CS557: Cryptography

Random number

S. Tripathy
IIT Patna

- Term-project Update
  - Submission Deadline
    - 15th Oct 2024
    - Complete Report (including code and ppt) 15th Nov

Present Class

- Cryptography
  - Pseudo Random Bit Generator (PRBG)

- Cryptographically Secure PRBGs
  - Blum-Micali Generator
  - Blum-Blum-Shub Generator

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

- PRBG Test

Random number

- **Truly random** - is defined as exhibiting ``true'' randomness, such as
  - Noise in electrical circuits
  - Radio active decay.

- **Pseudorandom** - is defined as having the *appearance* of randomness, but nevertheless exhibiting a specific, repeatable pattern.
  - numbers calculated by a computer through a deterministic process, cannot, by definition, be random

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

(Desirable) Properties of Pseudorandom Numbers

- **Uncorrelated Sequences -** The sequences of random numbers should be serially *uncorrelated*

- **Long Period** - The generator should be of *long period* (ideally, the generator should not repeat; practically, the repetition should occur only after the generation of a very large set of random numbers).

- **Uniformity** - The sequence of random numbers should be uniform, and unbiased. That is, equal fractions of random numbers should fall into equal ``areas'' in space.
  - Eg. if random numbers on [0,1) are to be generated, it would be poor practice were more than half to fall into [0, 0.1), presuming the sample size is sufficiently large.

- **Efficiency** - The generator should be efficient. Low overhead for massively parallel computations.

Pseudo-random Bit Generator

- Pseudo-random bit *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
  - Portable
  - Large period
  - Secure (uniform and independent)

Cryptographically Secure

- Passing all polynomial-time statistical tests
  - Probability distributions is indistinguishable
    - 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

- Passing the next-bit test
  - Next-bit is unpredictable
    - 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

Linear Congruential Generator -

- Algorithm :

  Based on the linear recurrence:

  $x_i = a\, x_{i-1} + b \mod m \qquad i \geq 1$

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 $x_n = 3\,x_{n-1} + 5 \bmod 31 \quad n \geq 1$, 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:
  - 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, 30,2

  - When $x_0 = 3$
  - ……………………………………

- Fast, but insecure
  - Sensitive to the choice of parameters a, b, and m
  - Serial correlation between successive values
  - Short period,

Pseudo-random sequences of 10 bits

when $x_0 = 2$

1101010001

Cryptographically Secure PRGs

- A PRG 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 PRBG if and only if there is a one-way function

- One-way functions
  - The RSA function
  - The discrete logarithm functio
  - The squaring function

Cryptographically secure PRGs

RSA Generator

Blum-Micali Generator

Blum-Blum-Shub Generator

RSA Generator - Algorithm

- Based on the RSA one-way function:
  - $x_i = x_{i-1}^e \mod n$ $\quad i \geq 1$

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

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

RSA Generator is relatively slow

RSA Generator - Efficiency

- 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 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 $j=c(\log \log n)$ and c is a constant

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

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

Output

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

$y_i = 1 \quad \text{if } x_i \geq (p-1)/2$

$y_i = 0 \quad \text{otherwise}$

$Y = (y_1 y_2 \ldots y_k)$ ← 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)

- But inefficient

  Modular exponentiation

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

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

- 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

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

Standardized PRNGs

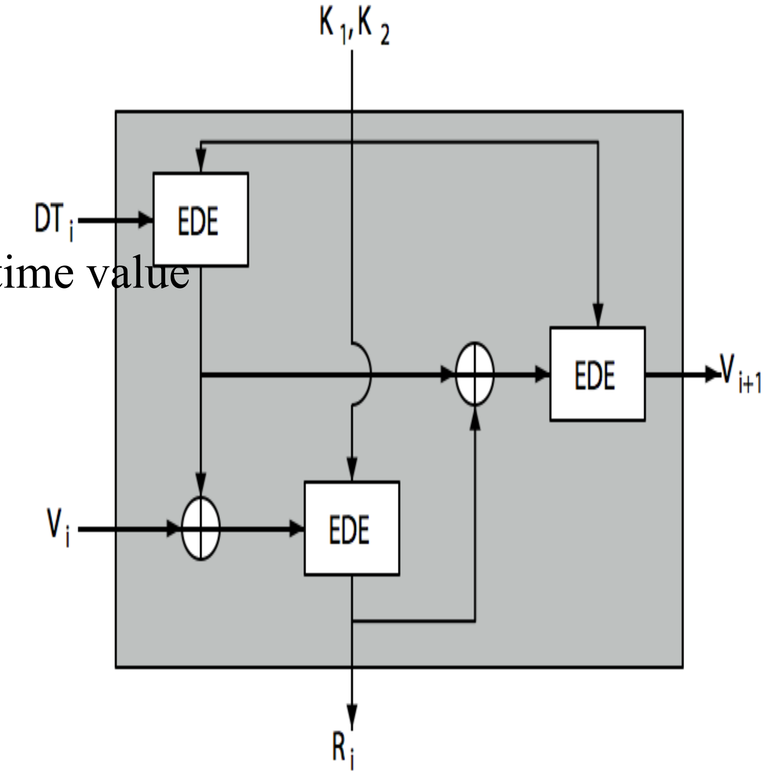
- General characteristics
  - Not been proven to be cryptographically secure
  - Sufficient for most applications
  - Using one-way functions such as hash function SHA-1 or block cipher DES with secret key k
- Examples
  - ANSI X9.17 Generator
  - FIPS 186 Generator

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

    $R_i = E_k(I \text{ XOR } V_i)$

    $V_{i+1} = E_k(R_i \text{ XOR } I)$
  - Return $(R_1, R_2, \ldots R_m)$ ← m pseudo-random 64-bit strings
- Used as an initialization vector or a key for DES



Classes of Attacks on PRNGs

- Direct Cryptanalytic Attack:
  - When the attacker can directly distinguish between PRNG numbers and random numbers (cryptanalyze the PRNG).
- Input Based Attack:
  - When the attacker is able to use knowledge and/ or control of PRNG inputs to cryptanalyze the PRNG.
- State Compromise Extension Attacks:
  - When the attacker can guess some information due to an earlier breach of security. The advantage of a previous attack is extended.

- Thanks