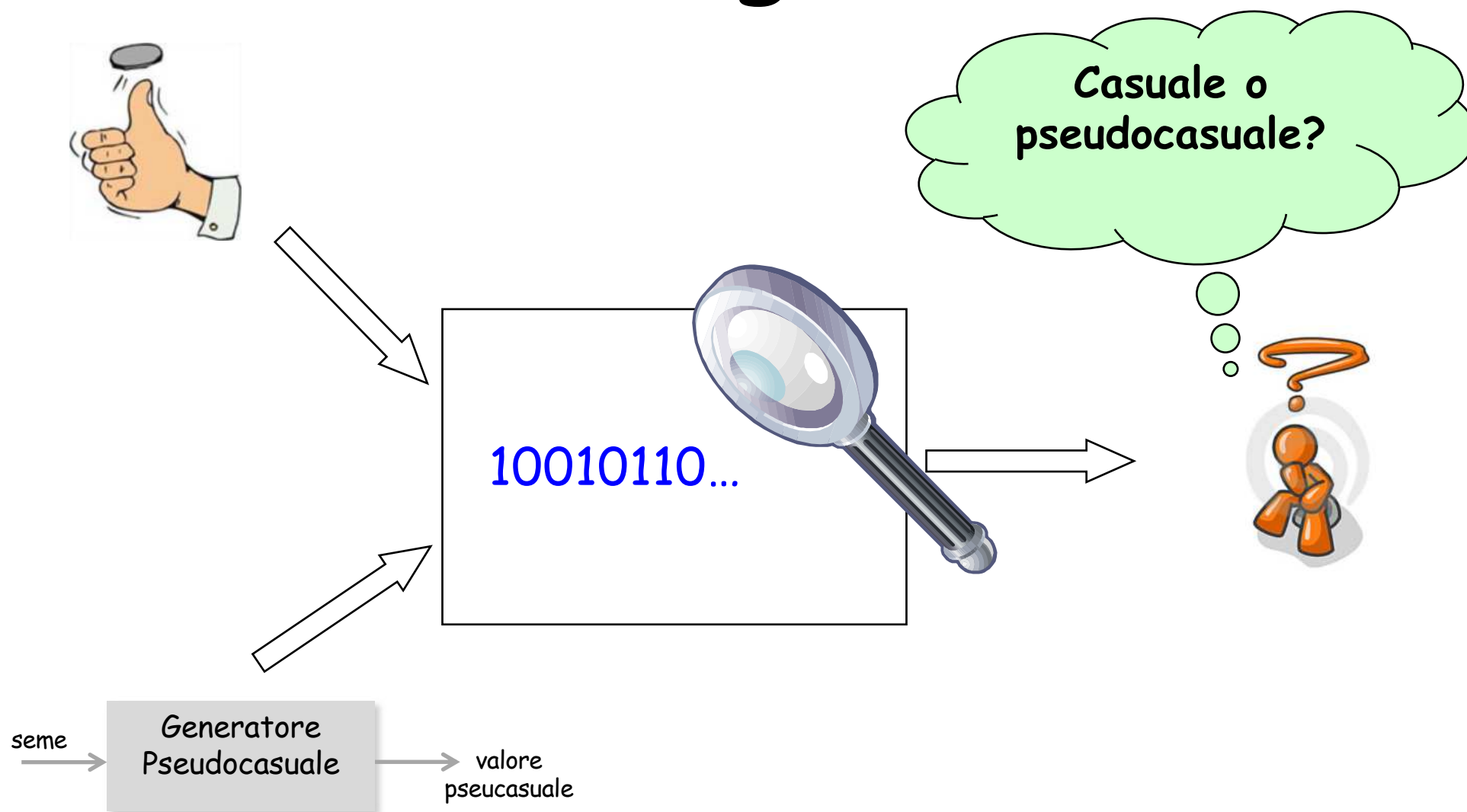
- Test del prossimo bit
 - Test statistico (indistinguibilità)
- con probabilità significativamente migliore di $1/2$

I due test sono equivalenti

Test del prossimo bit

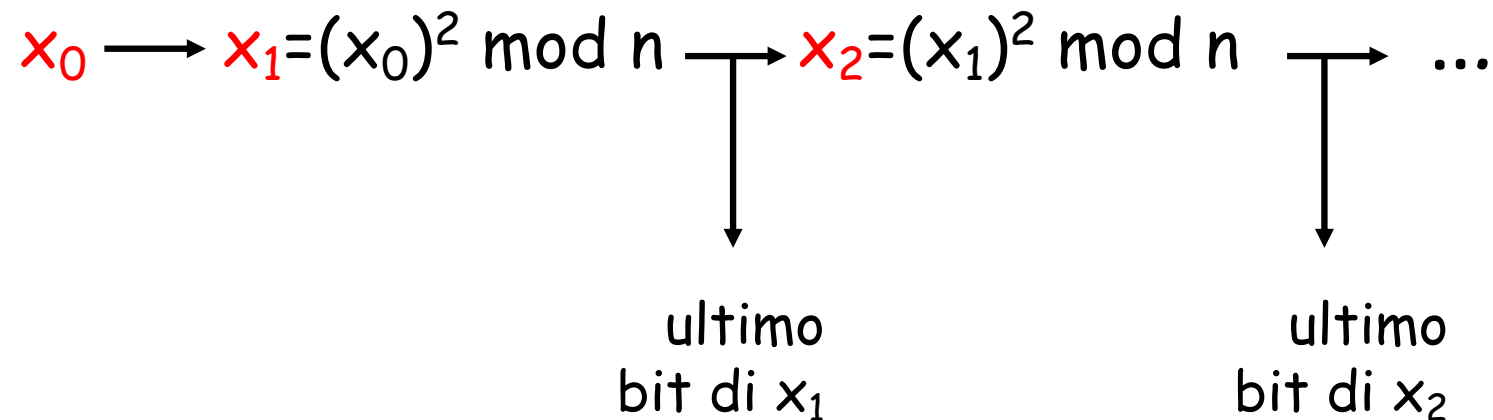


Indistinguibilità



Blum, Blum, Shub

[1986]


$$n = pq$$

p e q numeri primi congrui a 3 mod 4

 $x_0 \text{ in } Z_n^*$

Indicazioni del NIST

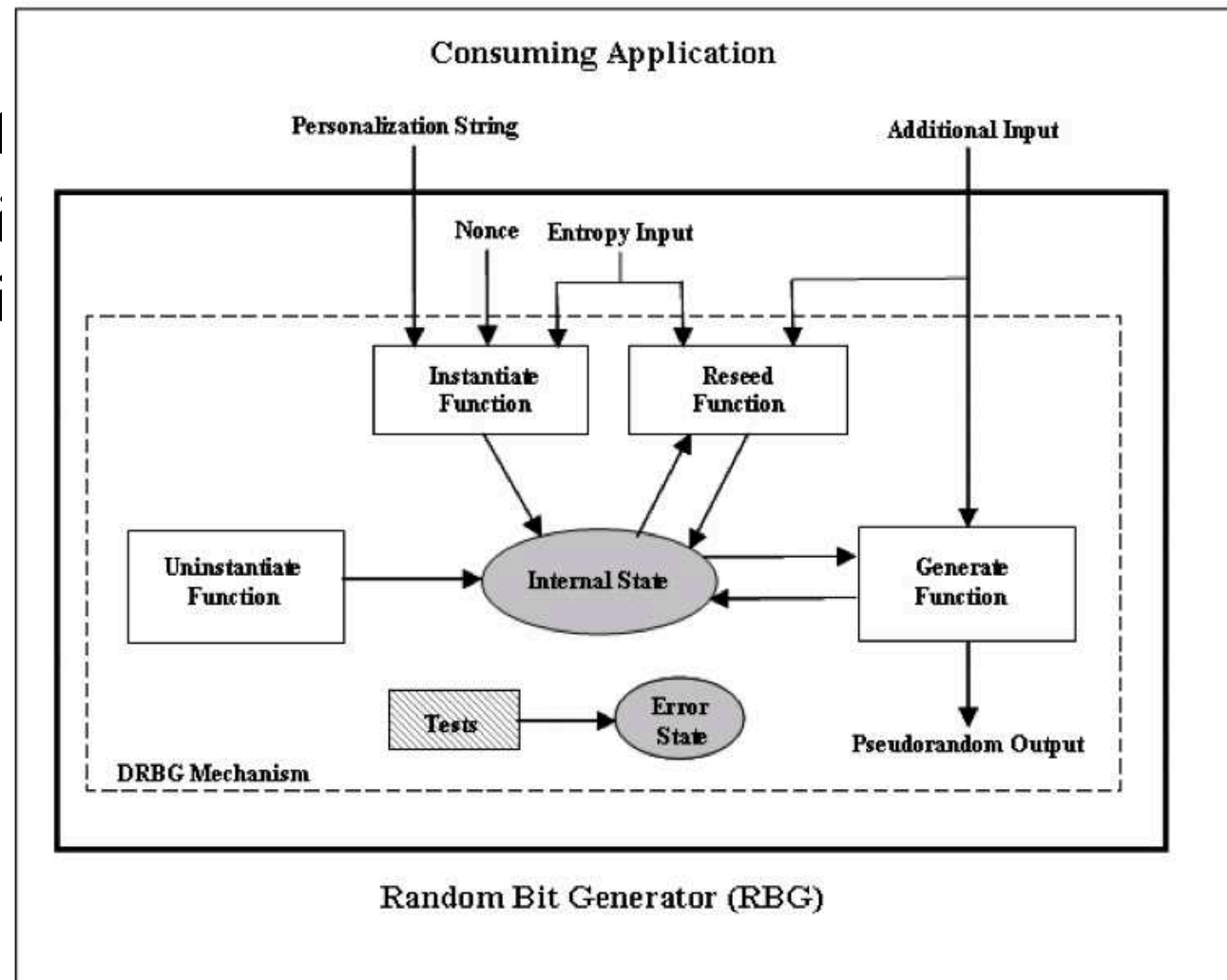
NIST Special Publication 800-90,
Recommendation for Random Number Generation
Using Deterministic Random Bit Generators
June 2006

Deterministic Random Bit Generator, 4 proposte:

- Basati su funzioni hash: [Hash_DRBG](#), [HMAC_DRBG](#)
- Basati su cifrari a blocchi: [CTR_DRBG](#)
- Basati su curve ellittiche: [Dual_EC_DRBG](#)

Modello funzionale

NIST Special Publication 800-90
Recommendation for Random Bit
Generation Using Deterministic
Random Bit Generators
June 2006



The New York Times September 5, 2013

Secret Documents Reveal N.S.A. Campaign Against Encryption

Documents show that the N.S.A. has been waging a war against encryption using a battery of methods that include working with industry to weaken encryption standards, making design changes to cryptographic software, and pushing international encryption standards it knows it can break.

http://www.nytimes.com/interactive/2013/09/05/us/documents-reveal-nsa-campaign-against-encryption.html?_r=0

The New York Times

Government Announces Steps to Restore Confidence on Encryption Standards

By NICOLE PERLROTH SEPTEMBER 10, 2013, 7:02 PM 9 Comments



Patrick Semansky/Associated Press

As part of its efforts to foil Web encryption, the National Security Agency inserted a backdoor into a 2006 security standard adopted by the National Institute of Standards and Technology, the federal agency charged with recommending cybersecurity standards.

But internal memos leaked by a former N.S.A. contractor, Edward Snowden, suggest that the N.S.A. generated one of the random number generators used in a 2006 N.I.S.T. standard — [called the Dual EC DRBG standard](#) — which contains a back door for the N.S.A. In publishing the standard, N.I.S.T. acknowledged “contributions” from N.S.A., but not primary authorship.

Internal N.S.A. memos describe how the agency subsequently worked behind the scenes to push the same standard on the International Organization for Standardization. “The road to developing this standard was smooth once the journey began,” one memo noted. “However, beginning the journey was a challenge in finesse.”

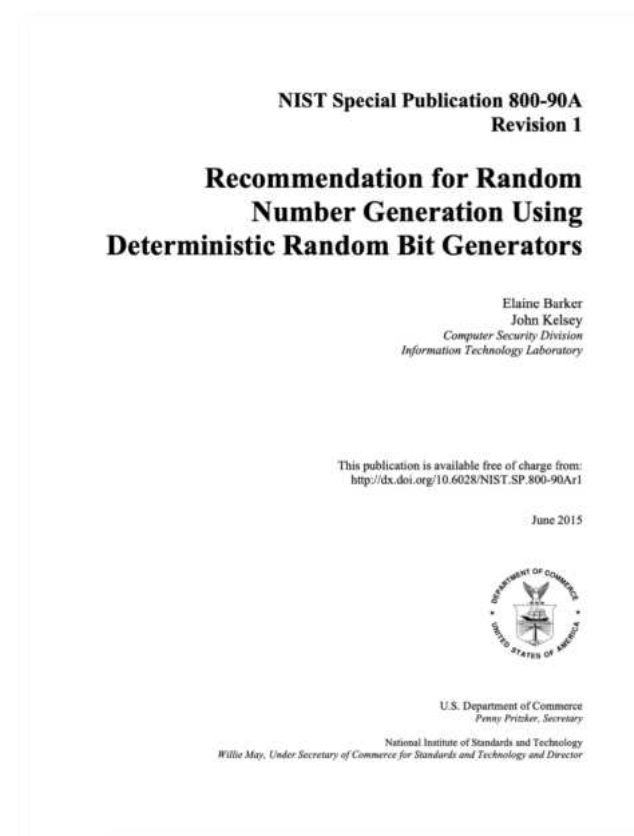
<http://bits.blogs.nytimes.com/2013/09/10/government-announces-steps-to-restore-confidence-on-encryption-standards/>

Indicazioni del NIST

NIST Special Publication 800-90A,
Recommendation for Random Number Generation Using
Deterministic Random Bit Generators
January 2012

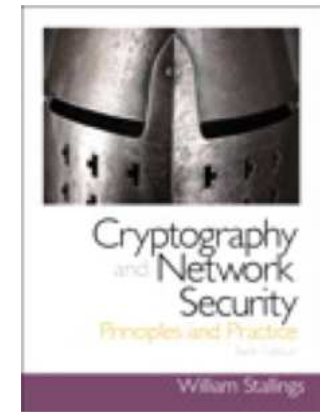
NIST Special Publication 800-90 A Rev. 1
Recommendation for Random Number
Generation Using Deterministic Random
Bit Generators

Eliminazione del Dual Elliptic Curve
Deterministic Random Bit Generator
(Dual_EC_DRBG).



Bibliografia

- **Cryptography and Network Security**
by W. Stallings, 6/e, 2014
 - cap. 7



- **Tesina su Randomness**
 - <http://www.di.unisa.it/~ads/corso-security/www/CORSO-9900/>
 - Sicurezza su reti, a.a. 1990-2000

Domande?

