On Stream Ciphers

Professor Agnes Chan

Stream vs. Block Ciphers

<table>
<thead>
<tr>
<th></th>
<th>Stream Ciphers</th>
<th>Block Ciphers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption</td>
<td>Individual characters (usually bits)</td>
<td>Groups of characters (in blocks)</td>
</tr>
<tr>
<td>Speed</td>
<td>Faster</td>
<td>Slower</td>
</tr>
<tr>
<td>Hardware Complexity</td>
<td>Simpler</td>
<td>More complex</td>
</tr>
<tr>
<td>Software Implementation</td>
<td>Not amenable</td>
<td>More efficient</td>
</tr>
<tr>
<td>Data Buffering</td>
<td>None of limited required</td>
<td>More space required</td>
</tr>
<tr>
<td>Error propagation</td>
<td>Limited – good for noisy channels</td>
<td>Propagates – good for assuring message integrity</td>
</tr>
</tbody>
</table>

Synchronous vs. Asynchronous

- Key is independent of plaintext and of ciphertext
- Easy to generate
- No error propagation
- Insertion, deletion can be detected
- Synchronization required
- Data authentication and integrity required
- Sequence needs to be “strong”

- Stream cipher is dependent on r previous ciphertext digits
- Self-synchronized and limited error propagation
- More difficult than synchronous ciphers with respect to detection of insertion and deletion
- Plaintext statistics are dispersed through ciphertext
- More resistant to eavesdropping
- Harder to generate
Feedback Shift Registers

• If F consists of only XOR operations, then F is a linear function, and it is called a Linear Feedback Shift Register
• In general, the feedback function can be written as a linear recurrence relation

\[ x_n = \sum_{i \in I} r_i x_{n-i} \]

where I represents a subset of \( \{0, 1, \ldots, n-1\} \)

Linear Feedback Shift Register

• Let \( F(s_0, s_1, \ldots, s_{n-1}) = s_n = \sum_{i} a_i s_{n-i} \)
• Using the shift operator E, we can express the equation as

\[ x_n = \sum_{i} a_i E^i x_i \]

\( (E^r - \sum_{i} a_i E^i) x_i = 0 \)

• The feedback polynomial \( s^r = \sum_{i} a_i s^i \)
• A LFSR sequence has maximum period \( 2^n - 1 \) (known as m-sequence) if and only if the feedback polynomial is primitive

Generation of LFSR Sequences

• Let \( f(x) = x^4 + x^3 + x^2 + x + 1 \) over GF(2)
  – initial loading is 0001: 00011
  – initial loading is 0101: 01010
  – initial loading is 0110: 01100
• Let \( f(x) = x^5 + x + 1 \) over GF(2)
  – initial loading is 0001: 0201001101111
  – note every quadruple appears exactly once except 0000
  – maximal period \( 2^5 - 1 = 15 \)
  – proving that \( f(x) \) is primitive
Finite Field

- GF(2) = Z_2
- GF(2^n) = \{ (a_{n-1}, \ldots, a_1, a_0) \mid a_i \in GF(2) \}
  - Addition can be carried out bit by bit
  - Multiplication
  - Generation of GF(2^n): done by polynomial modulo a
    primitive polynomial of degree n, m(x)
  - Elements of GF(2) can be represented as a polynomial
    \(a_{n-1}, \ldots, a_1, a_0 = a(x) = a_{n-1}x^{n-1} + \ldots + a_1x + a_0 \mod m(x)\)

Primitive Polynomial

- A polynomial f(x) over a field Q is said to be irreducible if
  f(x) cannot be factored over Q
- A polynomial f(x) over a field Q is said to be primitive if
  every root of f(x) generates the field Q
- Example.
  - f(x) = \(x^4 + x^3 + x^2 + x + 1\) over GF(2)
    f(x) is irreducible but not primitive
  - g(x) = \(x^3 + x + 1\)
    g(x) is primitive

Test Irreducibility

- To show \(x^8 + x^7 + x^5 + x + 1\) is irreducible:
  - If the number of terms is odd over GF(2), then it cannot
    be divisible by x+1
  - Try dividing by polynomials of degree 2, \(x^2 + x + 1\)
  - Try polynomials of degree 3, \(x^3 + x + 1\) and \(x^3 + x^2 + 1\)
  - Try polynomials of degree 4, \(x^4 + x^3 + x^2 + 1\), \(x^4 + x^3 + 1\),
    \(x^4 + x^2 + 1\), \(x^4 + x + 1\)
  - Do not require any more testing beyond degree 4
Test Primitivity

• To show $x^4 + x^3 + x^2 + x + 1$ is not primitive:
  – Take $\alpha$ to be a root of the polynomial, that is,
    $\alpha^4 = \alpha^3 + \alpha^2 + \alpha + 1$
  – $\alpha^5 = \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1 + \alpha^2 + \alpha^2 + \alpha = 1$

• To show $x^8 + x^4 + x^3 + x + 1$ is primitive
  – Take $\alpha$ to be a root of the polynomial, that is,
    $\alpha^8 = \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$ (00101011) = (ab)
    $\alpha^9 = \alpha^5 + \alpha^4 + \alpha^2 + \alpha$ (00110110) = (36)
    $\alpha^{12} = \alpha^7 + \alpha^5 + \alpha^3 + \alpha + 1$ (01010111) = (ab)

Multiplication in GF($2^8$)

• $\alpha^8 + \alpha^{12} = \alpha^{8+12} = \alpha^{20}$ mod 127
• (36) * (ab) = (00110110) * (10101011) = 11110010

Desired Properties of a Stream Cipher

• Long period
• Balanced 0’s and 1’s
• Bernoulli distribution of $k$-tuples for all $k>1$
• Good autocorrelation functions
  $A(\tau) = \sum_{i=0}^{p-1} (-1)^{s_i} (-1)^{s_{i+\tau}} = \begin{cases} p & \text{if } \tau = 0 \\ 0 & \text{if } \tau \neq 0 \end{cases}$
  where $p$ is the period of the sequence
• Generation algorithm should be simple and efficient
• No simple description of the generation mechanism
• Resilient to commonly known attacks
Commonly Known Attacks

• Exhaustive Key search
  – key size has to be large
  – if the generation algorithm depends variables that are not
    known/fixed, then the key consists of the parameters governing the
    variables as well as the initial loading
  – if the parameters for the algorithm are publicly known, then the
    key consists of the initial loading only

• Berlekamp-Massey Attack
  – efficient algorithm to attack periodic sequences

• Correlation Attack
  – to find the initial loading

Berlekamp-Massey Attack

• Basic Idea: every periodic sequence can be generated by a
deterministic finite state machine, namely

\[
\begin{bmatrix}
X_0 \\
X_1 \\
\vdots \\
X_{n-1}
\end{bmatrix} = \begin{bmatrix} a_0 & a_1 & \cdots & a_{l-1} \\
1 & 0 & \cdots & 0
\end{bmatrix} \begin{bmatrix}
X_{n-1} \\
X_{n-2} \\
\vdots \\
X_0
\end{bmatrix}
\]

Find the smallest such finite state machine.

• Approach:
  – find the smallest machine that generates the sequence
    obtained thus far by solving a system of linear
    equations
  – compare output of the machine with sequence bits
    obtained next. If equal, then continue; otherwise,
    compute a new solution and increase the length if
    needed

Definitions for Berlekamp-Massey Algorithm

\(n\) = length of the sequence \(s^n\) being considered
\(N\) = the \(N\)-th iteration of the sequence \(s^n\) being considered
\(L\) = the linear complexity computed so far
\(C(D)\) is the connection polynomial defined by

\[C(D) = 1 + a_1D + a_2D^2 + \ldots + a_lD^l\]

\(B(D)\) is the most recently computed connection polynomial:
let \(m\) be the largest integer \(< N\) such that

\[L(s^m) < L(s^n),\]

and \(B(D)\) is the connection polynomial that

generates \(s^n\).

NOTE: complexity of Berlekamp-Massey Algorithm is \(O(n^2)\)
Berlekamp–Massey Algorithm

1. Initialization: $C(D) \leftarrow 1; B(D) \leftarrow 1; m \leftarrow 1; L \leftarrow 0; N \leftarrow 0$
2. While $(N < n)$ do:
   2.1 Compute the discrepancy $d$, $d = (x_N + \sum_{i=1}^{m} c_i A_{i-N}) \mod 2$
   2.2 If $d=1$ then do:
       $T(D) \leftarrow C(D), C(D) \leftarrow C(D)+B(D) \cdot D^{|m|}$
       If $L \leq N/2$ then $L \leftarrow N+1-L$, $m \leftarrow N$, $B(D) \leftarrow T(D)$
   2.3 $N \leftarrow N+1$
3. Return $(L)$

Linear Complexity

- The goal is to find a Linear Feedback Shift Register that generates the sequence by solving for $a_i, i \geq 0$ in
  $F(x_0, x_1, \ldots, x_{n}) = x_n = \sum_{i \geq 0} a_i x_i$
- Linear complexity of a sequence $s$, denoted by $L(s)$ is
  (1) if $s(0)$, then $L(s)=0$
  (2) if $s$ is an infinite sequence, then $L(s)=\infty$
  (3) otherwise, $L(s)$ is length of the smallest LFSR that generates the sequence $s$
- Linear complexity profile must follow the $L=m/2$ line

LFSR Sequences

- Desirable Properties:
  - Simple and efficient
  - Balanced 0’s and 1’s
  - Bernoulli distribution of $k$-tuples for $k>1$
  - 2-valued autocorrelation function
    $A(t) = \begin{cases} 1 & \text{if } t = 0 \\ -1 & \text{if } t \neq 0 \end{cases}$
  - Used for noise generation and simulations
- Weakness:
  - susceptible to Berlekamp-Massey Attack, needs only $O(\log \rho)$ key bits to determine the key
Nonlinear Feedback Shift Register

- The feedback function contains AND-gates and converter
  \[ f(x_0, x_1, \ldots, x_n) = \sum x_{i-1} x_i \]
- Simple and efficient: if the function f can be found
- Period can be long: but difficult to analyze
- Balanced 0’s and 1’s can be obtained
- Bernoulli distribution can be achieved
- Linear complexity can be high
- Weakness: lack of mathematical theory to identify the properties of f

Nonlinear feedforward (filter)

- Linear feedback function: to guarantee long period
- Nonlinear feedforward: to introduce complexity
- Desirable properties can be achieved if g is carefully chosen
- Linear complexity is bounded above by \( \sum_{i=1}^{n} \binom{n}{i} \) where k is the degree of the nonlinear function g

Geometric Sequences

- Let q be a power of odd prime
- Linear function f: \( \text{GF}(q^n) \rightarrow \text{GF}(q) \)
- Nonlinear function g: \( \text{GF}(q) \rightarrow \text{GF}(2) \)
- Simple and efficient
- Long period: \( q^n - 1 \)
Nonlinear Combiner

- $z = g(s_1, s_2, \ldots, s_n)$ is a nonlinear function on $n$ variables
- Simple but requires more hardware
- Long period: $\text{lcm}(p_1, p_2, \ldots, p_n)$ where $p_i$ is period of LFSR $f_i$
- Desirable properties if $g$ is carefully chosen
- If $g$ is not $n$-th order correlation immune, then it is susceptible to correlation attack

Correlation Attack

- Goal: to find the initial loading of the registers
- Approach:
  - Makes use of the fact that the output bits are correlated with some specific part of the registers.
  - Reduces the complexity of exhaustive search from $\sum_{a}^{m}$ to $\sum_{a}^{n}$, where $m_i$ denotes the possibilities of the $i$-th variable and $n$ is the number of variables of the function
- Correlation-immune functions

The Geffe Generator

- Period of Geffe sequence
  - $\text{lcm}(p_1, p_2, p_3)$ where $p_i$ is the period of LFSR $f_i$
- Linear complexity
  - $L_1 + L_2 + L_3 - L_1 L_2 L_3$ where $L_i$ is the linear complexity of LFSR $f_i$
Correlation Immune

- A boolean function $f(x_1, x_2, \ldots, x_n)$ is said to be $m$-th order correlation immune if for every subset $J$ of $m$ random variables, the function value $Z = f(x_1, x_2, \ldots, x_n)$ is independent of the subset $J$; equivalently, $I(Z; J) = 0$.
- A nonlinear function is $k$-th order correlation immune if the function does not contain any product terms of degree higher than $n-k$.
- Example: any linear function is $(n-1)$-th order correlated immune.

Alternating Stop-and-Go Generator

- LFSR 2 is clocked when output of LFSR 1 is 1 and LFSR 3 is clocked when output of LFSR 1 is 0.
- The outputs of LFSR 2 and LFSR 3 are then XORed.
- If $L_1$, $L_2$, $L_3$ are relatively prime, where $L_i$ is the length of LFSR $i$, then period $= (2^{L_1}-1)(2^{L_2}-1)(2^{L_3}-1)$.
- Let $m_i$ be the linear complexity of LFSR $i$, then linear complexity $L$: $\varepsilon(m_1 + m_2) 2^{L_1} < L < (m_1 + m_2) 2^{L_1}$.
- Susceptible to Differential Analysis Attack.

Reconfigurable Feedback Shift Register

- Motivation:
  - For Next Generation Internet, real-time ultra fast speed encryption is needed.
  - High speed gate technology is extremely expensive and usually has other constraints.
- Approach:
  - Uses a slow speed generator to control a high speed one.
  - The high speed technology is to ensure speed, but not on security.
  - The slow speed technology is to gain security.
Design of RFSR

- Assume the ratio of the two speeds is $\delta$
- At every $\delta$ interval, the feedback function $f$ is reconfigured according to the output of the controller $g$
- period of $g \leq$ period of $z \leq \text{period of } (\text{g}) \times \text{lcm}(\rho_1, \ldots, \rho_t) \times \delta$
- Bernoulli distribution of $k$-tuples: simulation results
- Long term security of $z \geq \delta(\text{security of } g)$

Software-Based Stream Ciphers

- Software Encryption Algorithm (SEAL)
  - Generates large tables for table look-up
- RC2, RC4, RC6 (proposed by Rivest)
  - RC4 is proprietary
  - RC6 is considered very efficient (AES candidate)
- Fibonacci Shrinking Generator (FISh)
- Software Stream Cipher 2 (SSC2)
  - Requires only 20 lines of C code and minimum memory
Strength of SSC2

- Simple Operations:
  - exclusive or; byte/word shifts; addition; logical operations
- Strong System Security
  - long period
  - high linear complexity
  - good statistical properties
  - resilient to correlation attacks

Stream Ciphers

<table>
<thead>
<tr>
<th>Message Size</th>
<th>SSC2</th>
<th>ARC4</th>
<th>SEAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 KB</td>
<td>32,608</td>
<td>44,584</td>
<td>42,281</td>
</tr>
<tr>
<td>5 KB</td>
<td>25,808</td>
<td>32,250</td>
<td>28,723</td>
</tr>
<tr>
<td>16 KB</td>
<td>16,608</td>
<td>21,560</td>
<td>19,800</td>
</tr>
</tbody>
</table>

Memory Requirement

- SSC2 - 84 bytes
- ARC4 - 256 bytes to store a state array
- SEAL - 7 Kilobytes