Introduction to Cryptology

Lecture 15

Announcements

- No Instructor Office Hours on Thursday, Friday
- Office Hours on Monday, 4/3 from 3-4pm
- EC opportunity, due on Tuesday, May 9.
  - Up to 10 points added to Midterm grade

Agenda

- Last time
  - Collision-Resistant Hash Functions (K/L 5.1)
  - Domain Extension (Merkle-Damgard) (K/L 5.2)
  - MACs from CRHF (5.3)
    - Please read about HMAC on your own

- This time
  - Sponge Construction
  - New topic: Practical constructions
    - Stream Ciphers (K/L 6.1)
    - Block Ciphers (K/L 6.2)
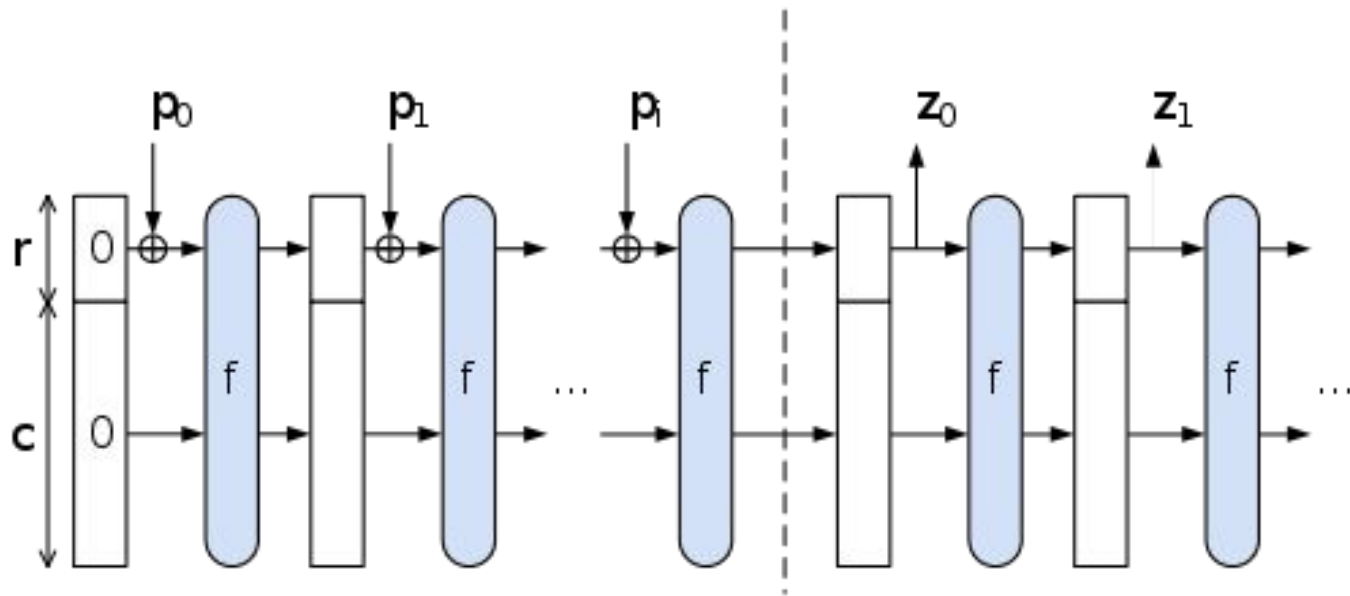
A sponge function is built from three components:
- a state memory, S, containing b bits,
- a public, truly random permutation f
- a padding function P

The state memory is divided into two sections:
- one of size r (the bitrate) and
- the other of size c (the capacity).

These sections are denoted R and C respectively.

The padding function appends enough bits to the input string so that the length of the padded input is a whole multiple of the bitrate, r. The padded input can thus be broken into r-bit blocks.

Operation

The sponge function operates as follows:

- The state S is initialized to zero
- The input string is padded. This means the input p is transformed into blocks of r bits using the padding function P.
- R is XORed with the first r-bit block of padded input
- S is replaced by $f(S)$
- R is XORed with the next r-bit block of padded input (if any)
- S is replaced by $f(S)$

…

The process is repeated until all the blocks of the padded input string are used up ("absorbed" in the sponge metaphor).

The sponge function output is now ready to be produced ("squeezed out") as follows:

- The R portion of the state memory is the first r bits of output
- If more output bits are desired, S is replaced by $f(S)$
- The R portion of the state memory is the next r bits of output

…

The process is repeated until the desired number of output bits are produced. If the output length is not a multiple of r bits, it will be truncated.

Sponge

- Is there a length-extension attack on Sponge?
- Plusses and minuses of Sponge versus Merkle Damgard.
- A Sponge-based hash known as Keccak was selected as SHA-3. Standardized by NIST in 2015.

LFSR

- Consists of an array of n registers $\vec{s} := s_{n-1}, \ldots, s_0$
- Feedback loop specified by a set of n feedback coefficients $\vec{c} := c_{n-1}, \ldots, c_0$.
- The size of the array is called the degree of the LFSR.
- Each register stores a single bit
- The state st of the LFSR at any point is the set of bits contained in the registers
- State of the LFSR is updated in a series of "clock ticks" by shifting the values to the right and setting the new value of the leftmost register.
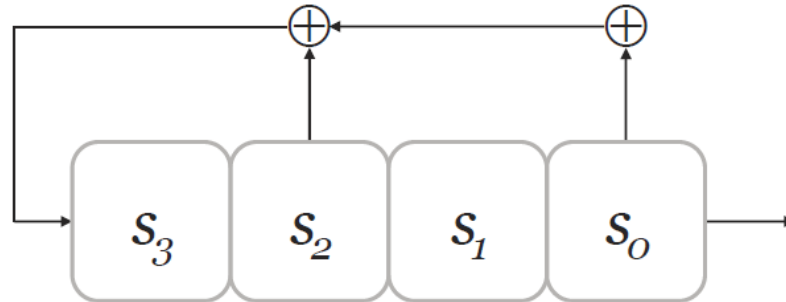
LFSR

FIGURE 6.1: A linear feedback shift register.

If state in registers at time t is:

$$\vec{s}^{(t)} := s_3^{(t)}, s_2^{(t)}, s_1^{(t)}, s_0^{(t)}$$

Then state in registers at time $t + 1$ is:

$$s_3^{(t+1)} = \langle \vec{c}, \vec{s}^{(t)} \rangle$$
$$s_2^{(t+1)} := s_3^{(t)}$$
$$s_1^{(t+1)} := s_2^{(t)}$$
$$s_0^{(t+1)} := s_1^{(t)}$$
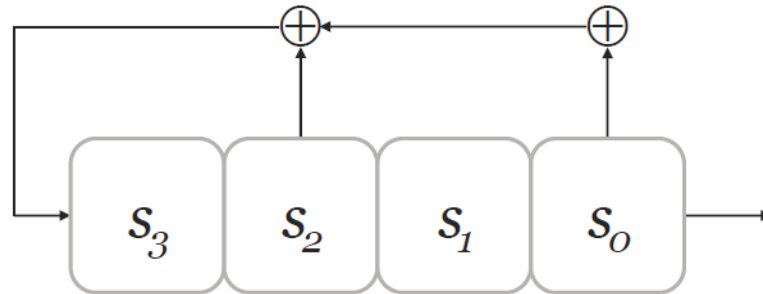
Example

FIGURE 6.1:   A linear feedback shift register.

Initial state: 0 0 1 1
1 0 0 1
1 1 0 0
1 1 1 0
1 1 1 1
0 1 1 1
0 0 1 1

- LFSR can cycle through at most 2^n states before repeating
- A maximum-length LFSR cycles through all 2^{n-1} non-zero states before repeating
- Depends only on feedback coefficients, not on initial state
- Maximum-length LFSR's can be constructed efficiently

Reconstruction Attacks

- LFSR are always insecure. We have the following generic attack:

- If state has n bits, then

  - First n output bits $y_0, \ldots, y_{n-1}$ reveal initial state $s_0, \ldots, s_{n-1}$

  - Can use next n output bits $y_n, \ldots, y_{2n-1}$ to determine $c_0, \ldots, c_{n-1}$ by setting up a system of n linear equations in n unknowns:

$$
\begin{bmatrix} y_0 & y_1 & y_2 & y_3 \\ y_1 & y_2 & y_3 & y_4 \\ y_2 & y_3 & y_4 & y_5 \\ y_3 & y_4 & y_5 & y_6 \end{bmatrix} \times \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix}
$$

Adding Non-Linearity

- Non-linear feedback
  - New value in leftmost register is a non-linear function of the current registers

- Non-linear combination generators
  - Output is non-linear function of current registers

Hardware vs. Software

- LFSR are very efficient when implemented in hardware but have poor performance in software.
- Alternate designs of stream cipher for software.
- Well-known example is RC4
  - Designed by Ron Rivest in 1987 (proprietary)
  - Code was first publicized in 1994
- Attacks on RC4
  - Various attacks are known for several years
  - Extreme care must be taken when using RC4
  - Or avoid RC4 altogether.