Cryptography

Lecture 15

Announcements

- HW5 posted on course webpage, due Wednesday 4/3
- Office Move:
  - Moving to Iribe 5238

Agenda

- Last time
  - Domain Extension for CRHF:
    - Merkle-Damgard (5.2)
    - Sponge Construction
  - Practical constructions of Stream Ciphers (K/L 6.1)
- This time
  - Practical constructions of Stream Ciphers (K/L 6.1)
    - LFSR, RC4 (Class Ex handed out, go over next time)
  - Practical constructions of Block Ciphers (K/L 6.2)

LFSR

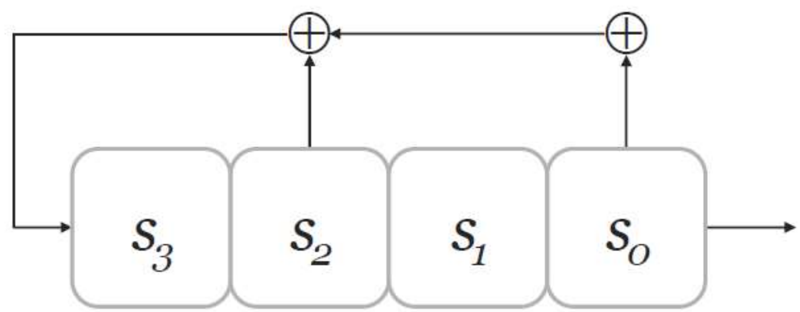

FIGURE 6.1: A linear feedback shift register.

If state in registers at time t is:

$$\vec{s}^{(t)} := s_3^{(t)}, s_2^{(t)}, s_1^{(t)}, s^{(t)}{}_0$$

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

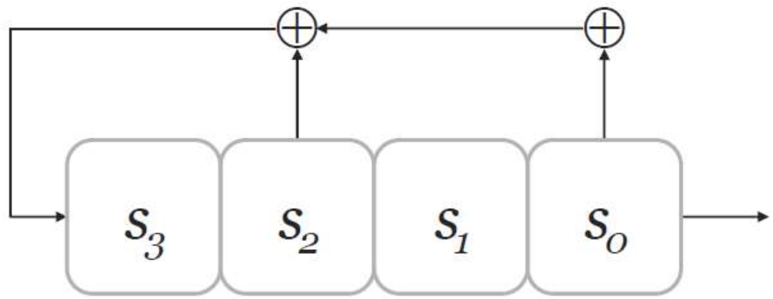

FIGURE 6.1: A linear feedback shift register.

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.

Block Ciphers

Recall: A block cipher is an efficient, keyed permutation $F: \{0,1\}^n \rightarrow \{0,1\}^\ell$. This means the function $F_k(x) := F(k, x)$ is a bijection, and moreover F_k and its inverse $F_k{}^{-1}$ are efficiently computable given k.

- n is the key length
- ℓ is the block length

Block Cipher Security

Call for proposals for AES competition: 1997 - 2000

"The security provided by an algorithm is the most important factor. . . Algorithms will be judged on the following factors. . . The extent to which the algorithm output is indistinguishable from a random permutation. . ."

get to query F_K, F_K^{-1}
f, f^{-1} } strong pseudorandom permutation

First Idea

- Random permutations over small domains are "efficient."
  - What does this mean?
- First attempt to define F_k:
  - The key k for F will specify 16 permutations $f_1, \ldots, f_{16}$ that each have an 8-bit block length.
  - Given an input $x \in \{0,1\}^{128}$, parse it as 16 bytes $x_1, \ldots, x_{16}$ and then set
  $$F_k(x) = f_1(x_1)|| \cdots ||f_{16}(x_{16})$$
- Is this a permutation?
- Is this indistinguishable from a random permutation?

Shannon's Confusion-Diffusion Paradigm

Above step is called the "confusion" step. Is combined with a "diffusion" step: the bits of the output are permuted or "mixed," using a mixing permutation.

- Confusion/Diffusion steps taken together are called a round.
- Multiple rounds required for a secure block cipher.

Example: First compute intermediate value $y = f_1(x_1)|| \cdots ||f_{16}(x_{16})$. Then permute the bits of y.

Substitution-Permutation Network (SPN)

In practice, round-functions are not random permutations, since it would be difficult to implement this in practice.

- Why?

Instead, round functions have a specific form:

- Rather than having a portion of the key k specify an arbitrary permutation f, we instead fix a public "substitution function" (i.e. permutation) S, called an S-box.
- Let k define the function f given by $f(x) = S(k \oplus x)$.

Informal Description of SPN

1. Key mixing: Set $x := x \oplus k$, where k is the current-round sub-key.
2. Substitution: Set $x := S_1(x_1)|| \cdots ||S_8(x_8)$, where x_i is the i-th byte of x.
3. Permutation: Permute the bits of x to obtain the output of the round.
4. Final mixing step: After the last round there is a final key-mixing step. The result is the output of the cipher.
   - Why is this needed?

- Different sub-keys (round keys) are used in each round.
  - Master key is used to derive round sub-keys according to a key schedule.

Formal description of SPN

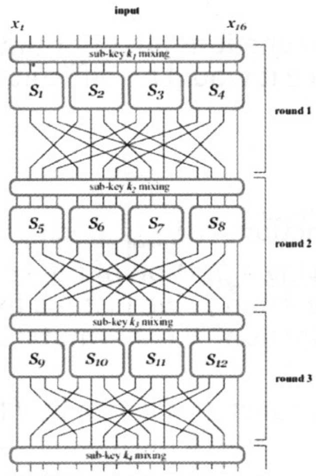

FIGURE 6.2: A substitution-permutation network.

SPN is a permutation

Proposition: Let F be a keyed function defined by an SPN in which the S-boxes are all permutations. Then regardless of the key schedule and the number of rounds, F_k is a permutation for any k.

How many rounds needed for security?

The avalanche effect.

Random permutation: When a single input bit is changed to go from x to x', each bit of $f(x)$ should be flipped with probability ½ .

- S-boxes are designed so that changing a single bit of the input to an S-box changes at least two bits in the output of the S-box.
- The mixing permutations are designed so that the output bits of any given S-box are used as input to multiple S-boxes in the next round.

The Avalanche Effect

$f(x)$ vs. $f(x')$ where x, x' differ in one bit:

1. After the first round the intermediate values differ in exactly two bit-positions. Why?
2. The mixing permutation spreads these two bit positions into two different S-boxes in the second round.
   - At the end of the second round, intermediate values differ in 4 bits.
3. Continuing the same argument, we expect 8 bits of the intermediate value to be affected after the 3rd round, 16 after the 4th round, and all 128 bits of the output to be affected at the end of the 7th round.

Practical SPN

- Usually use many more than 7 rounds.
- S-boxes are NOT random permutations.

Attacking Reduced-Round SPN

Trivial case: Attacking one round SPN with no final key-mixing step.