Medindo a qualidade de geradores de números aleatórios

Daniel Chicayban Bastos
Luis Antonio Brasil Kowada

Universidade Federal Fluminense
Instituto de Computação
Av. Gal. Milton Tavares de Souza
São Domingos, Niterói, RJ, 24210-310

10 de dezembro de 2018

Autores em ordem alfabética.

Introdução

Um exemplo de gerador pseudo-aleatório:

```
static unsigned int y = 2463534242U;
unsigned int xorshift(void) {
  y = y ^ (y << 13);
  y = y ^ (y >> 17);
  y = y ^ (y << 5);
  return y;
}
```

Acrônimos:
PRNG := pseudo-random number generator
CSPRNG: := cryptographically secure PRNG

Introdução

- 1996: Netscape usa o PID do navegador como parte da semente de um PRNG para gerar chaves privadas. (Goldberg, Wagner, 1996.)
- 2012: análise de chaves TLS e SSH: 64 mil chaves privadas de servidores TLS e 108 mil chaves de servidores SSH foram obtidas, todas devido a má geração de números aleatórios. (Heninger et al., 2012.)
- 2013: Taiwan inicia (em 2003) projeto de cartões inteligentes. Chaves RSA de 1024 bits eram geradas pelos cartões usando um PRNG. Em 2013, 184 chaves foram fatoradas. (Bernstein et al., 2013.)

O estado da arte em testes estatísticos

- 1969, Knuth, TAOCP v2, “statistical tests”
- 1996, Marsaglia, DIEHARD
- 2001, Basham et al., NIST SP 800-22
- 2004, Brown, DieHarder
- 2007, l’Ecuyer e Simard, TestU01

Comparando e classificando

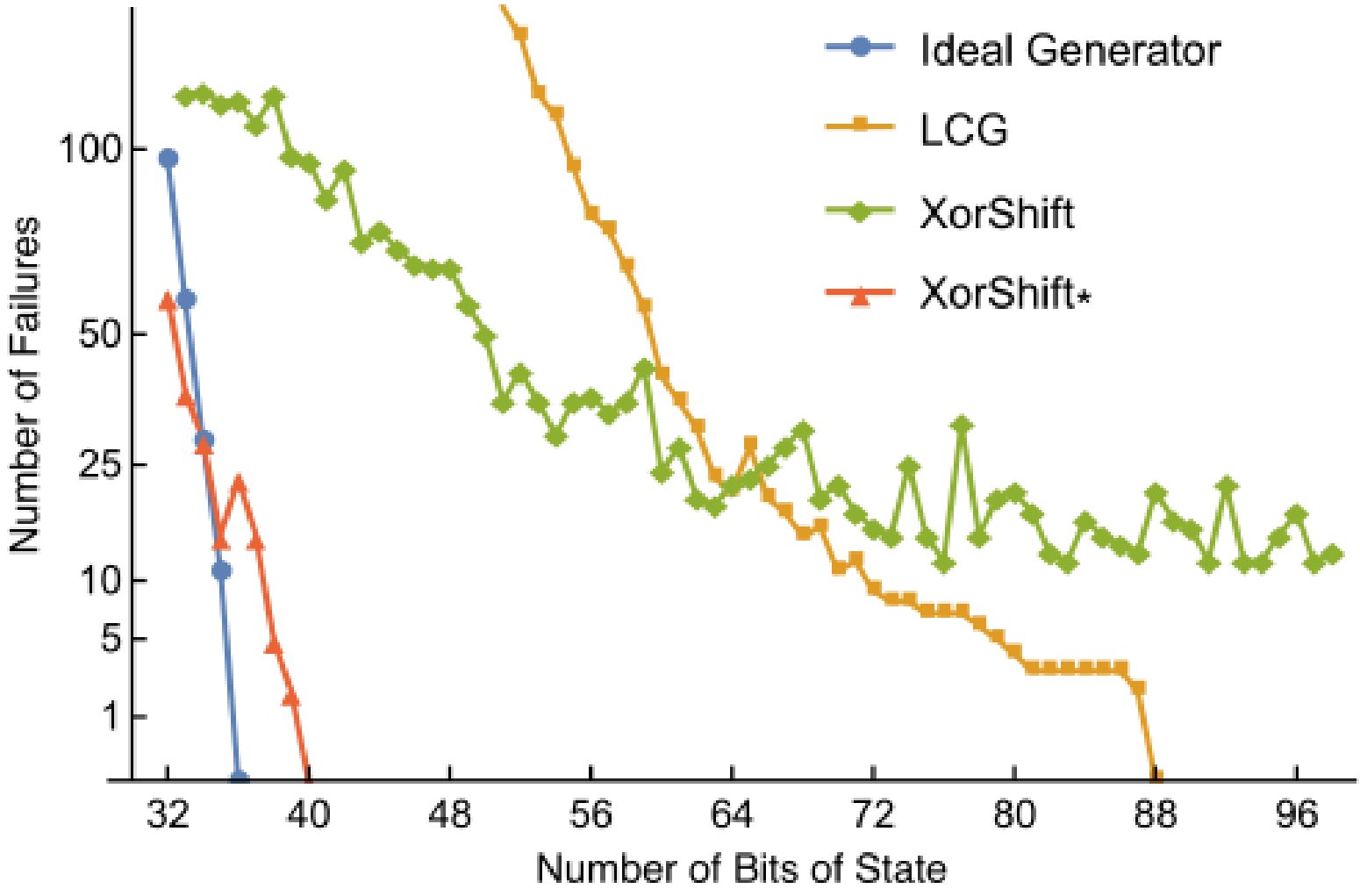
Dois geradores passam pela bateria `SmallCrush`, TestU01. Qual dos dois é melhor? Como podemos compará-los? Seja b o tamanho (em bits) do estado interno de um gerador G_i ideal.

- G_i fracassará na `SmallCrush` se $b < 32$
- G_i fracassará na `Crush` se $b < 35$
- G_i fracassará na `BigCrush` se $b < 36$

Então G_i com $b = 32$ não tem como passar pela `BigCrush`, mas tem seu mérito se puder passar pela `SmallCrush` com um mínimo de estado. Podemos comparar geradores por essa métrica do mínimo número de bits para obter sucesso numa bateria.

Fonte: O'Neill, M. E. (2014). PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation. Technical Report HMC-CS-2014-0905, Harvey Mudd College, Claremont, CA.

Comparando e classificando



Problemas em aberto

Suponha que:

- G com 100 bits de estado passe numa bateria T
- G com 99 bits de estado não passe em T

Isso significa que 100 bits é o mínimo para G obter sucesso em T. Aumente o estado interno de G para 110 bits. Isso dá a G uma folga de 10 bits relativo a T. Essa folga significa a mesma coisa para diferentes geradores? A inclinação da reta significa alguma coisa a respeito da qualidade do gerador?

Geradores criptograficamente seguros

Definição 1. Um PRNG é chamado de **computacionalmente seguro** se o melhor algoritmo para violar sua segurança requer pelo menos N operações, sendo N algum natural específico grande.

"*We might define a cryptosystem to be computationally secure if the best algorithm for breaking it requires at least N operations, where N is some specified, very large number.*"

Fonte: Stinson, D. R. (2006). Cryptography, Theory and Practice. Chapman and Hall, CRC, 3rd edition. Capítulo 2, seção 2.1, página 45.

Geradores criptograficamente seguros

Definição 2. Um PRNG é chamado de **comprovadamente seguro** se violar sua segurança requer solucionar um problema considerado difícil em tempo polinomial.

"*If a cryptosystem can be 'broken' in some specific way, then it would be possible to efficiently solve some well-studied problem that is thought to be difficult. For example, it may be possible to prove a statement of the type "a given cryptosystem is secure if a given integer n cannot be factored." [...] [B]ut it must be understood that this approach only provides a proof of security relative to some other problem, not an absolute proof of security. This is a similar situation to proving that a problem is NP-complete [...].*"

Fonte: Stinson, D. R. (2006). Cryptography, Theory and Practice. Chapman and Hall, CRC, 3rd edition. Capítulo 2, seção 2.1, página 45.

Geradores criptograficamente seguros

Definição 3. Um PRNG possui **segredo perfeito** se não se pode obter qualquer informação sobre seu estado interno a partir de seu *output*.

"A cryptosystem has perfect secrecy if the *a posteriori* probability that the plaintext is x, given that the cyphertext y is observed, is identical to the *a priori* probability that the plaintext is x."

Fonte: Stinson, D. R. (2006). Cryptography, Theory and Practice. Chapman and Hall, CRC, 3rd edition. Capítulo 3, seção 2.3, definição 2.3, página 50.

Ferramentas

- TestU01 é uma biblioteca C.
- Como testar um gerador escrito em linguagem *L*?

```
$./testu01 --help
Usage:  testu01 -b small|medium|big -n generator-name

$./java-prgn | ./testu01 -b small -n java
...
$
```

- TestU01 trabalha com 32 bits.
- Como testar um gerador de 64 bits? De 31 bits?
- TestU01 tem 3 baterias.
- Como testar um gerador em testes avulsos?
- Como repetir automaticamente baterias com amostras maiores?

Um pequeno experimento

Geradores de linguagens populares sob escrutínio de `SmallCrush`.

linguagem	gerador	período	estado	fracassos
Java	`drand48`	2^{31}	48	5
Python, PHP	`mt19937`	$2^{19937} - 1$	20032	0
C++	`minstd`	$2^{31} - 2$	32	9
Racket	`mrg32k3a`	$2^{191} - 209$	256	0

- `mt19937` fracassa no *linear complexity test*, que não faz parte de `SmallCrush`, embora seja um teste rápido de ser feito.
- `mt19937` tem seu estado interno completamente determinado com 624 números.