Pseudo-random Number Generation

Qiuliang Tang

Random Numbers in Cryptography

- The keystream in the one-time pad
- The secret key in the DES encryption
- The prime numbers p, q in the RSA encryption
- The private key in DSA
- The initialization vectors (IVs) used in ciphers

Pseudo-random Number Generator

- Pseudo-random number generator:
  - A polynomial-time computable function f (x) that expands a short random string x into a long string f (x) that appears random
- Not truly random in that:
  - Deterministic algorithm
  - Dependent on initial values
- Objectives
  - Fast
  - Secure

Pseudo-random Number Generator

- Classical PRNGs
  - Linear Congruential Generator

- Cryptographically Secure PRNGs
  - RSA Generator
  - Blum-Micali Generator
  - Blum-Blum-Shub Generator

- Standardized PRNGs
  - ANSI X9.17 Generator
  - FIPS 186 Generator

Linear Congruential Generator - Algorithm

- Based on the linear recurrence:
  $x_i = a x_{i-1} + b \mod m \qquad i \geqslant 1$

  Where
  - x_0 is the seed or start value
  - a is the multiplier
  - b is the increment
  - m is the modulus

  Output
  - $(x_1, x_2, \ldots, x_k)$
  - $y_i = x_i \mod 2$
  - $Y = (y_1 y_2 \ldots y_k)$ ← pseudo-random sequence of K bits

Linear Congruential Generator - Example

- Let $x_n = 3 x_{n-1} + 5 \bmod 31 \quad n \geqslant 1$, and $x_0 = 2$
  - 3 and 31 are relatively prime, one-to-one (affine cipher)
  - 31 is prime, order is 30

- Then we have the 30 residues in a cycle:
  - 2, 11, 7, 26, 21, 6, 23, 12, 10, 4, 17, 25, 18, 28, 27, 24, 15, 19, 0, 5, 20, 3, 14, 16, 22, 9, 1, 8, 29, 30

- Pseudo-random sequences of 10 bits
  - when $x_0 = 2$
    1101010001
  - When $x_0 = 3$
    0001101001

Linear Congruential Generator - Security

- Fast, but insecure
  - Sensitive to the choice of parameters a, b, and m
  - Serial correlation between successive values
  - Short period, often $m=2^{32}$ or $m=2^{64}$

Linear Congruential Generator - Application

- Used commonly in compilers
  - Rand()
- Not suitable for high-quality randomness applications
  - Issues with the RANDU random number algorithm
  - Use Mersenne Twister algorithm in Monte Carlo simulations
  - Longer period $2^{19937}-1$
- Not suitable for cryptographic applications
  - Use cryptographically secure pseudo-random number generators

Cryptographically Secure

- Passing all polynomial-time statistical tests
  - There is no polynomial-time algorithm that can correctly distinguish a string of k bits generated by a pseudo-random bit generator (PRBG) from a string of k truly random bits with probability significantly greater than ½
  - Probability distributions indistinguishable
- Passing the next-bit test
  - Given the first k bits of a string generated by PRBG, there is no polynomial-time algorithm that can correctly predict the next $(k+1)^{th}$ bit with probability significantly greater than ½
  - Next-bit unpredictable
- The two notions are equivalent
  - Proved by Yao

Cryptographically Secure PRNGs

- A PRNG from any one-way function
  - A function f is one-way if it is easy to compute $y = f(x)$ but hard to compute $x = f^{-1}(y)$
  - There is a PRNG if and only if there is a one-way function
- One-way functions
  - The RSA function
  - The discrete logarithm function
  - The squaring function
- Cryptographically secure PRNGs
  - RSA Generator
  - Blum-Micali Generator
  - Blum-Blum-Shub Generator

RSA Generator - Algorithm

- Based on the RSA one-way function:
  - $x_i = x_{i-1}^b \bmod n \qquad\qquad i \geqslant 1$

Where

- x_0 is the seed, an element of Z_n^*
- $n = p*q$, p and q are large primes
- $\gcd(b, \Phi(n)) = 1$ where $\Phi(n) = (p-1)(q-1)$
- n and b are public, p and q are secret

Output

$(x_1, x_2, \ldots, x_k)$

$y_i = x_i \bmod 2$

$Y = (y_1 y_2 \ldots y_k)$ ← pseudo-random sequence of K bits

RSA Generator - Security

- RSA Generator is provably secure
  - It is difficult to predict the next number in the sequence given the previous numbers, assuming that it is difficult to invert the RSA function (Shamir)

RSA Generator - Efficiency

- RSA Generator is relatively slow
  - Each pseudo-random bit y_i requires a modular exponentiation operation
  - Can be improved by extracting j least significant bits of x_i instead of 1 least significant bit, where j=cloglogn and c is a constant
  - Micali-Schnorr Generator improves the efficiency

Blum-Micali Generator - Concept

- Discrete logarithm
  - Let p be an odd prime, then $(Z_p^*, \cdot)$ is a cyclic group with order p-1
  - Let g be a generator of the group, then $|\langle g \rangle| = p-1$, and for any element a in the group , we have $g^k = a \bmod p$ for some integer k
  - If we know k, it is easy to compute a
  - However, the inverse is hard to compute, that is, if we know a, it is hard to compute $k = \log_g a$

- Example
  - $(Z_{17}^*, \cdot)$ is a cyclic group with order 16, 3 is the generator of the group and $3^{16} = 1 \bmod 17$
  - Let k=4, $3^4 = 13 \bmod 17$, which is easy to compute
  - The inverse: $3^k = 13 \bmod 17$, what is k? what about large p?

Blum-Micali Generator - Algorithm

- Based on the discrete logarithm one-way function:
  - Let p be an odd prime, then $(Z_p^*, \cdot)$ is a cyclic group
  - Let g be a generator of the group, then for any element a, we have $g^k = a \bmod p$ for some k
  - Let x_0 be a seed

$$x_i = g^{x_{i-1}} \bmod p \qquad i \geqslant 1$$

Output

$(x_1, x_2, \ldots, x_k)$

$y_i = 1$ if $x_i \geqslant (p-1)/2$

$y_i = 0$ otherwise

$Y = (y_1 y_2 \ldots y_k)$ ← pseudo-random sequence of K bits

Blum-Micali Generator - Security

- Blum-Micali Generator is provably secure
  - It is difficult to predict the next bit in the sequence given the previous bits, assuming it is difficult to invert the discrete logarithm function (by reduction)

Blum-Blum-Shub Generator - Concept

- Quadratic residues
  - Let p be an odd prime and a be an integer
  - a is a quadratic residue modulo p if a is not congruent to 0 mod p and there exists an integer x such that $a \equiv x^2 \bmod p$
  - a is a quadratic non-residue modulo p if a is not congruent to 0 mod p and a is not a quadratic residue modulo p
- Example
  - Let p=5, then $1^2 = 1$, $2^2 = 4$, $3^2 = 4$, $4^2 = 1$
  - 1 and 4 are quadratic residues modulo 5
  - 2 and 3 are quadratic non-residues modulo 5

Blum-Blum-Shub Generator - Algorithm

- Based on the squaring one-way function
  - Let p, q be two odd primes and $p \equiv q \equiv 3 \bmod 4$
  - Let $n = p*q$
  - Let x_0 be a seed which is a quadratic residue modulo n

$x_i = x_{i-1}^2 \bmod n \qquad i \geqslant 1$

Output

$(x_1, x_2, \ldots, x_k)$

$y_i = x_i \bmod 2$

$Y = (y_1 y_2 \ldots y_k)$ ← pseudo-random sequence of K bits

Blum-Blum-Shub Generator - Security

- Blum-Blum-Shub Generator is provably secure
  - Euler's criterion
  - Legendre symbol
  - Jacobi symbol
  - Composite quadratic residues

Blum-Blum-Shub Generator - Security

- Euler's criterion
  - Let p be an odd prime. Then a is a quadratic residue modulo p if and only if $a^{(p-1)/2} \equiv 1 \bmod p$

- Proof:
  - Suppose $a \equiv x^2 \bmod p$, then
    $a^{(p-1)/2} \equiv x^{2*(p-1)/2} \equiv x^{p-1} \equiv 1 \bmod p$ (By Fermat's little theorem)

  - Suppose $a^{(p-1)/2} \equiv 1 \bmod p$.
    Let g be a generator of the group $(Z_p^*, \cdot)$, then $a \equiv g^k \bmod p$ for some integer k
    We have $a^{(p-1)/2} \equiv g^{k*(p-1)/2} \equiv g^{k/2} \equiv 1 \bmod p$
    Then k must be even
    Let k=2m, then $a \equiv (g^m)^2 \bmod p$
    which means that a is a quadratic residue modulo p

Blum-Blum-Shub Generator - Security

- Legendre symbol
  - Let p be an odd prime and a be an integer

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{if } a \equiv 0 \pmod{p} \\ 1 & \text{if } a \text{ is a quadratic residue modulo } p \\ -1 & \text{if } a \text{ is a quadratic non-residue modulo } p \end{cases}$$

  - If a is a multiple of p, then $a^{(p-1)/2} \equiv 0 \bmod p$
  - If a is a quadratic residue modulo p, then $a^{(p-1)/2} \equiv 1 \bmod p$
  - If a is a quadratic non-residue modulo p, then $a^{(p-1)/2} \equiv -1 \bmod p$ since $(a^{(p-1)/2})^2 \equiv a^{p-1} \equiv 1 \bmod p$

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod{p}$$

  - Example: Let p=5
    (0/5)=0; (1/5)=(4/5)=1; (2/5)=(3/5)=-1

Blum-Blum-Shub Generator - Security

- Jacobi symbol
  - Let n be an odd positive integer
  - p_i is the prime factor of n and e_i is the power of the prime factor
  - (a/p_i) is the Legendre symbol and (a/n) is the Jacobi symbol

$$n = \prod_{i=1}^{k} {p_i}^{e_i}$$

$$\left(\frac{a}{n}\right) = \prod_{i=1}^{k} \left(\frac{a}{p_i}\right)^{e_i}$$

  - Example: Let n=15=3*5
    (9/15)=(9/3)(9/5)=0
    (11/15)=(11/3)(11/5)=(2/3)(1/5)=(-1)(1)=-1
    (8/15)=(8/3)(8/5)=(2/3)(3/5)=(-1)(-1)=1
    (4/15)=(4/3)(4/5)=(1)(1)=1

Blum-Blum-Shub Generator - Security

- Composite quadratic residues
  - Let p, q be two odd primes and n = p*q
  - If (x/n) = (x/p)(x/q) = 1, then

    either (x/p) = (x/q) = 1   x is a quadratic residue modulo n

    or (x/p) = (x/q) = -1   x is a pseudo-square modulo n
  - It is difficult to determine if x is a quadratic residue modulo n as factoring n=p*q is difficult

$$\left(\frac{x}{n}\right) = \begin{cases} 0 & \text{if } \gcd(x, n) > 1 \\ 1 & \text{if } \left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = 1 \text{ or if } \left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = -1 \\ -1 & \text{if one of } \left(\frac{x}{p}\right) \text{ and } \left(\frac{x}{q}\right) = 1 \text{ and the other} = -1 \end{cases}$$

  - Example: Let n=15=3*5

    (8/15)=(8/3)(8/5)=(2/3)(3/5)=(-1)(-1)=1; 8 is a pseudo-square

    (4/15)=(4/3)(4/5)=(1)(1)=1; 4 is a quadratic residue

Blum-Blum-Shub Generator - Security

- Why $p \equiv q \equiv 3 \bmod 4$
  - Such that every quadratic residue x has a square root y which is itself a quadratic residue
  - Denote the square root of x to be y, that is, $x=y^2 \bmod n$
  - Let $p= 4m+3$, then $m=(p-3)/4$.
    - $y = x^{(p+1)/4} \bmod p$ is a principal square root of x modulo p
      $x^{(p-1)/2}=x^{(4m+3-1)/2}=x^{2m+1}=1 \bmod p \Rightarrow x^{2m+2}=x \bmod p$
      $\Rightarrow (x^{m+1})^2 = x \bmod p \Rightarrow y = x^{m+1} = x^{(p+1)/4}$
    - y is a quadratic residue
      $y^{(p-1)/2}= (x^{(p+1)/4})^{(p-1)/2}= (x^{(p-1)/2})^{(p+1)/4}=1^{(p+1)/4}=1 \bmod p$
  - Similar for q, $y = x^{(q+1)/4} \bmod q$
  - Since $n=p*q$ and x is a quadratic residue modulo n, then x has a unique square root modulo n (Chinese remainder theorem)

  - As a result, the mapping from x to $x^2 \bmod n$ is a bijection from the set of quadratic residues modulo n onto itself

Blum-Blum-Shub Generator - Application

- The basis for the Blum-Goldwasser probabilistic public-key encryption
  - To generate the keystream during encryption and decryption

Standardized PRNGs

- General characteristics
  - Not been proven to be cryptographically secure
  - Sufficient for most applications
  - Using one-way functions such as hash function SHA-1 or block cipher DES with secret key k

- Examples
  - ANSI X9.17 Generator
  - FIPS 186 Generator

ANSI X9.17 Generator

- Algorithm
  - Let s be a random secret 64-bit seed, E_k be the DES E-D-E two-key triple-encryption with key k, and m be an integer
  - $I = E_k(D)$, where D is a 64-bit representation of the date/time with finest available resolution
  - For i=1,...,m do

    $x_i = E_k(I \text{ XOR } s)$

    $s = E_k(x_i \text{ XOR } I)$
  - Return $(x_1, x_2, \ldots x_m)$ ← m pseudo-random 64-bit strings
- Used as an initialization vector or a key for DES

FIPS 186 Generator

- Used for DSA private keys
- Algorithm
  - Let q be a 160-bit prime number, and m be an integer
  - Let (b, G) = (160, DES) or (b, G) = (160..512, SHA-1)
  - Let s be a random secret seed with b bits
  - Let t be a 160-bit constant, t= 67452301 efcdab89 98badcfe 10325476 c3d2e1f0
  - For i=1…m do

    Either select a b-bit string y_i, or set y_i=0 (optional user input)

    $z_i = (s + y_i) \bmod 2^b$

    $a_i = G(t, z_i) \bmod q$

    $s = (1 + s + a_i) \bmod 2^b$
  - Return $(a_1, a_2, \ldots, a_m)$ ← m pseudo-random numbers in [0, q-1]

FIPS 186 Generator

- Used for DSA per message secret numbers
- Algorithm
  - Let q be a 160-bit prime number, and m be an integer
  - Let (b, G) = (160, DES) or (b, G) = (160..512, SHA-1)
  - Let s be a random secret seed with b bits
  - Let t be a 160-bit constant, t= efcdab89 98badcfe 10325476 c3d2e1f0 67452301
  - For i=1…m do

    $k_i = G(t, s) \bmod q$

    $s = (1 + s + k_i) \bmod 2^b$
  - Return $(k_1, k_2, \ldots, k_m)$ ← m pseudo-random numbers in [0, q-1]

References

1. D. Stinson. Cryptography, Theory and Practice. 3rd Ed. Chapman & Hall/CRC, 2006
2. A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. Handbook of applied cryptography. CRC Press, 1997
3. J. Hastad, R. Impagliazzo, L. Levin, and M. Luby. *A pseudorandom generator from any one-way function*. SIAM Journal on Computing, 28(4): 1364-1393, 1999
4. S. Goldwasser and M. Bellare. *Lecture Notes on Cryptography*. 2008. http://cseweb.ucsd.edu/~mihir/papers/gb.pdf
5. P. Junod. *Cryptographic Secure Pseudo-Random Bits Generation: The Blum-Blum-Shub Generator*. 1999. http://crypto.junod.info/bbs.pdf
6. M. J. Fischer. *Pseudorandom Sequence Generation*. Yale University. http://zoo.cs.yale.edu/classes/cs467/2006f/course/handouts/ho15.pdf
7. Federal Information Processing Standards Publication. *Digital Signature Standard (DSS).* 2000. http://csrc.nist.gov/publications/fips/archive/fips186-2/fips186-2.pdf

Quiz

1. Name one criterion when considering a pseudo-random number generator to be cryptographically secure
2. Name the one-way function that the Blum-Micali generator is based on
3. What are the four concepts that are used when considering the security of the Blum-Blum-Shub generator ?
4. Let p be an odd prime and $p \equiv 3 \bmod 4$. Let x be a quadratic residue modulo p. Let y be the principal square root of x. What is y in terms of x and p ?
5. Name the two standardized pseudo-random number generators

Bonus:

What are the two objectives when designing a pseudo-random number generator ?