Chapter 3

Modern Stream Cipher

Course website: https://ece.uwaterloo.ca/~j25ni/CP460
Outline

• Idea and general concepts
• Linear Feedback Shift Register (LFSR)
• BBS Generator
• RC4
Introduction

• Vernam cipher (one-time pad)
  – perfect secrecy
  – impractical – long key that cannot be reused
• (some) stream ciphers examples:
  – RC4 – sware, WEP, SSL/TLS etc.
  – A5/1 – GSM communication (phone$base station)
    remark: UMTS uses KASUMI (block cipher in OFB+CTR mode)
  – E0 – Bluetooth (BR/EDR – basic rate/enhanced data rate)
    remark: Bluetooth Low Energy uses AES-CCM
• basic types of stream ciphers: synchronous and self-synchronizing
Basic Concept

Plaintext \( m = m_0 m_1 m_2 \cdots \)
\( m_i \in GF(2) \)

Key \( z = z_0 z_1 z_2 \cdots \)
\( z_i \in GF(2) \)

Ciphertext \( c = c_0 c_1 c_2 \cdots \)
\( c_i \in GF(2) \)

\( c_i = m_i \oplus z_i \)

\( m_i = c_i \oplus z_i \)
Synchronous stream ciphers

- the most common stream ciphers used in practice
- encryption and decryption are the same
- keystream does not depend on plaintext
- usually binary additive stream ciphers (XOR of plaintext and keystream)
Synchronous stream ciphers (2)

- periodic
- require synchronization
  - decryption breaks after losing some bits of ciphertext
- vulnerable to active attacks
  - e.g. changing bits in ciphertext results in change of corresponding plaintext bits
- errors are not propagated
- IV and key must not repeat (otherwise . . . two-time pad)
  - be careful of possible keystreams overlaps
Self-synchronizing stream ciphers

- keystream depends on ciphertext (and therefore on plaintext)
- ability to self-synchronize after the loss of some ciphertexts
- aperiodic
- hard to analyze, hard to guarantee security properties
Outline

• Idea and General Concepts
• Linear Feedback Shift Register (LFSR)
• BBS Generator
• RC4
Feedback Shift Register (FSR)

- common primitive for stream cipher construction
- easy to implement in hardware
- the output sequence has good (basic) statistical properties
- easy to analyze
- \( f \) is a Boolean function in \( n \) variables
\( n \)-stage FSR Sequences

Initial state  \[(a_0, a_1, \ldots, a_{n-1})\]
State transition  \[(a_0, a_1, \ldots, a_{n-1}) \rightarrow (a_1, a_2, \ldots, a_n)\]
Feedback bit  \[a_n = f(a_0, a_1, \ldots, a_{n-1})\]
Outputs  \[a_0, a_1, \ldots, a_n, \ldots\]
The recursive relation  \[a_{k+n} = f(a_k, a_{k+1}, \ldots, a_{k+n-1}), k = 0, 1, \ldots\]
The \( k \)th state in the FSR (after clocked \( k \) times)  \[(a_k, a_{k+1}, \ldots, a_{k+n-1})\]
FSR: example

Initial state: \((a_0, a_1, a_2) = (1, 0, 1)\)

Output sequence: 101110111011..., period: 4
A 4-stage FSR

Nonlinear feedback function $f(x_0, x_1, x_2, x_3) = x_0 \oplus x_1 \oplus x_1 x_2 x_3 \oplus 1$

The initial state $(a_3, a_2, a_1, a_0) = 1011$

The output sequence $a_0, a_1, \cdots = 1101100101000011 \cdots$

Period 16
LFSR Sequences

LFSR: when the feedback function is a linear function

\[ f(a_0, a_1, a_2, \ldots, a_{n-1}) = c_0a_0 \oplus c_1a_1 \oplus \cdots \oplus c_{n-1}a_{n-1}, \quad c_i \in \{0, 1\} \]

\[ \text{n-stage LFSR in GF}(2) \]
LFSR Sequences

\[ f(a_0, a_1, a_2, \cdots, a_{n-1}) = c_0 a_0 \oplus c_1 a_1 \oplus \cdots \oplus c_{n-1} a_{n-1}, \quad c_i \in \{0, 1\} \]

\[
\begin{align*}
a_n &= c_{n-1} a_{n-1} \oplus c_{n-2} a_{n-2} \oplus \cdots \oplus c_0 a_0 \\
a_{n+1} &= c_{n-1} a_n \oplus c_{n-2} a_{n-1} \oplus \cdots \oplus c_0 a_1 \\
\vdots & \quad \vdots \\
a_{n+t} &= c_{n-1} a_{n+t-1} \oplus c_{n-2} a_{n+t-2} \oplus \cdots \oplus c_0 a_t, \quad t = 0, 1, 2, \cdots
\end{align*}
\]
LFSR: example

Initial state: \((a_0, a_1, a_2, a_3, a_4) = (1, 0, 0, 1, 1)\)

Output sequence: 
1001101001000010101110110001111

Output sequence: 
1001101001000010101110110001111100110..., period: 31
LFSR Sequences

\[ f(a_0, a_1, a_2, \cdots, a_{n-1}) = c_0 a_0 \oplus c_1 a_1 \oplus \cdots \oplus c_{n-1} a_{n-1}, \ c_i \in \{0, 1\} \]

Suppose at least one of \( c_0, c_1, \cdots, c_{n-1} \neq 0 \), otherwise, \( f(a_0, a_1, \cdots, a_{n-1}) = 0 \).

We always assume \( c_0 = 1 \).

LFSR sequence properties: Totally based on the feedback function;

\#. states in n-stage LFSR: \( \leq 2^n - 1 \).

\#. periods in n-stage LFSR: \( \leq 2^n - 1 \).

\#. states = \#. periods in n-stage LFSR: \( \leq 2^n - 1 \).

**m-sequences**: If an LFSR sequence of \( n \) stages has the maximal period \( 2^n - 1 \), then it is called a maximal length sequence, shortened as m-sequence, or pseudo noise (PN) sequence in communications.
m-sequences of periods 7, 15, 31 and 63

\(m\)-sequence of period 7: 1001011.

\(m\)-sequence of period 15: 0001 0011 0101 111.

\(m\)-sequence of period 31: 100001010111011001111100110100

\(m\)-sequence of period 63:

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

Question: How can you verify that those are \(m\)-sequences?
Randomness Measurements

- **Run**: For a binary sequence $a$ with period $N$, $k$ consecutive zeroes (or ones) preceded by one (or zero) and followed by one (or zero) is called a run of zeroes (or ones) of length $k$.

- **Autocorrelation and crosscorrelation**: Let $a$ and $b$ be two binary sequences with period $N$, for $0 \leq \tau \leq N - 1$

$$C_{a,b}(\tau) = \begin{cases} 
\sum_{i=0}^{N-1} (-1)^{a_i+a_{i+\tau}} & \text{autocorrelation function of } a \\
\sum_{i=0}^{N-1} (-1)^{a_i+b_{i+\tau}} & \text{if } a = b \\
\sum_{i=0}^{N-1} (-1)^{a_i+b_{i+\tau}} & \text{crosscorrelation function of } a \\
\sum_{i=0}^{N-1} (-1)^{a_i+b_{i+\tau}} & \text{if } b \text{ is not a cyclic shift of } b
\end{cases}$$
Golomb’s Three Randomness Postulates for Binary Sequences

R-1. In every period, the number of zeroes is nearly equal to the number of ones, i.e., the disparity is not to exceed 1.

R-2. In every period, half the runs have length one, one-fourth have length two, one-eighth have length three, etc., as long as the number of runs so indicted exceeds 1. Moreover, for each of these lengths, there are equally many runs of 0’s and of 1’s.

R-3. The autocorrelation function $C(\tau)$ is two-valued, given by

$$C(\tau) = \begin{cases} N & \text{if } \tau \equiv 0 \pmod{N} \\ K & \text{if } \tau \not\equiv 0 \pmod{N} \end{cases}$$

where $K$ is a constant. If $K = -1$ for $N$ odd and $K = 0$ for $N$ even, then we say that the sequence has the (ideal) 2-level autocorrelation function.
Golomb’s Three Randomness Postulates for Binary Sequences

R-3. The autocorrelation function $C(\tau)$ is two-valued, given by

$$C(\tau) = \begin{cases} N & \text{if } \tau \equiv 0 \pmod{N} \\ K & \text{if } \tau \not\equiv 0 \pmod{N} \end{cases}$$

where $K$ is a constant. If $K = -1$ for $N$ odd and $K = 0$ for $N$ even, then we say that the sequence has the (ideal) 2-level autocorrelation function.

- The probability that 0 and 1 occur is nearly equal.
- The probability that 0 or 1 occurs at a specific position nearly equal.
- No information can be obtained from the comparison between the sequence and the shifted sequence.
Randomness of m-Sequences under Golomb’s condition

Any m-sequence satisfies Golomb’s three randomness postulates.

For the cryptosystem, the pseudo-random sequence should also satisfy the following three conditions:

• The sequence should be sufficiently large;
• It is computationally easy to generate the sequence;
• It is impossible to determine the sequence from the ciphertext and partial information of the corresponding plaintext.
Linear Span

**Linear span:** the linear span of a sequence \( a \) is the shortest length of LFSR that generates \( a \).

**Berlekamp-Massey algorithm (BMA):** an algorithm for computing an LFSR with the shortest length, which generates any given binary sequence.

**Threats to the security of m-sequences:**

- If the linear span of \( a \) is \( n \) and \( 2^n < N \), then the LFSR with the shortest length which generates \( a \) can be computed from \( 2n \) consecutive elements of \( a \).
- If \( a \) is used as a key stream in a cryptosystem, then computing an LFSR with the shortest length which generates \( a \) is known as **linear span attack**.

**Question:** You are given a random bit stream, is it possible to regenerate this sequence by some LFSRs?
Nonlinear Generators

Filtering Sequence Generator

Clock Controlled and Shrinking Generators

Combinatorial Sequence Generators:

Goal: The generator should be designed in a way that increases linear spans of the output sequences while keeping the randomness of m-sequences as much as possible.
Outline

- Idea and general concepts
- Linear Feedback Shift Register (LFSR)
- BBS Generator
- RC4
Computational hard problems based PRSGs: Blum-Blum-Shub (BBS) Generators

\[ x^2 \mod N \text{ Generator} \]

- Let \( N = pq \) where \( p \) and \( q \) are distinct primes \( = 3 \pmod{4} \)
- Inputs are \((N, x_0)\) where \( x_0 \) is referred to as a seed of the generator.
- The outputs of the generator is a pseudo-random sequence \( s = \{s_i\} \) whose elements are given by

  \[
  \begin{align*}
  \text{Computing the integer:} & \quad x_{i+1} = x_i^2 \pmod{N}, i = 0, 1, \ldots \\
  \text{Extracting the bit:} & \quad s_i = \text{parity}(x_i)
  \end{align*}
  \]

where the parity function is defined by

\[
\text{parity}(x) = \begin{cases} 
1 & \text{if } x \text{ is odd} \\
0 & \text{if } x \text{ is even}
\end{cases}
\]
BBS as a Filtering Generator

The BBS generator can be considered as a filtering generator where the LFSR is replaced by an non-LFSR of one stage with the feedback function \( x^2 \mod N \), and the filtering function is the parity check function which maps log \( N \) bits to one bit.

• BBS is secure if the problem of factorization integer is infeasible.

**Question:** What is the cost to generate one bit if \( N \) is a 1024-bit number?
Let $N = 7 \times 19 = 133$ and $x_0 = 4$.
The BBS sequence $x_0, x_1 = x_0^2 \pmod{133}, \ldots$:

$$\{x_i\} = 4, 16, 123, 100, 25, 93, \ldots$$
$$\{s_i\} = 0, 0, 1, 0, 1, 1, \ldots$$

which has period 6.
Outline

• Idea and general concepts
• Linear Feedback Shift Register (LFSR)
• BBS Generator
• RC4
RC4

• (Ron’s Code 4) A variable length stream cipher introduced in 1987.
• Algorithm was kept secret for 7 years until source code was posted on the Internet. During 1987-1994, RC4 is a trade secret, but it was broken within days by Bob Jenkins after the code was posted.
RC4

• Key sizes are from 1 to 256 bytes, typically between 5 and 16, corresponding to a key length of 40 – 128 bits.
• To generate the keystream, the cipher makes use of a secret internal state which consists of two parts:
  — A permutation of all 256 possible bytes (denoted "S" below).
  — Two 8-bit index-pointers (denoted "i" and "j").
KSA and PGA of RC4

- RC4 Key-Scheduling Algorithm (KSA) and the Pseudorandom Generation Algorithm (PRGA)

**KSA (K)**

**Initialization**

For $i = 0$ to $2^n - 1$

$S[i] = i$

**Scrambling**

$j = 0$

For $i = 0$ to $2^n - 1$

$j = j + S[i] + K[i(\text{mod} l)]$

swap$(S[i], S[j])$

**PRGA (S)**

**Initialization**

$i = 0$, $j = 0$

**Generation Loop**

$i = i + 1$

$j = j + S[i]$

swap$(S[i], S[j])$

$t = S[i] + S[j](\text{mod} N)$

Output $z = S[t]$
RC4 has very large internal states: $2^{256 \times 8}$. However, there is no any guaranteed randomness properties, and suffered various distinguish attacks.
Thank You