Stream Ciphers

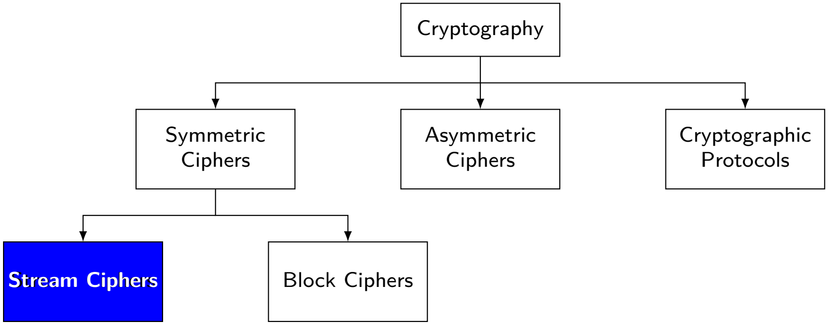
Cryptography - CS 411 / CS 507

Erkay Savaş

Department of Computer Science and Engineering
Sabancı University

October 4, 2019

Taxonomy of Cryptographic Algorithms



Stream ciphers

- Basic idea comes from One-Time-Pad cipher,
- Encryption:

$$c_i = m_i \oplus k_i \qquad i = 1, 2, 3, \ldots$$

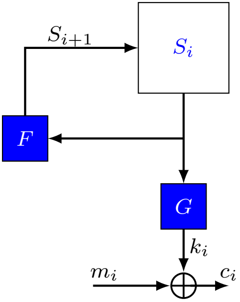
- Decryption:

$$m_i = c_i \oplus k_i \qquad i = 1, 2, 3, \ldots$$

- Drawback:
  - Key-stream should be as long as plain-text.
  - Key distribution & management difficult.
- Solution: Stream Ciphers
  - Key-stream is generated using a pseudo-random generator from a relatively short secret key

Stream ciphers

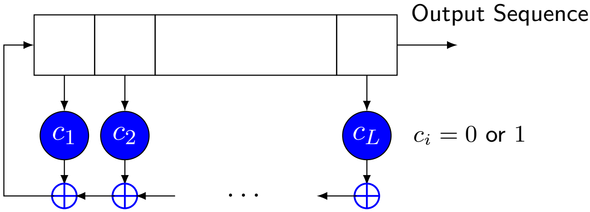
- Randomness: Closely related to unpredictability.
- Pseudo-randomness: Pseudo-random sequences appears random to a computationally bounded adversary.
- Stream ciphers can be modeled as finite-state machines.



Si : state at time $t = i$.
G : output function.
F : next-state function.

Initial state, output and next-state functions are controlled by the secret key.

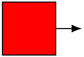
Linear Feedback Shift Registers (LFSR)



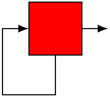
$$C(x) = 1 + c_1x + c_2x^2 + \cdots + c_Lx^L$$

- If $C(x)$ is chosen carefully the output of LFSR can have maximum period of $2^L - 1$

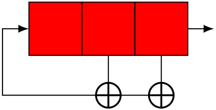
LFSR - Connection Polynomial



$$1$$



$$1 + x$$

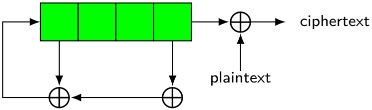


$$1 + x^2 + x^3$$

http://fchabaud.free.fr/English/

- m-sequences have good statistical properties.
- However, they are predictable

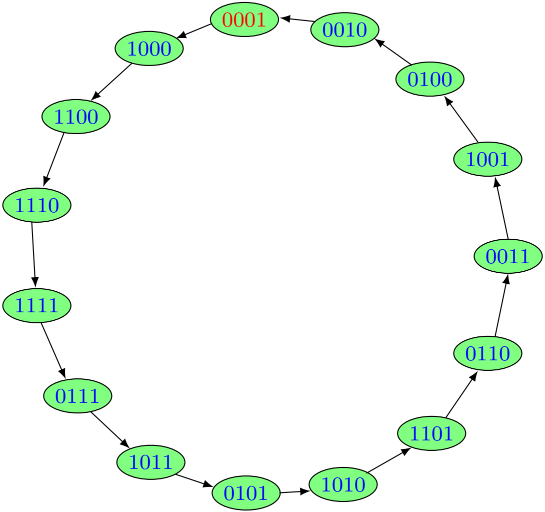
Example: $1 + x + x^4$



Output of LFSR when initial state is (0001):

$100011110101100 \quad 100011110101100\ldots$

LFSR ciphers



Linear Complexity of a Sequence

- Definition: The linear complexity of a binary sequence s^n, denoted $L(s^n)$, is the length of the shortest LFSR that generates this sequence.
- Use: It can be used as a tool to asses the randomness (or unpredictability) of a sequence.
  - **NIST - Special Publication 800-22 (Revision 1a)**: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications - Revised:April 2010
  - https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf
- Problem: Is it easy to construct the LFSR for a given sequence?

Berlekamp-Massey Algorithm(BMA)

- BMA is an efficient algorithm for determining the linear complexity of a finite binary sequence.
  - Let s be an binary sequence of linear complexity L, and
  - let t be any subsequence of s of length at least $2L$.
  - Then the BMA with input t determines an LFSR of length L which generates s.
- Expected linear complexity of a random sequence $E(L(s^n)) \approx n/2 + 2/9$.

Berlekamp-Massey Algorithm

Algorithm 1 Berlekamp-Massey Algorithm

Input: $s^n = s_0, s_1, s_2, \ldots, s_{n-1}$
Output: $L(s^n)$ and $C(x)$

$C(x) = B(x) = 1$, $L = 0$, $m = -1$ and $i = 0$
while $i < n$ **do**
  $\Delta = (s_i + c_1 s_{i-1} + c_2 s_{i-2} + \ldots + c_L s_{i-L})$
  if $\Delta = 1$ **then**
    $T(x) = C(x)$ and $C(x) = C(x) + B(x) \cdot x^{i-m}$
    if $L \leq i/2$ **then**
      $L = i + 1 - L$, $m = i$ and $B(x) = T(x)$
    end if
  end if
  $i = i + 1$
end while
return L and $C(x)$

Example

$s^{31} = 1000010101110110001111100110100$

$C(x)$	L	m	$B(x)$	i	Δ
1	0	-1	1	0	$\Delta = s_0 = 1$
$1+x$	1	0	1	1	$\Delta = s_1 + s_0 = 1$
1	1	0	1	2	$\Delta = s_2 = 0$
1	1	0	1	3	$\Delta = s_3 = 0$
1	1	0	1	4	$\Delta = s_4 = 0$
1	1	0	1	5	$\Delta = s_5 = 1$
$1+x^5$	5	5	1	6	$\Delta = s_6 + s_1 = 0$
$1+x^5$	5	5	1	7	$\Delta = s_7 + s_2 = 1$

Example

$s^{31} = 1000010101110110001111100110100$

$C(x)$	L	m	$B(x)$	i	Δ
$1+x^2+x^5$	5	5	1	8	$\Delta = s_8 + s_6 + s_3 = 0$
$1+x^2+x^5$	5	5	1	9	$\Delta = s_9 + s_7 + s_4 = 0$
$1+x^2+x^5$	5	5	1	10	$\Delta = s_{10} + s_8 + s_5 = 0$
$1+x^2+x^5$	5	5	1	11	$\Delta = s_{11} + s_9 + s_6 = 0$
$1+x^2+x^5$	5	5	1	12	$\Delta = s_{12} + s_{10} + s_7 = 0$
$1+x^2+x^5$	5	5	1	13	$\Delta = s_{13} + s_{11} + s_8 = 0$
$1+x^2+x^5$	5	5	1	14	$\Delta = s_{14} + s_{12} + s_9 = 0$
$1+x^2+x^5$	5	5	1	15	$\Delta = s_{15} + s_{13} + s_{10} = 0$

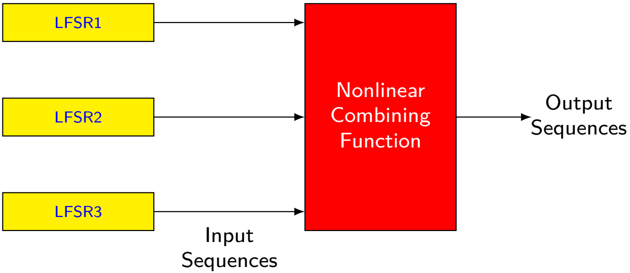
Properties of Linear Complexity

1. For any $n \geq 1$, the linear complexity of the sequence s^n satisfies $0 \leq L(s^n) \leq n$.
2. $L(s^n) = 0$ iff s^n is an all-zero sequence.
3. $L(s^n) = n$ iff $s^n = 0, 0, \ldots, 0, 1$.
4. If s is periodic with period N, then $L(s) \leq N$
5. $L(s \oplus t) = L(s) + L(t)$
6. $L(s \cdot t) = L(s) \cdot L(t)$ if $\gcd(L(s), L(t)) = 1$

Stream Ciphers Based on LFSRs

- Desirable properties of LFSR-based key-stream generators:
  - Large period
  - Good statistical properties

Nonlinear Combination Generator



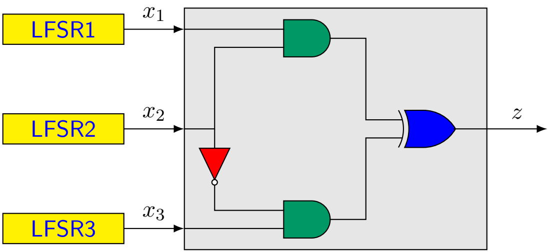
Nonlinear Combination Generator

Combiner function must be

1. balanced
2. carefully selected so that there is no statistical dependence between any small subset of n LFSR sequences and the output sequence
3. highly nonlinear (Nonlinearity of a function is given as the maximum of the order of the terms in function's algebraic normal form).

Example:
$F(x_1, x_2, x_3, x_4, x_5) = 1 \oplus x_2 \oplus x_3 \oplus x_4x_5 \oplus x_1x_3x_4x_5$
has nonlinear order 4.

The Geffe Generator



- Utilizing the algebraic normal form of the combiner function we can compute the linear complexity of the output sequence.
- $F(x_1, x_2, x_3) = x_1x_2 \oplus x_2x_3 \oplus x_3$

Properties of the Geffe Generator

- If the lengths of the LFSRs are relatively prime and all connection polynomials are primitive, then
  - $L = L_1 \cdot L_2 + L_2 \cdot L_3 + L_3$
  - $T = (2^{L_1} - 1) \cdot (2^{L_2} - 1) \cdot (2^{L_3} - 1)$
- When we inspect the truth table of the combiner function we gain more insight about the security of Geffe generator.

Correlation in the Geffe Generator

x_1	x_2	x_3	$z = F(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

- The combiner function is balanced
- However, the correlation of z to x_1 is $P(z = x_1) = \frac{3}{4}$

Correlation Attacks

- The method
  - requires a sufficiently long output sequence.
  - We assumed that the connection polynomials of the LFSRs are known. (Recall Kerckhoffs' principle)
  - We do not know the current or initial states of the LFSRs.
  - Input sequences of the same length will be compared against the output sequence.
  - the input sequence yielding a correlation that is matching to the predefined correlation will be taken.

Example: Correlation Attacks

- Geffe Generator:
  - $LFSR_1 : 1 + x + x^4$ and Initial key 1: 0001
  - $LFSR_2 : 1 + x + x^3$ and Initial key 2: 010
  - $LFSR_3 : 1 + x^2 + x^5$ and Initial key 3: 10101

x_1	1	0	0	0	1	1	1	1	0	1	0	1	1	0	0
x_2	0	1	0	0	1	1	1	0	1	0	0	1	1	1	0
x_3	1	0	1	0	1	1	1	0	1	1	0	0	0	1	1
z	1	0	1	0	1	1	1	0	0	1	0	1	1	0	1

Example: Correlated, Indeed

x_1	1	0	0	0	1	1	1	1	0	1	0	1	1	0	0	12/15
x_2	0	1	0	0	1	1	1	0	1	0	0	1	1	1	0	8/15
x_3	1	0	1	0	1	1	1	0	1	1	0	0	0	1	1	11/15
z	1	0	1	0	1	1	1	0	0	1	0	1	1	0	1	

Let us start with 0111 for $LFSR_1$
(real seed is 0001)

Example: Computing Correlation

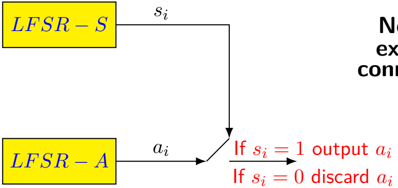
z	1	0	1	0	1	1	1	0	0	1	0	1	1	0	1	
0111	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1	8/15
1011	1	1	0	1	0	1	1	0	0	1	0	0	0	1	1	8/15
0101	1	0	1	0	1	1	0	0	1	0	0	0	1	1	1	10/15
1010	0	1	0	1	1	0	0	1	0	0	0	1	1	1	1	6/15
1101	1	0	1	1	0	0	1	0	0	0	1	1	1	1	0	8/15
0110	0	1	1	0	0	1	0	0	0	1	1	1	1	0	1	10/15
0011	1	1	0	0	1	0	0	0	1	1	1	1	0	1	0	6/15
1001	1	0	0	1	0	0	0	1	1	1	1	0	1	0	1	6/15
0100	0	0	1	0	0	0	1	1	1	1	0	1	0	1	1	8/15
0010	0	1	0	0	0	1	1	1	1	0	1	0	1	1	0	4/15
0001	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**12/15**
1000	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1	8/15

Example: Cost of the Attack

- Brute force attack: $15 \times 7 \times 31 = 3255$ trial
- Correlation attack: $15 + 7 + 31 = 53$ trial.
- If we have n LFSRs, the key space ideally is $\prod_{i=1}^{n} 2^{L_i} - 1$
- If there is correlation between the output and inputs, the *effective* key space can be reduced to $\sum_{i=1}^{n} 2^{L_i} - 1$

Other Constructions

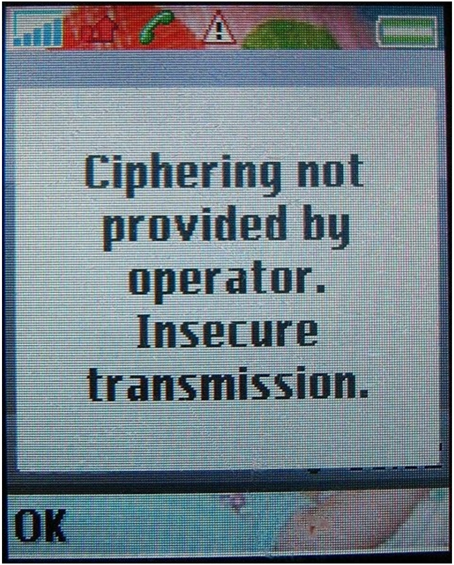
- An LFSR can be clocked by the output of another LFSR.
  - This introduces an irregularity in clocking of the first LFSR, hence increase the linear complexity of its output.
- Example: Shrinking Generator



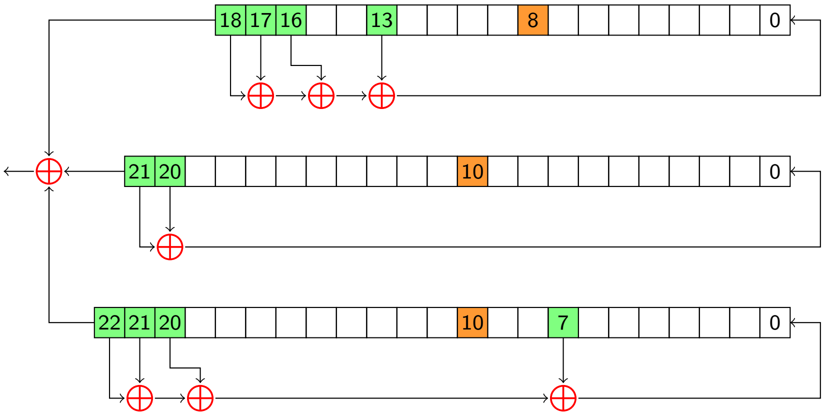
No attacks better than exhaustive search if the connection polynomials are secret

if $gcd(L_S, L_A) = 1 \rightarrow T = (2^{L_A}-1)\cdot 2^{L_S-1}$
$L_A \cdot 2^{L_S-2} < L < L_A \cdot 2^{L_S-1}$

GSM Cryptography



GSM Cryptography: A5/1



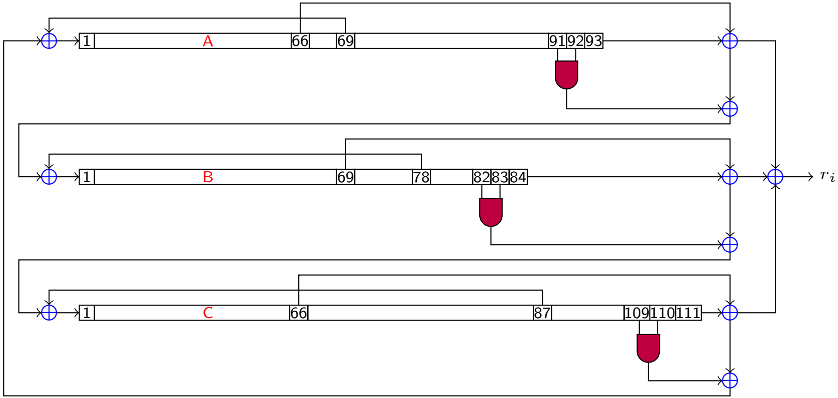
A5/1

- A5/2 is a weaker version of A5/1
  - Both A5/1 and A5/2 are weak ciphers.
- A5/3 , a.k.a. KASUMI, is a block cipher.
  - In 2006, ciphertext-only attack against A5/3
  - Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication, by E. Barkan, E. Biham and N. Keller, July 2006
  - http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/2006/CS/CS-2006-07.pdf
  - O. Dunkelman, N. Keller, A. Shamir (2010-01-10). A Practical-Time Attack on the A5/3 Cryptosystem Used in Third Generation GSM Telephony

eSTREAM Competition - Trivium

- **Trivium** is a synchronous stream cipher designed to provide a flexible trade-off between speed and gate count in hardware, and reasonably efficient software implementation.
- Three shift registers: A, B, and C (93, 84, and 111 bits, respectively)
  - $a_i = c_{i-66} + c_{i-111} + c_{i-110}c_{i-109} + a_{i-69}$
  - $b_i = a_{i-66} + a_{i-93} + a_{i-92}a_{i-91} + b_{i-78}$
  - $c_i = b_{i-69} + b_{i-84} + b_{i-83}b_{i-82} + c_{i-87}$
- The output bits $r_0 \ldots r_{2^{64}-1}$ are then generated by
  - $r_i = c_{i-66} + c_{i-111} + a_{i-66} + a_{i-93} + b_{i-69} + b_{i-84}$

Trivium



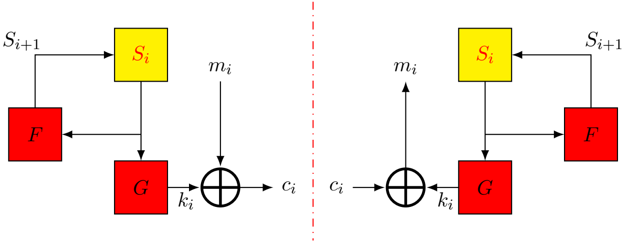
Trivium

- Given an 80-bit key $k_0 \ldots k_{79}$ and an l-bit IV $v_0 \ldots v_{l-1}$ (where $0 \leq l \leq 80$), Trivium is initialized as follows:
  - $(a_{-1245} \ldots a_{-1153}) = (0, 0 \ldots 0, k_0 \ldots k_{79})$
  - $(b_{-1236} \ldots b_{-1153}) = (0, 0 \ldots 0, v_0 \ldots v_{l-1})$
  - $(c_{-1263} \ldots c_{-1153}) = (1, 1, 1, 0, 0 \ldots 0)$
- The large negative indices on the initial values reflect the 1152 steps that must take place before output is produced.

Performance

Design	Technology	Max. Frequency	Area	Throughput	bits/cycle
Trivium by Gaj et al.	90nm	800 MHz	≈5645	51.2 Gpbs	64
AES by Satoh	0.11μm	145 MHz	12454	1.595 Gpbs	11
AES by Hodjat	0.18μm	606 MHz	473000	77.6 Gpbs	128
AES by . Northpole Eng	0.25μm	323 MHz	26000	41.3 Gpbs	127,86

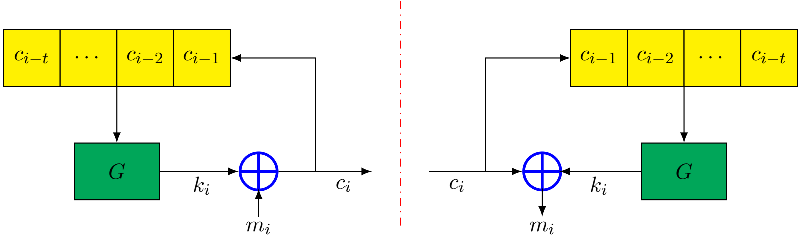
Synchronous Stream Ciphers



- Sender and receiver must be synchronized.
- Resynchronization is needed.
- Key-stream is independent of plaintext and ciphertext. (confusion)
- No error propagation.
- Active attacks can easily be detected (i.e. insertion, deletion, replay)

Asynchronous Stream Ciphers

- a.k.a. self-synchronizing stream ciphers



Asynchronous Stream Ciphers

- The key stream is generated as a function of a fixed number of previous ciphertext bits
- Limited error propagation (up to **t** bits).
- After at most **t** bits later than synchronization is lost, it resynchronizes itself
- It helps to diffuse plain-text statistics.

Salsa20 Stream Cipher Family

- A family of 256-bit stream cipher
- The internal state is made of sixteen 32-bit words
- Initial state: 8 words of key + 2 words of block number + 2 words of nonce + 4 fixed words
- Salsa20 generates the key stream in 64 B (512-bit) blocks XORed to message stream
- Each block is an independent function of the secret key, the nonce, and 64-bit block number