Stream Cipher

CUI TINGTING

School of Cypberspace, Hangzhou Dianzi University

October 9, 2023

Contents

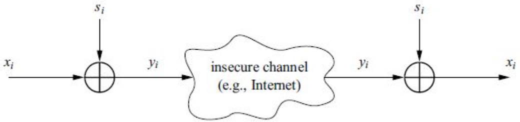
what is OTP

One-Time Pad (OTP)

A cipher for which

- Encryption where a keystream is bitwise added to plaintext
- Keystream is generated perfect randomly
- Keystream is only known to the legitimate communicating parties
- Every keystream bit k_i is only used once

is called a **one-time pad**.



OTP is Unconditionally Secure

Unconditional Security

A cryptosystem is unconditionally or information-theoretically secure if it cannot be broken even with infinite computational resources.

OTP is Unconditionally Secure

Unconditional Security

A cryptosystem is unconditionally or information-theoretically secure if it cannot be broken even with infinite computational resources.

- OTP is unconditionally secure.

OTP is Unconditionally Secure

Unconditional Security

A cryptosystem is unconditionally or information-theoretically secure if it cannot be broken even with infinite computational resources.

- OTP is unconditionally secure.
- OPT is impractice.

OTP is Unconditionally Secure

Unconditional Security

A cryptosystem is unconditionally or information-theoretically secure if it cannot be broken even with infinite computational resources.

- OTP is unconditionally secure.
- OPT is impractice.
  - A TRNG is needed to generate keystream.

OTP is Unconditionally Secure

Unconditional Security

A cryptosystem is unconditionally or information-theoretically secure if it cannot be broken even with infinite computational resources.

- OTP is unconditionally secure.
- OPT is impractice.
  - A TRNG is needed to generate keystream. NOT EASY
  - Seure Channel is needed to transform keystream.

OTP is Unconditionally Secure

Unconditional Security

A cryptosystem is unconditionally or information-theoretically secure if it cannot be broken even with infinite computational resources.

- OTP is unconditionally secure.
- OPT is impractice.
  - A TRNG is needed to generate keystream. NOT EASY
  - Seure Channel is needed to transform keystream.NOT EASY
  - The key is as long as the plaintext!

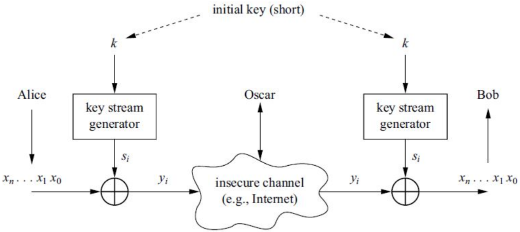
OTP is Unconditionally Secure

Unconditional Security

A cryptosystem is unconditionally or information-theoretically secure if it cannot be broken even with infinite computational resources.

- OTP is unconditionally secure.
- OPT is impractice.
  - A TRNG is needed to generate keystream. NOT EASY
  - Seure Channel is needed to transform keystream.NOT EASY
  - The key is as long as the plaintext! MAJOR DRAWBACK

Practical Stream Cipher



- Keystream is generated by PRNG
- Hope Stream Cipher is computational security

Computational Security

A cryptosystem is computationally secure if the best known algorithm for breaking it requires at least t operations.

True Random Number Generators (TRNG)

- Output CANNOT be predicted or reproduced. e.g. flip a coin 100 times.
- TRNGs are based on physical processes. e.g. coin flipping, rolling of dice, semiconductor noise, clock jitter in digital circuits and radioactive decay.
- TRNGs are often needed for generating session keys.

Pseudorandom Number Generators (PRNG)

Pseudorandom number generators (PRNGs)

- Generate sequences from an initial seed value.
- Often they are computed recursively:

$$
\begin{aligned}
s_0 =& seed, \\
s_{i+1} =& f(s_i, s_{i-1}, \ldots, s_{i-t}), i = 0, 1, \ldots.
\end{aligned}
$$

where t is a fixed integer.

- Typically, output stream has good statistical properties.
- Output can be reproduced and can be predicted.
- Most PRNGs have bad cryptographic properties!

Pseudorandom Number Generators (PRNG)

Pseudorandom number generators (PRNGs)

- Generate sequences from an initial seed value.
- Often they are computed recursively:

$$
\begin{aligned}
s_0 &= seed, \\
s_{i+1} &= f(s_i, s_{i-1}, \ldots, s_{i-t}), i = 0, 1, \ldots .
\end{aligned}
$$

where t is a fixed integer.

- Typically, output stream has good statistical properties.
- Output can be reproduced and can be predicted.
- Most PRNGs have bad cryptographic properties! How to break?

Pseudorandom Number Generators (PRNG)

Example 1 (Linear congruential generator)

$$s_0 = seed,$$

$$s_{i+1} = as_i + b \mod m, i = 0, 1, \ldots$$

where a, b, m are integer constants.

Pseudorandom Number Generators (PRNG)

Example 1 (Linear congruential generator)

$$s_0 = seed,$$

$$s_{i+1} = as_i + b \mod m, i = 0, 1, \ldots$$

where a, b, m are integer constants.

Example 2 (rand() function used in ANSI C)

$$s_0 = 12345,$$

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

where a, b, m are integer constants.

Cryptographically Secure Pseudorandom Number Generators (CSPRNG)

CSPRNG

Cryptographically secure pseudorandom number generators (CSPRNGs) are a special type of PRNG which is unpredictable.

Cryptographically Secure Pseudorandom Number Generators (CSPRNG)

CSPRNG

Cryptographically secure pseudorandom number generators (CSPRNGs) are a special type of PRNG which is unpredictable.

- Given n consecutive bits of output $s_i, s_{i+1}, \ldots, s_{i+n-1}$, the following output bits s_{i+n} cannot be predicted (in polynomial time) with pro. better than 50%.

Cryptographically Secure Pseudorandom Number Generators (CSPRNG)

CSPRNG

Cryptographically secure pseudorandom number generators (CSPRNGs) are a special type of PRNG which is unpredictable.

- Given n consecutive bits of output $s_i, s_{i+1}, \ldots, s_{i+n-1}$, the following output bits s_{i+n} cannot be predicted (in polynomial time) with pro. better than 50%.
- The need for unpredictability of CSPRNGs is unique to cryptography.

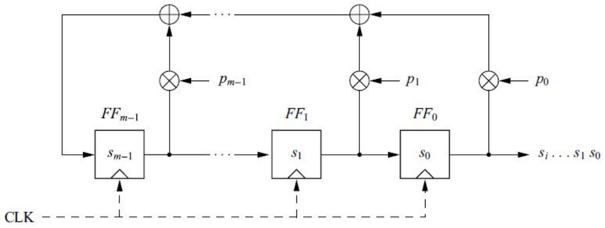
1 One-Time Pad (OTP)

2 Random number generators (RNGs)

3 Linear feedback shift registers (LFSRs)

4 Trivium: a modern stream cipher

Linear feedback shift registers (LFSRs)



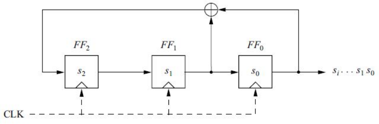
- An LFSR: storage elements (flip-flops) and a feedback path.
- Degree of the LFSR: #storage elements.
- The feedback computes fresh FF as XOR-sum of certain FFs.
- If $p_i = 1$ (closed switch), the feedback is active. Otherwise, there is not feedback from this flip-flop (open switch).

$$s_{m+i} = s_{m+i-1}p_{m-1} + \ldots + s_{i+1}p_1 + s_i p_0 \mod 2,$$

Linear feedback shift registers (LFSRs)

Example 3 (LFSR)

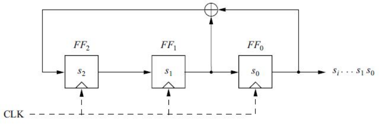
Initial state is $s_2 = 1, s_1 = 0, s_0 = 0$, what is the complete output sequence?



Linear feedback shift registers (LFSRs)

Example 3 (LFSR)

Initial state is $s_2 = 1, s_1 = 0, s_0 = 0$, what is the complete output sequence?



clk	FF_2	FF_1	$FF_0 = s_i$
0	1	0	0
1	0	1	0
2	1	0	1
3	1	1	0
4	1	1	1
5	0	1	1
6	0	0	1
7	1	0	0

$$s_{i+3} = s_{i+1} + s_i \mod 2$$

0010111 0010111 0010111...

Linear feedback shift registers (LFSRs)

How about the period of output sequence?

Linear feedback shift registers (LFSRs)

How about the period of output sequence?

Maximum sequence length

The maximum sequence length generated by an LFSR of degree m is $2^m - 1$.

Linear feedback shift registers (LFSRs)

How about the period of output sequence?

Maximum sequence length

The maximum sequence length generated by an LFSR of degree m is $2^m - 1$.

Example 4

Given an LFSR of degree $m = 4$ and the feedback path $(p_3 = 0, p_2 = 0, p_1 = 1, p_0 = 1)$, the output sequence of the LFSR has a period of $2^m - 1 = 15$, i.e., it is a maximum-length LFSR.

Attack on LFSR: Exhaustive Key Search

A stream cipher using LFSR with degree n as the keystream generator. Assume initial key K is n bits.

Attack on LFSR: Exhaustive Key Search

A stream cipher using LFSR with degree n as the keystream generator. Assume initial key K is n bits.

- Setting: ciphertext-only attack.

Attack on LFSR: Exhaustive Key Search

A stream cipher using LFSR with degree n as the keystream generator. Assume initial key K is n bits.

- Setting: ciphertext-only attack.
- Exhaustive key search.
  - Guess initial key K
  - Generate the corresponding keystream S'
  - Compute $P' = C + S'$ and check if P' is meaningful.
  - If so, ready. Otherwise, keep on guessing.

Attack on LFSR: Exhaustive Key Search

A stream cipher using LFSR with degree n as the keystream generator. Assume initial key K is n bits.

- Setting: ciphertext-only attack.
- Exhaustive key search.
  - Guess initial key K
  - Generate the corresponding keystream S'
  - Compute $P' = C + S'$ and check if P' is meaningful.
  - If so, ready. Otherwise, keep on guessing.
- For k-bit key, probability to find key after N guesses: $N2^{-k}$

Attack on LFSR: Exhaustive Key Search

A stream cipher using LFSR with degree n as the keystream generator. Assume initial key K is n bits.

- Setting: ciphertext-only attack.
- Exhaustive key search.
  - Guess initial key K
  - Generate the corresponding keystream S'
  - Compute $P' = C + S'$ and check if P' is meaningful.
  - If so, ready. Otherwise, keep on guessing.
- For k-bit key, probability to find key after N guesses: $N2^{-k}$

Upper bound to the security strength s of a cipher

Security strength s of a cipher with a k-bit key is at most k.

Attack on LFSR: state reconstruction using linear algebra

Linearity

A function f is linear (over $\mathbb{Z} = 2\mathbb{Z}$) if $f(x+y) = f(x) + f(y)$ If f_1 and f_2 are linear, $f_2 \circ f_1$ is linear.

Attack on LFSR: state reconstruction using linear algebra

Linearity

A function f is linear (over $\mathbb{Z} = 2\mathbb{Z}$) if $f(x + y) = f(x) + f(y)$ If f_1 and f_2 are linear, $f_2 \circ f_1$ is linear.

- Setting: known plaintext attack. Adversary can obtain n subsequent bits of keystream.

Attack on LFSR: state reconstruction using linear algebra

Linearity

A function f is linear (over $\mathbb{Z} = 2\mathbb{Z}$) if $f(x+y) = f(x) + f(y)$ If f_1 and f_2 are linear, $f_2 \circ f_1$ is linear.

- Setting: known plaintext attack. Adversary can obtain n subsequent bits of keystream.
- Actually, n keystream bits allow reconstructing the full state!

Attack on LFSR: state reconstruction using linear algebra

Linearity

A function f is linear (over $\mathbb{Z} = 2\mathbb{Z}$) if $f(x+y) = f(x) + f(y)$ If f_1 and f_2 are linear, $f_2 \circ f_1$ is linear.

- Setting: known plaintext attack. Adversary can obtain n subsequent bits of keystream.
- Actually, n keystream bits allow reconstructing the full state!
  - Assume we know the state S^t of clock t
  - $S^t \leftarrow M \cdot S^{t-1}$, $S^{t-1} \leftarrow M \cdot S^{t-2}$, ...
  - Hence, $S^t = M^t S^0$, while $S^0 = K$
  - Solving: Gaussian elimination with negligible effort: $O(n^3)$

Attack on LFSR: state reconstruction using linear algebra

Linearity

A function f is linear (over $\mathbb{Z} = 2\mathbb{Z}$) if $f(x+y) = f(x) + f(y)$ If f_1 and f_2 are linear, $f_2 \circ f_1$ is linear.

- Setting: known plaintext attack. Adversary can obtain n subsequent bits of keystream.
- Actually, n keystream bits allow reconstructing the full state!
  - Assume we know the state S^t of clock t
  - $S^t \leftarrow M \cdot S^{t-1}$, $S^{t-1} \leftarrow M \cdot S^{t-2}$, ...
  - Hence, $S^t = M^t S^0$, while $S^0 = K$
  - Solving: Gaussian elimination with negligible effort: $O(n^3)$

Need for non-linearity

Purely linear ciphers offer no security.

Specifications of Trivium

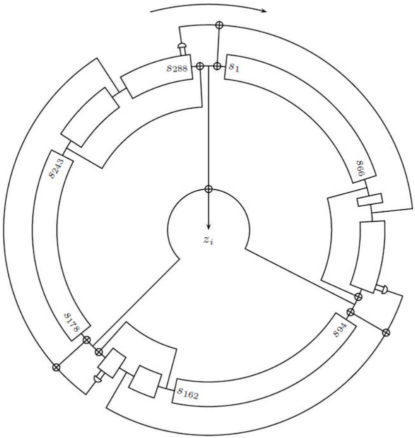
Trivium is one finial cipher of eSTREAM Stream Cipher Project.

Parameters	
Key size	80 bits
IV size	80 bits
Internal state size	288 bits
Keystream size	2^{64}

Desgin Document:
https://link.springer.com/chapter/10.1007/11836810_13

Trivium: Key and IV Setup

$(s_1, s_2, \ldots, s_{93}) \leftarrow (K_{80}, \ldots, K_1, 0, \ldots, 0)$
$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (IV_{80}, \ldots, IV_1, 0, \ldots, 0)$
$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$
for $i = 1$ to $4 \cdot 288$ **do**
  $t_1 \leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}$
  $t_2 \leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}$
  $t_3 \leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}$
  $(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$
  $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$
  $(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$
end for

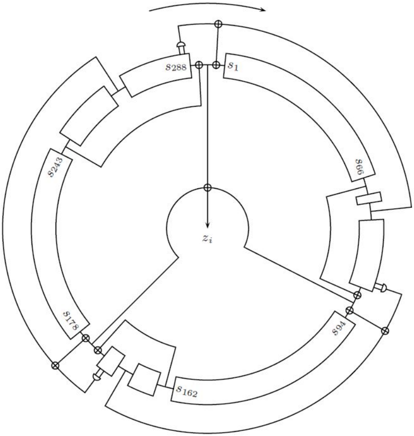


Trivium: Key Stream Generation

for $i = 1$ **to** N **do**

$$t_1 \leftarrow s_{66} + s_{93}$$
$$t_2 \leftarrow s_{162} + s_{177}$$
$$t_3 \leftarrow s_{243} + s_{288}$$
$$z_i \leftarrow t_1 + t_2 + t_3$$
$$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$$
$$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$$
$$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$$
$$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$$
$$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$$
$$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$$

end for



Thanks & Questions