CS 6530 Applied Cryptography

July-Nov 2025

Introduction to Cryptography and Data Security

04th August 2025 – Session 2, 05th August 2025

Dr. Manikantan Srinivasan

(Material covered is based on Chapter 2 of

Understanding Cryptography – Second Edition

Courtesy: Slides by Authors - Christof Paar and Jan Pelzl)

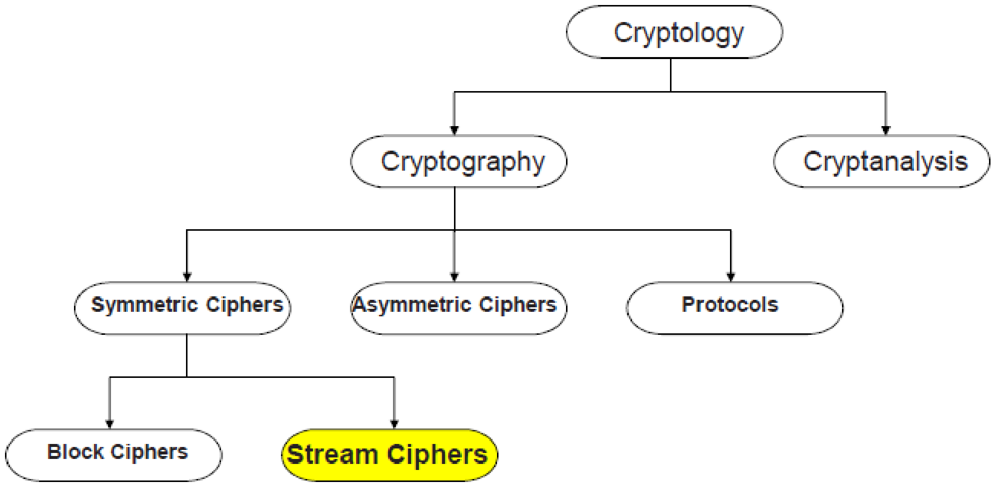
Contents of Chapter 2

- **Introduction to Stream Ciphers**
- Random Number Generators (RNGs)
- One-Time-Pad (OTP)
- Salsa20, ChaCha20
- Trivium

Introduction to Stream Ciphers

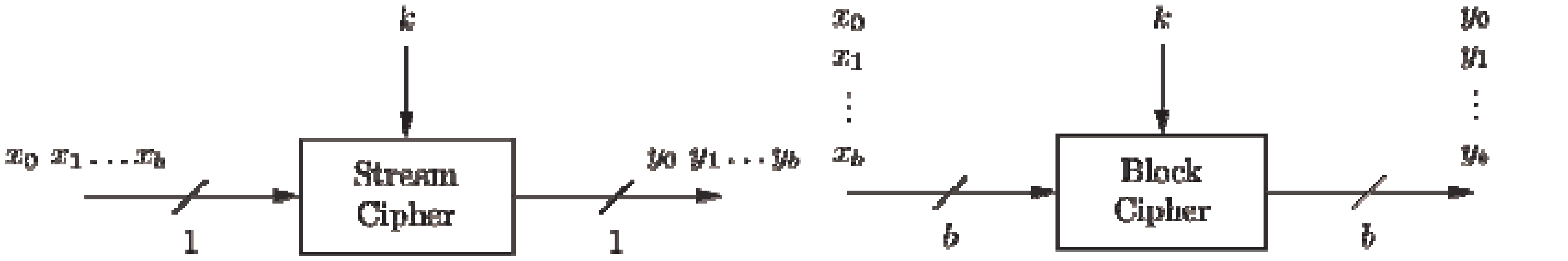
Stream Ciphers in the Field of Cryptology

- Symmetric Ciphers Divided into two families – Stream Ciphers and Block Ciphers
- Stream Ciphers were invented in 1917 by Gilbert Vernam



- Symmetric Ciphers encrypts bits individually. This is achieved by adding a bit from a *key stream* to a plaintext bit
- Syncrhonous stream ciphers -> key stream depends only on the key
- Asyncrhonous stream ciphers -> key stream also depends on the cipher text (Example Cipher feedback (CFB) mode
- Stream Ciphers were invented in 1917 by Gilbert Vernam

Stream Cipher vs. Block Cipher

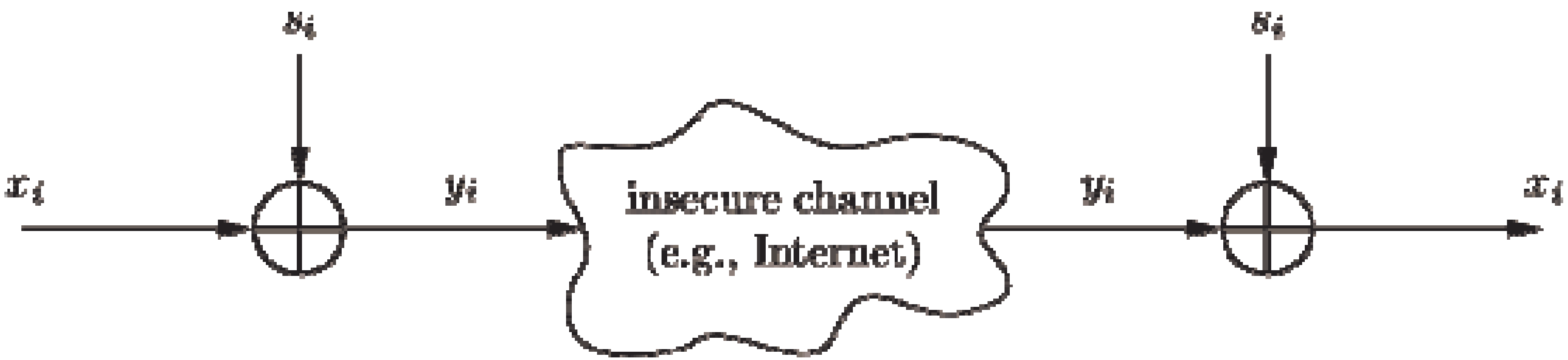


- **Stream Ciphers**
  - Encrypt bits individually
  - Usually small and fast → common in embedded devices (e.g., A5/1 for GSM phones)

- **Block Ciphers:**
  - Always encrypt a full block (several bits)
  - Are common for Internet applications

Encryption and Decryption with Stream Ciphers

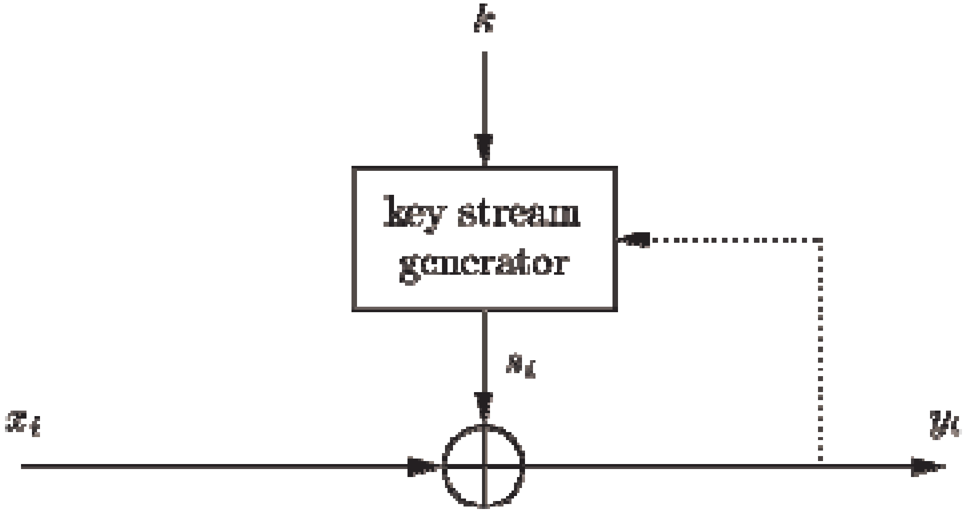
Plaintext x_i, ciphertext y_i and key stream s_i consist of individual bits



- Encryption and decryption are simple additions modulo 2 (aka XOR)
- Encryption and decryption are the same functions

- **Encryption:** $y_i = e_{si}(x_i) = x_i + s_i \bmod 2$ $\quad\quad x_i, y_i, s_i \in \{0,1\}$
- **Decryption:** $x_i = e_{si}(y_i) = y_i + s_i \bmod 2$

Synchronous vs. Asynchronous Stream Cipher



- Security of stream cipher depends entirely on the key stream s_i :
  - Should be **random** , i.e., $\Pr(s_i = 0) = \Pr(s_i = 1) = 0.5$
  - Must be **reproducible** by sender and receiver

- **Synchronous Stream Cipher**
  - Key stream depend only on the key (and possibly an initialization vector IV)

- **Asynchronous Stream Ciphers**
  - Key stream depends also on the ciphertext (dotted feedback enabled)

Why is Modulo 2 Addition a Good Encryption Function?

- Modulo 2 addition is equivalent to XOR operation
- For perfectly random key stream s_i , each ciphertext output bit has a 50% chance to be 0 or 1

  → Good statistic property for ciphertext
- Inverting XOR is simple, since it is the same XOR operation

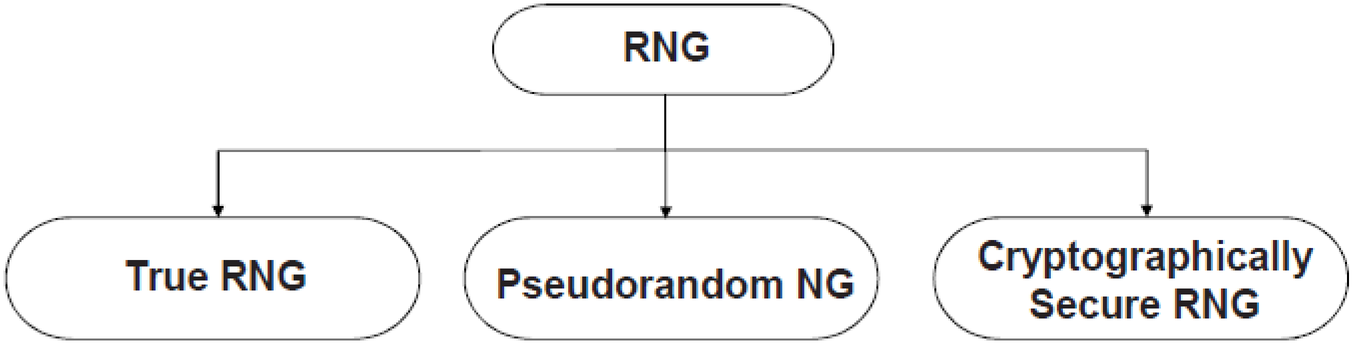
x_i	s_i	y_i
0	0	0
0	1	1
1	0	1
1	1	0

What Exactly is the Nature of the Key Stream

- Generation of the values S_i, which are called Key Stream, is the central issue for the security of the stream ciphers.
- Security of Stream Cipher completely depends on the key stream
- The key stream bits S_i are not the key bits themselves.
- Stream ciphers are about generation of Key Stream.
- Central requirement for the key stream bits should be that they appear like a random sequence to an attacker.

Random Numbers and unbreakable Stream Cipher

Random number generators (RNGs)



True Random Number Generators (TRNGs)

- Based on physical random processes: coin flipping, dice rolling, semiconductor noise, radioactive decay, mouse movement, clock jitter of digital circuits
- Output stream s_i should have good statistical properties:
  $Pr(s_i = 0) = Pr(s_i = 1) = 50\%$ (often achieved by post-processing)
- Output can neither be predicted nor be reproduced

Typically used for generation of keys, nonces (used only-once values) and for many other purposes

Pseudorandom Number Generator (PRNG)

- Generate sequences from initial seed value
- Typically, output stream has good statistical properties
- Output can be reproduced and can be predicted

Often computed in a recursive way:

$$s_0 = seed$$

$$s_{i+1} = f(s_i, s_{i-1}, \ldots, s_{i-t})$$

Example: *rand() function* in ANSI C:

$$s_0 = 12345$$

$$s_{i+1} = 1103515245 s_i + 12345 \bmod 2^{31}$$

Most PRNGs have bad cryptographic properties!

Cryptanalyzing a Simple PRNG

Simple PRNG: **Linear Congruential Generator**

$$S_0 = seed$$

$$S_{i+1} = AS_i + B \bmod m$$

Assume

- unknown A, B *and* S_0 as key
- Size of A, B and S_i to be 100 bit
- 300 bit of output are known, i.e. S_1, S_2 and S_3

Solving

$$S_2 = AS_1 + B \bmod m$$

$$S_3 = AS_2 + B \bmod m$$

…directly reveals A and B. All S_i can be computed easily!

Bad cryptographic properties due to the linearity of most PRNGs

Cryptographically Secure Pseudorandom Number Generator (CSPRNG)

- Special PRNG with additional property:
  - Output must be **unpredictable**

More precisely: Given n consecutive bits of output s_i , the following output bits s_{n+1} cannot be predicted (in polynomial time).

- Needed in cryptography, in particular for stream ciphers

- Remark: There are almost no other applications that need unpredictability, whereas many, many (technical) systems need PRNGs.

One-Time Pad (OTP)

Unconditionally secure cryptosystem:

- A cryptosystem is unconditionally secure if it cannot be broken even with *infinite* computational resources

One-Time Pad

- A cryptosystem developed by Mauborgne that is based on Vernam's stream cipher:
- Properties:

  Let the plaintext, ciphertext and key consist of individual bits $x_i, y_i, k_i \in \{0,1\}$.

Encryption:	$e_{k_i}(x_i) = x_i \oplus k_i$.
Decryption:	$d_{k_i}(y_i) = y_i \oplus k_i$

OTP is unconditionally secure if and only if the key k_i. is used once!

OTP Definition: A Stream cipher for which

- The key stream is S_0, S_1, S_2, … is generated by a true random number generator, and
- The Key Stream is known only to the legitimate parties., and
- Every key stream bit S_i is only used once.

One-Time Pad (OTP)

Unconditionally secure cryptosystem:

$$y_0 = x_0 \oplus k_0$$

$$y_1 = x_1 \oplus k_1$$

$$\vdots$$

Every equation is a linear equation with two unknowns

$\Rightarrow$ for every y_i are $x_i = 0$ and $x_i = 1$ equiprobable!

$\Rightarrow$ This is true iff k_0, k_1, ... are independent, i.e., all k_i have to be generated truly random

$\Rightarrow$ It can be shown that this systems can *provably* not be solved.

Disadvantage: For almost all applications the OTP is **impractical** since the key must be as long as the message! (Imagine you have to encrypt a 1GByte email attachment.)

A Problem

- The OTP can be used to encrypt data of arbitrary length by encrypting binary symbols i.e. $x_i = 0$ or $x_i = 1$. Decrypt the following cipher text by hand. The ciper text is given in hexa decimal notation

  26 34 05 18 0c 06 07 15 1c 2a 13 3c 0c 23 04 27 07 27 18

- The key is given by:

  6a 51 71 6b 49 68 64 67 65 5a 67 68 64 4a 77 65 68 48 73

Thank you – Q&A