COMP 8920: Cryptography Winter 2025

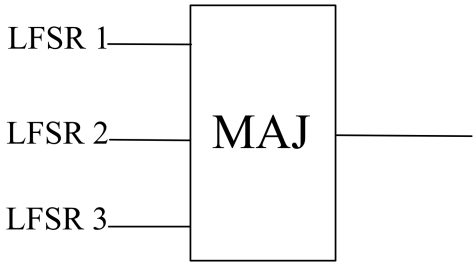
Lecture 10 — Feb 6, 2025

Prof. Curtis Bright *Scribe: Mahzabin Chowdhury*

Overview

Continuing the Attack on a Combination Generator ....

Say, we have the LFSRs whose degrees are L_1, L_2, L_3. The total key length is $L = L_1 + L_2 + L_3$.



Known Plaintext Attack on the LFSR Generator

Using the known plaintext attack on the LFSR generator with a message of length L, we can compute the first L bits of the keystream $Z_1, \ldots, Z_L$. However, this does not reveal the key, which consists of the initial values of each of the LFSRs.

We could perform an exhaustive search, over all $2^L = 2^{L_1+L_2+L_3}$ possible keys, but this is inefficient. By exploiting the fact that the bits output by this generator have a strong correlation with the bits output by each individual LFSR, a more efficient approach can be used.

The idea is to search over all 2^{L_1} (here 2^4) keys for the first LFSR and generate L bits of output for each. For the correct key, about $\frac{3}{4}$ of its bits are expected to correspond with the keystream derived from the known plaintext attack. For an incorrect key, the match probability is $\frac{1}{2}$. Once the key to the first LFSR is found, the same approach can be applied to the second and third LFSRs.

Example

Suppose the first LFSR is defined by:

$$q_i = q_{i-3} + q_{i-5}$$

in $\mathbb{Z}_2$ (degree 5).
We iterate over all possible starting values $(a_1, \ldots, a_5) \in \mathbb{Z}_2^5 \setminus \{(0,0,0,0,0)\}$.

Of which there are $2^5 - 1 = 31$ possible values. For example, suppose $(a_1, \ldots, a_5) = (0, 1, 1, 0, 1)$. If there are 24 matches $a_i = z_i$ for $i \leq 30$, and all other keys have at most 19 matches, then we likely know the initial values of LFSR 1. The other LFSRs are handled independently, resulting in:

$$(2^{L_1} - 1) + (2^{L_2} - 1) + (2^{L_3} - 1)$$

Total keys examined. Which is much better than $2^{L_1+L_2+L_3} - 1$.

Attack on a Filter Generator

We describe an algebraic attack assuming, enough bits of the keystream are known, as well as the recurrence and filtering function f. The key is the initial state of the LFSR.

Example

Suppose the recurrence is:

$$Z_{i+4} = Z_{i+1} + Z_i \quad \text{for} \quad i \geq 0,$$

and the filtering function is defined as:

$$f(a, b, c, d) = ab + cd.$$

Suppose, the first 10 bits of the keystream are $s_1, \ldots, s_{10} = 0000010100$. Thus:

$$\begin{aligned} f(z_0, z_1, z_2, z_3) &= z_0 z_1 + z_2 z_3 = s_0 = 0, \\ f(z_1, z_2, z_3, z_4) &= f(z_1, z_2, z_3, z_1 + z_0) \\ &= z_1 z_2 + z_3(z_1 + z_0) \\ &= z_1 z_2 + z_0 z_3 + z_1 z_3 \\ &= s_1 = 0. \end{aligned}$$

In general:

$$(z_{i+1}, z_{i+2}, z_{i+3}, z_{i+4}) = (z_i, z_{i+1}, z_{i+2}, z_{i+3}) \cdot \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{(denoted } A\text{)}}.$$

Thus,

$$(z_n, z_{n+1}, z_{n+2}, z_{n+3}) = (z_0, z_1, z_2, z_3) \cdot \mathbf{A}^n,$$

so the n-th keystream bit is:

$$S_n = f((z_0, z_1, z_2, z_3) \cdot \mathbf{A}^n).$$

Taking $0 \leq n \leq 9$, we derive polynomial equations in z_0, z_1, z_2, z_3:

$$\begin{aligned} z_0z_1 + z_2z_3 &= 0, \\ z_0z_3 + z_1z_2 + z_1z_3 &= 0, \\ &\vdots \\ z_0z_2 + z_1z_2 + z_1z_3 + z_2 + z_2z_3 + z_3 &= 0. \end{aligned}$$

Note that all terms have degree ≤ 2, because we can replace any z_i^2 with z_i (since $z_i^2 = z_i$ in $\mathbb{Z}_2$). This forms a polynomial system, which in general is not easy to solve. However, the system can be linearized by introducing new variables for each degree-2 term. For example:

$$\begin{aligned} X_1 &= z_0z_1, \\ X_2 &= z_0z_2, \\ &\vdots \end{aligned}$$

Resulting in:

$$\begin{aligned} X_1 + X_8 &= 0, \\ X_3 + X_5 + X_6 &= 0, \\ &\vdots \end{aligned}$$

This system can be solved for $(X_0, \ldots, X_4) = (1, 0, \ldots, 0)$. Thus:

$$\begin{aligned} X_0 = z_0 &= 1, \\ X_4 = z_1 &= 0, \\ X_7 = z_2 &= 0, \\ X_9 = z_3 &= 0. \end{aligned}$$

So the initial state of the LFSR is $(1, 0, 0, 0)$. In general, for an LFSR of degree m and if the filter f is a polynomial of degree d, the system has:

$$\sum_{i=1}^{d} \binom{m}{i}$$

distinct terms, which is $O(m^d)$. Thus, this many keystream entries are needed.

Hash Functions

Recall that encryption alone is not enough to ensure message integrity. For example, with a stream cipher, an adversary can perform a bit-flip attack and change bits of the plaintext at locations they control, even if decryption is not possible.

Bob wants a way to ensure that what Alice sent was what he received, this is known as *data integrity*. Hash functions produce a *fingerprint* of arbitrary data and can be used for this purpose. If the fingerprint is securely stored, the integrity of the data can be checked by comparing its fingerprint.

If the fingerprint does not match, the data were corrupted. If it matches, we want assurance that it was not corrupted, even though collisions may exist.

If x is a binary string and h a hash function, $h(x)$ is typically called a *digest* and is a short binary string (e.g. 160 or 256 bits). Keyed hash functions can also take a secret key k and are useful for generating *Message Authentication Codes* (MACs). If Alice and Bob share k, then a *tag* $y = h_k(x)$ can be computed by Alice, who sends (x, y) to Bob. Bob verifies that $y = h_k(x)$ and is confident in the integrity and source.

Formally, a hash family is a tuple $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ where:

- $\mathcal{X}$ is the set of possible messages,
- $\mathcal{Y}$ is the set of digests (finite),
- $\mathcal{K}$ is the keyspace.

For each $k \in \mathcal{K}$, there is a function:

$$H_k : \mathcal{X} \to \mathcal{Y}, \quad h_k \in \mathcal{H}.$$

A pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ is *valid* if $h_k(x) = y$. We want to prevent an adversary from generating a valid pair without knowing k. The only way to generate a valid pair should be by choosing x and computing $h_k(x)$.

For a hash function h to be considered secure, the following problems should be infeasible:

- **Preimage**: Given $y \in \mathcal{Y}$, find $x \in \mathcal{X}$ such that $h(x) = y$.
- **Second Preimage**: Given $x \in \mathcal{X}$, find a different $x' \in \mathcal{X}$ such that $h(x) = h(x')$ (with $x \neq x'$).
- **Collision**: Find two distinct $x, x' \in \mathcal{X}$ such that $h(x) = h(x')$. The pair (x, x') is called a collision.