

Pseudo-Random Numbers and Stream Ciphers

CSS322: Security and Cryptography

Sirindhorn International Institute of Technology
Thammasat University

Prepared by Steven Gordon on 28 October 2013
css322y13s2l05, Steve/Courses/2013/s2/css322/lectures/random.tex, r2965

Contents

Principles of Pseudo-Random Number Generation

Pseudo-Random Number Generators

PRNGs using Block Ciphers

Stream Ciphers

RC4

Random Numbers

Use of Random Numbers

- ▶ Key distribution and authentication schemes
- ▶ Generation of session keys or keys for RSA
- ▶ Generation of bit stream for stream ciphers

Randomness

- ▶ Uniform distribution: frequency of occurrence of 1's and 0's approximately equal
- ▶ Independence: no sub-sequence can be inferred from others

Unpredictability

- ▶ Hard to predict next value in sequence

TRNG, PRNG and PRF

True Random Number Generator

- ▶ Non-deterministic source, physical environment
- ▶ Detect ionizing radiation events, leaky capacitors, thermal noise from resistors or audio inputs
- ▶ Mouse/keyboard activity, I/O operations, interrupts
- ▶ Inconvenient, small number of values

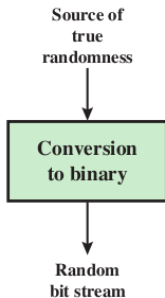
Pseudo Random Number Generator

- ▶ Deterministic algorithms to calculate numbers in “relatively random” sequence
- ▶ **Seed** is algorithm input
- ▶ Produces continuous stream of random bits

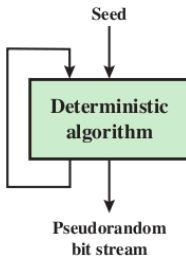
Pseudo Random Function

- ▶ Same as PRNG but produces string of bits of some

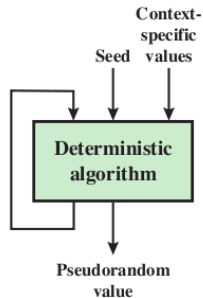
Random and Pseudo-Random Number Generators



(a) TRNG



(b) PRNG



(c) PRF

Requirements of PRNG

Hard to determine pseudo-random stream if don't know seed
(but know algorithm)

- ▶ Randomness
 - ▶ Test for uniformity, scalability, consistency
 - ▶ Examples: Frequency, runs, compressability
- ▶ Unpredictability
 - ▶ Forward and backward unpredictability
- ▶ Seed must be secure
 - ▶ Use TRNG to generate seed

Generation of Seed Input to PRNG

Random Numbers

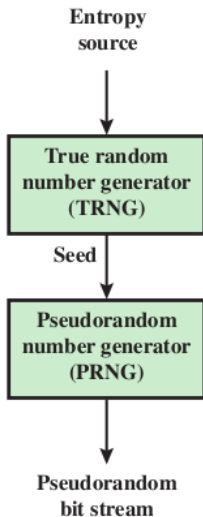
Principles

PRNGs

PRNG+Block

Stream Ciphers

RC4



Contents

Principles of Pseudo-Random Number Generation

Pseudo-Random Number Generators

PRNGs using Block Ciphers

Stream Ciphers

RC4

Linear Congruential Generator

Parameters:

- ▶ m , the modulus, $m > 0$
- ▶ a , the multiplier, $0 < a < m$
- ▶ c , the increment, $0 \leq c < m$
- ▶ X_0 , the seed, $0 \leq X_0 < m$

Generate sequence of pseudo-random numbers, $\{X_n\}$:

$$X_{n+1} = (aX_n + c) \bmod m$$

Choice of a , c and m is important:

- ▶ m should be large, prime, e.g. $2^{31} - 1$
- ▶ If $c=0$, few good values of a , e.g. $7^5 = 16807$

If attacker knows parameters and one number, can easily determine subsequent numbers

Blum Blum Shub Generator

Parameters:

- ▶ p, q : large prime numbers such that $p \equiv q \equiv 3 \pmod{4}$
- ▶ $n = p \times q$
- ▶ s , random number relatively prime to n

Generate sequence of bits, B_j :

$$X_0 = s^2 \pmod{n}$$

$$\text{for } i = 1 \rightarrow \infty$$

$$X_i = (X_{i-1})^2 \pmod{n}$$

$$B_i = X_i \pmod{2}$$

Cryptographically secure pseudo-random bit generator

Example Operation of BBS Generator

Random Numbers

Principles

PRNGs

PRNG+Block

Stream Ciphers

RC4

$$n = 192649 = 383 \times 503, s = 101355$$

i	X_i	B_i
0	20749	
1	143135	1
2	177671	1
3	97048	0
4	89992	0
5	174051	1
6	80649	1
7	45663	1
8	69442	0
9	186894	0
10	177046	0

i	X_i	B_i
11	137922	0
12	123175	1
13	8630	0
14	114386	0
15	14863	1
16	133015	1
17	106065	1
18	45870	0
19	137171	1
20	48060	0

Contents

Principles of Pseudo-Random Number Generation

Pseudo-Random Number Generators

PRNGs using Block Ciphers

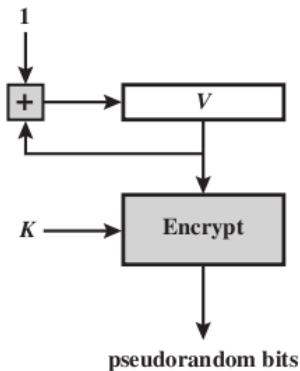
Stream Ciphers

RC4

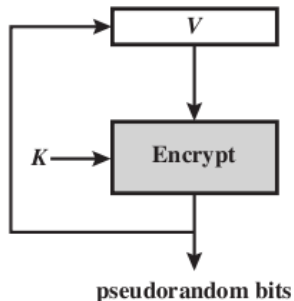
PRNG Mechanisms Based on Block Ciphers

Use symmetric block ciphers (e.g. AES, DES) to produce pseudo-random bits

- ▶ Seed is encryption key, K , and value V (which is updated)



Counter Mode



OFB Mode

ANSI X9.17 PRNG

Cryptographically secure PRNG using Triple DES
Parameters:

- ▶ 64-bit date/time representation, DT_i ;
- ▶ 64-bit seed value, V_i ;
- ▶ Pair of 56-bit DES keys, K_1 and K_2

Operation:

- ▶ Uses Triple DES three times
- ▶ (see next slide)

Output:

- ▶ 64-bit pseudo-random number, R_i ;
- ▶ 64-bit seed value, V_{i+1}

ANSI X9.17 PRNG

Random Numbers

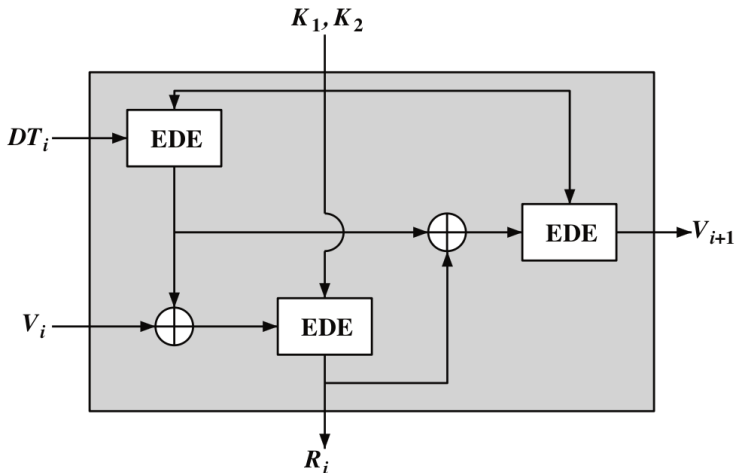
Principles

PRNGs

PRNG+Block

Stream Ciphers

RC4



Contents

Principles of Pseudo-Random Number Generation

Pseudo-Random Number Generators

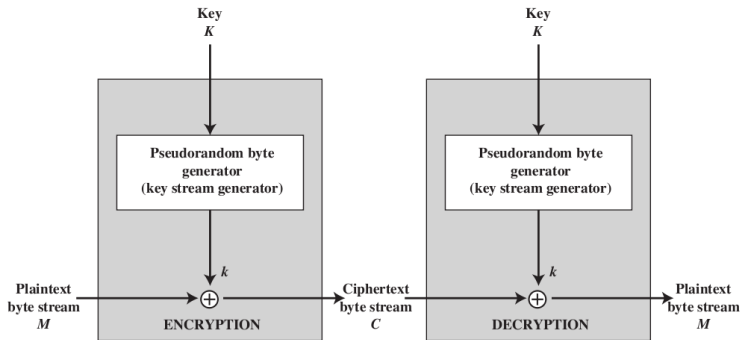
PRNGs using Block Ciphers

Stream Ciphers

RC4

Stream Ciphers

Encrypt one byte at a time by XOR with pseudo-random byte



Output of generator is called **keystream**

Design Criteria for Stream Ciphers

Important Considerations

- ▶ Encryption sequence should have large period
- ▶ Keystream should approximate true random number stream
- ▶ Key must withstand brute force attacks

Comparison to Block Ciphers

- ▶ Stream ciphers often simpler to implement, faster
- ▶ Block ciphers can re-use keys

Contents

Principles of Pseudo-Random Number Generation

Pseudo-Random Number Generators

PRNGs using Block Ciphers

Stream Ciphers

RC4

RC4

Random Numbers

Principles

PRNGs

PRNG+Block

Stream Ciphers

RC4

- ▶ Designed by Ron Rivest in 1987
- ▶ Used in secure web browsing and wireless LANs
- ▶ Very simple and efficient implementation
- ▶ Can use variable size key: 8 to 2048 bits
- ▶ Several theoretical limitations of RC4
 - ▶ No known attacks if use 128-bit key and discard initial values of stream
 - ▶ RC4 is used in WEP (shown to be weak security for wireless LANs)—problem with how keys are used, not RC4 algorithm

RC4 Algorithm

Parameters and Variables

- ▶ Variable length key, K , from 1 to 256 Bytes
- ▶ State vector, S , 256 Bytes
- ▶ Temporary vector, T , 256 Bytes
- ▶ A byte from keystream, k , generated from S

Steps

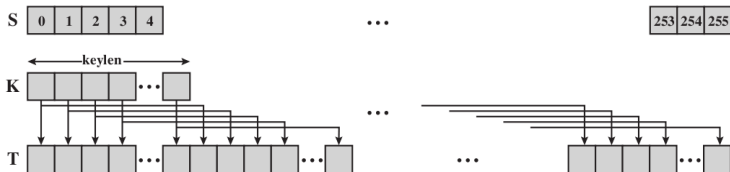
1. Initialise S to values 0 to 255; initialise T with repeating values of key, K
2. Use T to create initial permutation of S
3. Permute S and generate keystream, k from S
4. Encrypt a byte of plaintext, p , by XOR with k

Initial State of S and T

```

for i = 0 to 255 do
  S[i] = i;
  T[i] = K[i mod keylen];

```

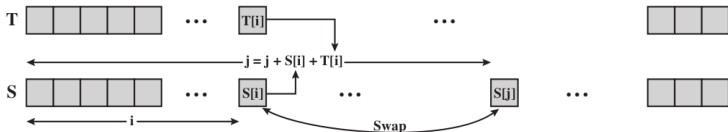


Initial Permutation of S

```

j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    Swap (S[i], S[j]);

```



Stream Generation

Random Numbers

Principles

PRNGs

PRNG+Block

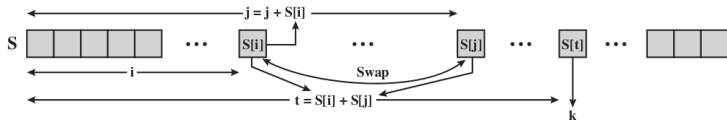
Stream Ciphers

RC4

```

i, j = 0;
while (true)
  i = (i + 1) mod 256;
  j = (j + S[i]) mod 256;
  Swap (S[i], S[j]);
  t = (S[i] + S[j]) mod 256;
  k = S[t];

```



To encrypt: $C = p \text{ XOR } k$

To decrypt: $p = C \text{ XOR } k$