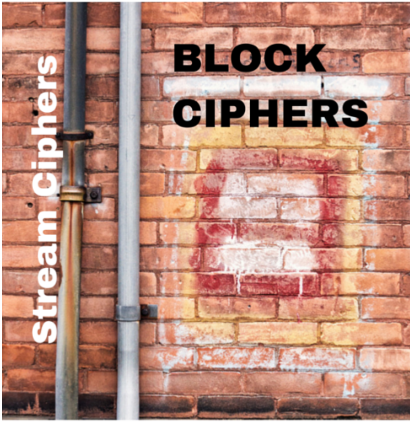


CS 553
CRYPTOGRAPHY

Lecture 16
Stream Ciphers

Instructor
Dr. Dhiman Saha

Image Source: Google

Encrypts a stream of bits. How long?

By producing a **pseudorandom** stream of bits called the **keystream**

Stream Ciphers from Block Ciphers

- **Block ciphers** being used in a particular **mode of operation** behave like a stream cipher
- CFB, OFB, CTR

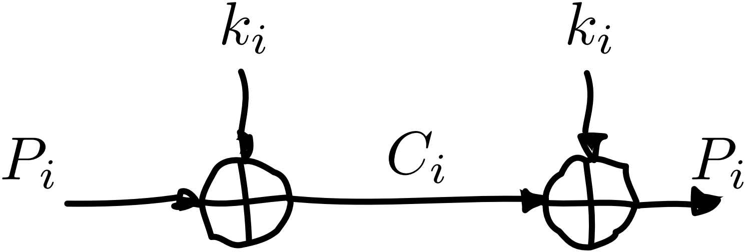
Stream Ciphers

Encrypts a stream of bits. How long?

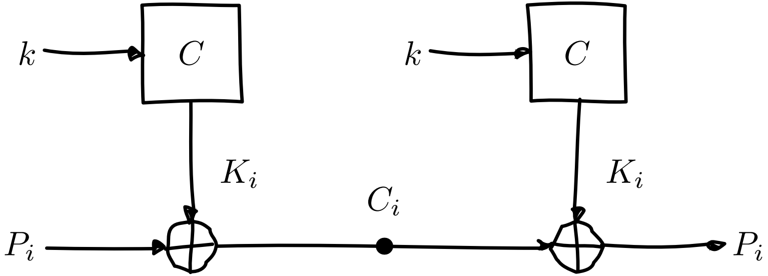
By producing a **pseudorandom** stream of bits called the **keystream** △

Stream Ciphers from Block Ciphers

- **Block ciphers** being used in a particular **mode of operation** behave like a stream cipher
- CFB, OFB, CTR


k_i
k_i
P_i
C_i
P_i

- Designed from the ground up to be a stream cipher.



- Construction looks quite similar to a one-time pad,
- Except that key-stream for OTP is truly random while here K_i is **pseudorandom**

Properties

- No error propagation
- Speed
- On-the-fly encryption
- Implementation efficiency
- One main issue: Need for synchronisation

Performance: Stream vs. block ciphers

Crypto++ 5.6.0 [Wei Dai]

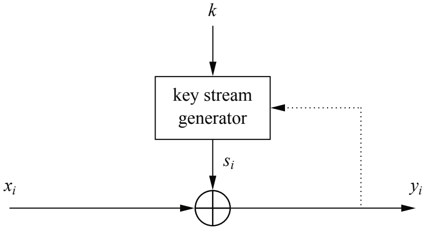
AMD Opteron, 2.2 GHz (Linux)

	Cipher	Block/key size	Throughput [MB/s]
Stream	RC4		126
	Salsa20/12		643
	Sosemanuk		727
Block	3DES	64/168	13
	AES128	128/128	109

Image Source: Google

Synchronous Vs Asynchronous

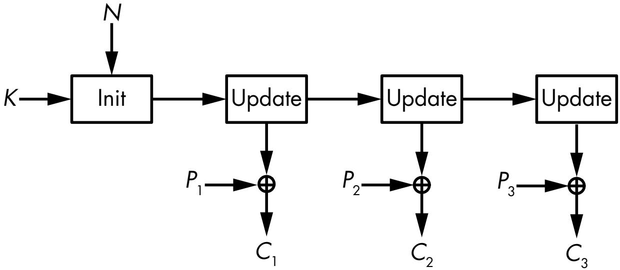
- The most common type of stream cipher is called a **synchronous** stream cipher.
- These algorithms produce a long stream of pseudorandom bits from a secret symmetric key.

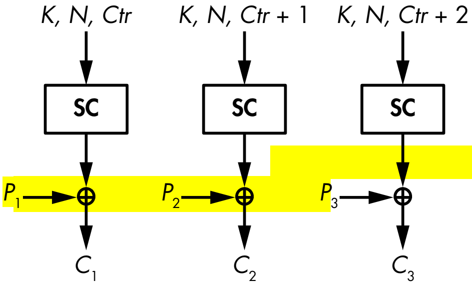


- There are also **asynchronous** or **self-synchronizing** stream ciphers,
- Where the previously produced ciphertext bits are used to produce the current keystream bit.

Stateful Vs Counter-Based

Stateful
(RC4)





Counter-Based
Salsa20

Trivium — Hardware Oriented

- Dedicated Hardware
- ASICs, PLDs, and FPGAs
- Notion: A ciphers hardware implementation is an electronic circuit that implements the cryptographic algorithm at the bit level and that **cannot** be used for anything else.

Rabbit — Software Oriented

- General purpose software implementation
- Using microprocessor level instructions
- These instructions operate on bytes or words and then call pieces of electronic circuit that implement general-purpose operations such as addition and multiplication

Hardware Oriented Vs Software Oriented

Trivium — Hardware Oriented

- Dedicated Hardware
- ASICs, PLDs, and FPGAs
- Notion: A ciphers hardware implementation is an electronic circuit that implements the cryptographic algorithm at the bit level and that **cannot** be used for anything else.

Rabbit — Software Oriented

- General purpose software implementation
- Using microprocessor level instructions
- These instructions operate on bytes or words and then call pieces of electronic circuit that implement general-purpose operations such as addition and multiplication


ECRYPT II

eSTREAM portfolio

Home

Profile 1 (SW)

HC-128

Rabbit

Salsa20/12

SOSEMANUK

Profile 2 (HW)

Grain v1

MICKEY 2.0

Trivium

eSTREAM: the ECRYPT Stream Cipher Project

Welcome to the home page of eSTREAM, the ECRYPT Stream Cipher Project. The eSTREAM project was a multi-year effort, running from 2004 to 2008, to promote the design of efficient and compact stream ciphers suitable for widespread adoption. As a result of the project, a portfolio of new stream ciphers was announced in April 2008. The eSTREAM portfolio was revised in September 2008, and currently contains seven stream ciphers. This website is dedicated to ciphers in this final portfolio. For information on the eSTREAM *project* and selection process, including a timetable of the project and further technical background, please visit the original eSTREAM Project website.

The eSTREAM Portfolio

The short report from April 2008 discussing the initial portfolio (with eight stream ciphers) and the end of the eSTREAM project can be found here. The eSTREAM portfolio was revised in September 2008, following the announcement of cryptanalytic results against one of the original algorithms (see here). The portfolio is periodically revisited, as the algorithms mature: the first review of the eSTREAM portfolio was published in October 2009, and is available here; the second review from January 2012 can be found here.

The eSTREAM portfolio ciphers fall into two profiles. Profile 1 contains stream ciphers more suitable for software applications with high throughput requirements. Profile 2 stream ciphers are particularly suitable for hardware applications with restricted resources such as limited storage, gate count, or power consumption.

The eSTREAM portfolio contains the following ciphers:

Profile 1 (SW)	Profile 2 (HW)
HC-128	Grain v1
Rabbit	MICKEY 2.0
Salsa20/12	Trivium
SOSEMANUK	

`http://www.ecrypt.eu.org/stream/`

Random Number Generators

TRNG
PRNG
CSPRNG

- The security of stream ciphers hinges entirely on a **suitable** key stream
- **Randomness** plays a major role

True random number generators (TRNGs)

Characterized by the fact that their output cannot be reproduced

Example

- Coin flipping,
- Rolling of dice
- Semiconductor noise
- Clock jitter in digital circuits
- Radioactive decay

Pseudorandom number generators (PRNGs)

Generate sequences which are computed from an initial **seed** value

- Computed recursively

$$s_0 = seed$$
$$s_{i+1} = f(s_i), \quad i = 0, 1, \ldots$$

- Generalization

$$s_{i+1} = f(s_i, s_{i-1}, \cdots, s_{i-t}), \quad t \leftarrow \text{fixed integer}$$

Note:

- PRNGs are not random in a true sense
- Because they can be computed and are thus completely deterministic

The linear congruential generator

a, b, m are integer constants

$$s_0 = seed$$
$$s_{i+1} = a \cdot s_i + b \bmod m, \quad i = 0, 1, \ldots$$

- A widely used analog is the `rand()` function used in ANSI C

$$s_0 = 12345$$
$$s_{i+1} = 1103515245 s_i + 12345 \bmod 2^{31}, \quad i = 0, 1, \ldots$$

- PRNGs $\rightarrow$ Good statistical properties
- Applications outside crypto: VLSI testing, simulation

Is a stream cipher based on the linear congruential generator prone to known plaintext attack?

- Note: Just recovering some key-bits is not enough
- Attacker should be able to generate the stream as well

A special type of PRNG

- Possess the following additional property
- It is **unpredictable**

- Given n consecutive bits of the key stream $(s_1, \cdots, s_n)$, there is no polynomial time algorithm that can **predict** the next bit s_{n+1} with better than 50% chance of success.
- (**Difficult-to-invert**) computationally infeasible to compute any preceding bits

Cryptographically Secure PRNG (CSPRNG)

A special type of PRNG

- Possess the following additional property
- It is **unpredictable**

- Given n consecutive bits of the key stream $(s_1, \cdots, s_n)$, there is no polynomial time algorithm that can **predict** the next bit s_{n+1} with better than 50% chance of success.
- (**Difficult-to-invert**) computationally infeasible to compute any preceding bits

Cryptographically Secure PRNG (CSPRNG)

A special type of PRNG

- Possess the following additional property
- It is **unpredictable**

- Given n consecutive bits of the key stream $(s_1, \cdots, s_n)$, there is no polynomial time algorithm that can **predict** the next bit s_{n+1} with better than 50% chance of success.
- (**Difficult-to-invert**) computationally infeasible to compute any preceding bits

Feedback Shift Registers

A standard way of producing a binary stream of data is to use an FSR

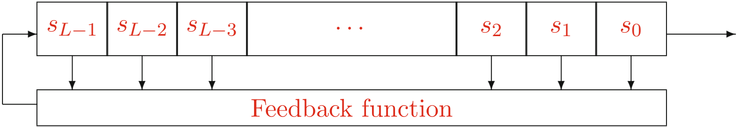
- These are small circuits containing a number of memory cells, each of which holds one bit of information.
- The set of such cells forms a register

s_{L-1}	s_{L-2}	s_{L-3}	$\cdots$	s_2	s_1	s_0

The feedback function

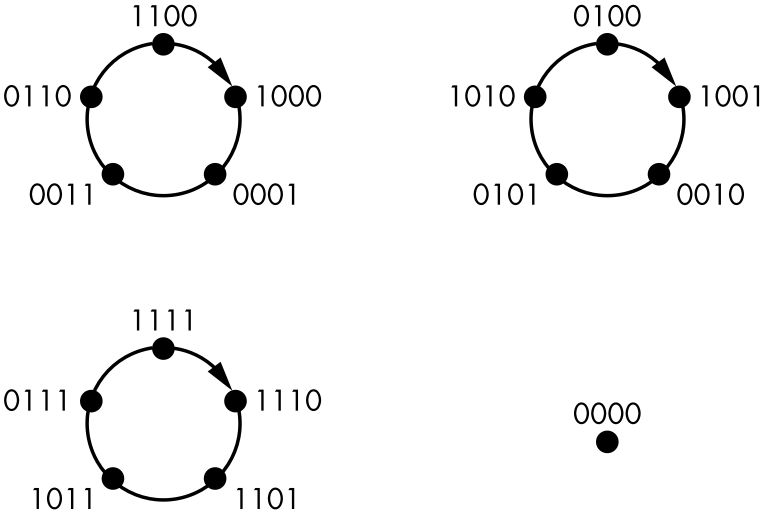
In each cycle a certain predefined set of cells are tapped and their value is passed through a function

- The register is then shifted down by one bit
- The output bit of the feedback shift register being the bit that is shifted out of the register
- The combination of the tapped bits is then fed into the empty cell at the top of the register.



- Let the feedback function be the XOR of all the bits.

FB = XOR the 4 bits together



Period

The period of an FSR, from some initial state, is the number of updates needed until the FSR enters the same state again.

Example

If the period of this FSR is 5, clocking the register 10 times will yield **twice** the **same** 5-bit sequence

- For use in a stream cipher, FSRs with with short periods should be avoided
- As they make the output more predictable
- Some types of FSRs make it easy to figure out their period
- But its almost impossible to do so with others

Period

The period of an FSR, from some initial state, is the number of updates needed until the FSR enters the same state again.

Example

If the period of this FSR is 5, clocking the register 10 times will yield **twice** the **same** 5-bit sequence

- For use in a stream cipher, FSRs with with short periods should be avoided
- As they make the output more predictable
- Some types of FSRs make it easy to figure out their period
- But its almost impossible to do so with others

Linear Feedback Shift Registers

Coming up in next lecture