

ECE 443/518 – Computer Cyber Security

Lecture 03 Stream Ciphers

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

August 25, 2025

Outline

One-Time Pad

Random Number Generators

Stream Ciphers

Reading Assignment

- ▶ This lecture: UC 2
- ▶ Next lecture: UC 3, 4 except 4.3, 5.1 – 5.1.5

Outline

One-Time Pad

Random Number Generators

Stream Ciphers

Overview: The Substitution Cipher

- ▶ Large key space helps to resist brute-force from computationally bounded passive adversaries.
- ▶ Effective cryptanalysis methods exist because ciphertext leaks statistics of plaintext.
- ▶ If a cipher could resist brute-force from computationally unbounded passive adversaries, will it also resist any cryptanalysis method?
 - ▶ Including those cryptanalysis methods designed by someone really smart in future?
- ▶ Unconditional security
 - ▶ A.k.a. information-theoretically secure
 - ▶ If a cryptosystem cannot be broken even with infinite computational resources.

What does “meaningful” mean?

Given y , $e()$, and $d()$, find x and k such that:

$$y = e_k(x), \text{ and } x = d_k(y).$$

- ▶ Consider ciphertext-only attack.
- ▶ What if there are many k 's such that $x = d_k(y)$ is meaningful?

One-Time Pad (OTP)

- ▶ Plaintext: $x = x_0, x_1, \dots$, where $x_j \in \{0, 1, \dots, N - 1\}$.
- ▶ Key: $k = k_0, k_1, \dots$, where $k_j \in \{0, 1, \dots, N - 1\}$.
 - ▶ Choose a key that is of the same length as the message.
- ▶ Ciphertext: $y = y_0, y_1, \dots$, where $y_j \in \{0, 1, \dots, N - 1\}$.
- ▶ $e()$: $y = e_k(x)$ where $y_j = (x_j + k_j) \bmod N$.
 - ▶ For N being power of 2, e.g. bytes, using xor is also popular.
- ▶ $d()$: $x = d_k(y)$ where $x_j = (y_j - k_j) \bmod N$.
- ▶ Indistinguishable plaintext
 - ▶ For any $y = e_k(x)$, there exists x' and k' such that $x' = d_{k'}(y)$.
 - ▶ So the adversary cannot tell whether the actual plaintext is x or x' .

OTP and Unconditional Security

- ▶ For unconditional security, the key k should be chosen uniformly from the key space.
 - ▶ Otherwise adversaries may learn that some plaintexts are more probable than others, eventually breaking the cryptosystem.
- ▶ One-Time
 - ▶ For different messages, when the key space is large enough, very unlikely you'll generate the same k twice for uniform distribution.
 - ▶ If you reuse k for the messages with the same length and the adversaries know that, then they can learn correlations among plaintext from correlations among ciphertext, potentially learning even more.

Practical Considerations

- ▶ Key establishment
 - ▶ Need a random key for every message.
 - ▶ Size of each random key is the same as each message.
- ▶ If Alice and Bob have a secure channel to communicate these keys, why don't they just use it to send messages?
- ▶ Pre-shared random bits
 - ▶ Work for finite number of messages
- ▶ How to generate random bits?
- ▶ Can we generate more random bits from some random "seeds" deterministically?
 - ▶ So Alice and Bob can get more key bits from existing key bits?

Outline

One-Time Pad

Random Number Generators

Stream Ciphers

True Random Number Generators (TRNG)

- ▶ True random number generators: output cannot be reproduced.
 - ▶ Via a random physical process, e.g. flipping a fair coin multiple times.
- ▶ Yes, computers can collect/generate true random bits.
 - ▶ Special TRNG devices: semiconductor noise, clock jitter, radioactive decay, etc.
 - ▶ Software measurements: delay variation between events, e.g. network packets and user inputs.
 - ▶ Concerns: speed, correlation between neighboring measurements.
- ▶ No, we can't generate more true random bits from some random "seeds" deterministically.
 - ▶ By definition of true random number.

Pseudorandom Number Generators (PRNG)

- ▶ Pseudorandom number generators: generate sequences using a seed deterministically, usually via a function f ,
$$s_0 = \text{seed}, s_{i+1} = f(s_i, s_{i-1}, \dots).$$
 - ▶ Statistically similar to true random sequences.
 - ▶ Reproducible.
 - ▶ Widely used for simulation and testing.
- ▶ Most are predictable: one can derive the seed by observing a sub-sequence, and then predict what comes next.
 - ▶ Not suitable for use in cryptosystem where the seed should be a secret.
 - ▶ A major source of weakness for homebrew cryptosystems.
- ▶ Cryptosystem need to use unpredictable cryptographically secure pseudorandom number generators (CSPRNG).

Outline

One-Time Pad

Random Number Generators

Stream Ciphers

Stream Ciphers

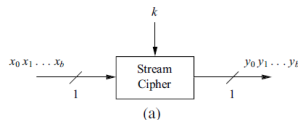


Fig. 2.2 Principles of encrypting b bits with a stream (a)

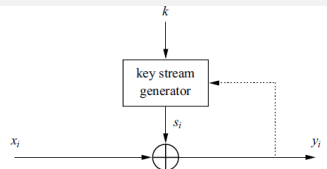


Fig. 2.3 Synchronous and asynchronous stream ciphers

(Paar and Pelzl)

- ▶ Encode plaintext x and ciphertext y both as binary strings.
- ▶ Generate a key stream s from the secret key k .
 - ▶ Synchronous: s depends only on k .
 - ▶ Asynchronous: s depends on both k and x
- ▶ Usually use xor \oplus to encrypt x into y using s .
 - ▶ Same function for both encryption and decryption.
 - ▶ Allow to process x , y , and s as blocks of bits.

(Synchronous) Stream Ciphers

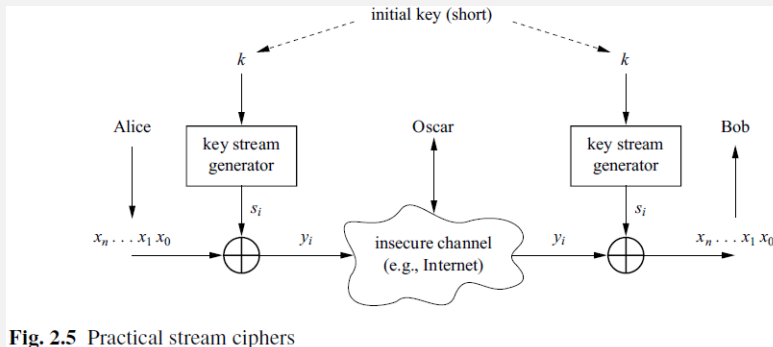


Fig. 2.5 Practical stream ciphers

(Paar and Pelzl)

- ▶ What's the difference between stream ciphers and OTP?
- ▶ What's the danger to NOT use CSPRNG for the key stream generator?
- ▶ If Alice want to send a second message to Bob using the same key K , should she restart the key stream generator?

Problem Formulation: Known-Plaintext Attack (KPA)

Given y , $e()$, $d()$, and x , find k such that:

$$y = e_k(x), \text{ and } x = d_k(y).$$

- ▶ Are some ciphers stronger than others?
 - ▶ We generally ignore brute-force as most ciphers can use a larger key space to resist brute-force.
 - ▶ Instead, we consider attacks that gives attackers more information.
- ▶ Compared to ciphertext-only attack, known-plaintext attack
 - ▶ Allow attackers to know plaintext, e.g. from packet headers, file headers, etc.
 - ▶ Allow attackers to find k without defining what are “meaningful” plaintexts.
- ▶ Ciphers that can resist known-plaintext attack are stronger than those that can only resist ciphertext-only attack.

Known-Plaintext Attack and CSPRNG

- ▶ With known-plaintext attack, attackers are able to recover part of the key stream without brute-force.
- ▶ For OTP, attackers cannot derive key streams for other messages because they are truly random.
- ▶ For stream ciphers, CSPRNG should resist known-plaintext attack.
 - ▶ Therefore, attackers cannot derive future key stream with known parts of the key stream.

How to design a CSPRNG?

- ▶ Can we prove that a PRNG is a CSPRNG?
 - ▶ A very hard question.
 - ▶ At least we could show some PRNGs are not CSPRNG because they cannot resist known-plaintext attack.

Linear Congruential Generator is NOT CSPRNG

$$S_0 = \text{seed},$$

...

$$S_{i+1} \equiv AS_i + B \pmod{m},$$

$$S_{i+2} \equiv AS_{i+1} + B \pmod{m},$$

...

- ▶ A widely used software PRNG.
- ▶ $k = (\text{seed}, A, B)$: secret.
- ▶ m : known cryptosystem parameter.
- ▶ S_i, S_{i+1}, S_{i+2} : consecutive blocks of bits in the key stream
- ▶ Possible to solve for A and B if S_i, S_{i+1}, S_{i+2} are obtained via known-plaintext attacks, and then to predict future S .

LFSR is NOT CSPRNG

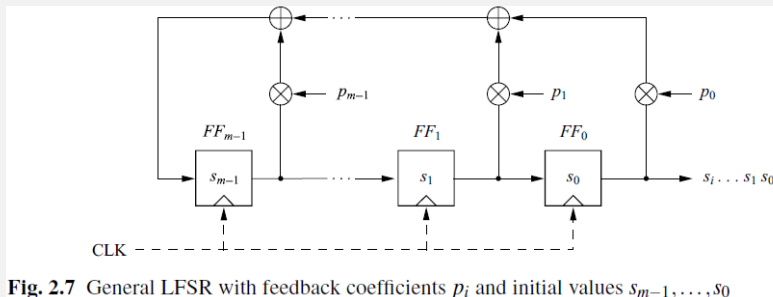


Fig. 2.7 General LFSR with feedback coefficients p_i and initial values s_{m-1}, \dots, s_0
(Paar and Pelzl)

$$s_{i+m} \equiv s_{i+m-1}p_{m-1} + \dots + s_{i+1}p_1 + s_i p_0 \pmod{2}.$$

- ▶ A widely used hardware PRNG: \oplus for xor, \otimes for and
- ▶ $k = (p_0, p_1, \dots, p_{m-1})$: secret.
- ▶ Possible to solve for p_0, p_1, \dots, p_{m-1} if $2m$ consecutive bits of s are obtained via known-plaintext attacks.

Summary

- ▶ One-time pad and unconditional security
- ▶ Stream ciphers and CSPRNG