Introduction to Cryptography
Lecture 5 and 6

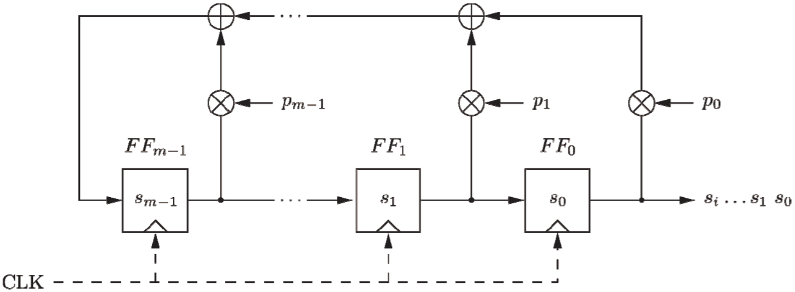
Monika K. Polak

February 10, 2021

Content of this Lecture

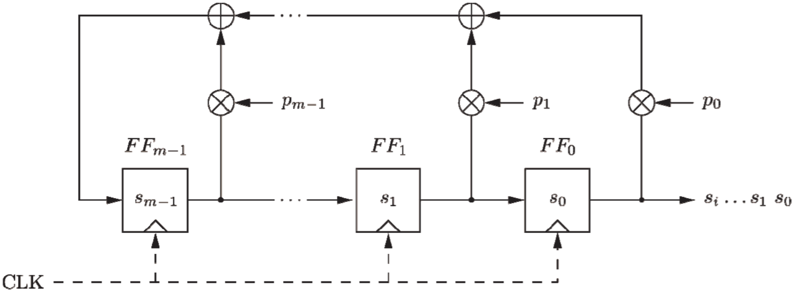
- Linear Feedback Shift Registers (LFSRs)
- Trivium
- RC4
- Intro to Block Ciphers

Linear Feedback Shift Registers (LFSRs)



- It is a cascade of flip flops, sharing the same clock, whose input bit is `a linear function` of its previous state
  - `flip-flop` – a circuit that has two stable states and can be used to store state information
- Feedback computes fresh input by XOR of certain state bits

Linear Feedback Shift Registers (LFSRs)



- Degree m given by number of storage elements
- If $p_i = 1$, the feedback connection is present ("closed switch), otherwise there is not feedback from this flip-flop ("open switch")
- Output sequence repeats periodically
- Maximum output length: $2^m - 1$

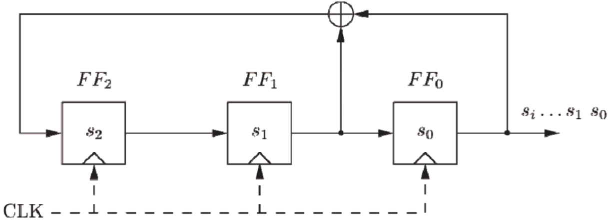
Linear Feedback Shift Registers (LFSRs)

LFSRs are typically described by polynomials:

$$P(x) = x^m + p_{m-1}x^{m-1} + \cdots + p_2x^2 + p_1x + p_0$$

- Single LFSRs generate highly predictable output
- If $2m$ output bits of an LFSR of degree m are known, the feedback coefficients p_i of the LFSR can be found by solving a system of linear equations (See Chapter 2 for details)
- Because of this many stream ciphers use combinations of LFSRs (A5/1 and Trivium)

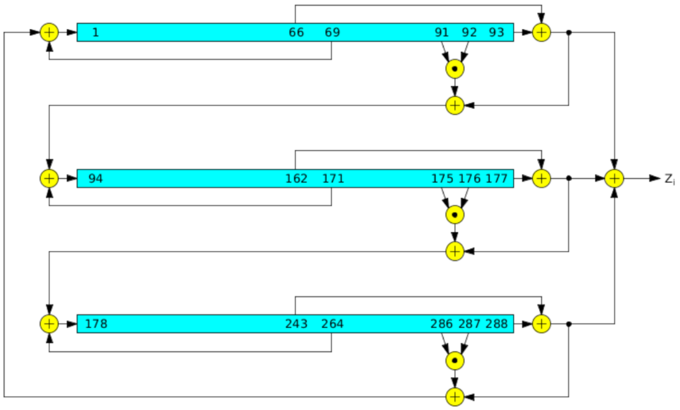
Linear Feedback Shift Registers (LFSRs):Example



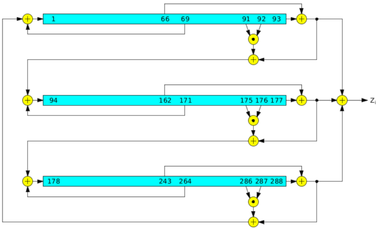
clk	FF_2	FF_1	$FF_0=s_i$
0	1	0	0
1	0	1	0
2	1	0	1
3	1	1	0
4	1	1	1
5	0	1	1
6	0	0	1
7	1	0	0
8	0	1	0

- LFSR output described by recursive equation: $s_{i+3} = s_{i+1} + s_i$ mod 2
- Maximum output length (of $2^3 - 1 = 7$) achieved `only` for certain feedback configurations, .e.g., the one shown here.

Trivium



- Three nonlinear LFSRs (NLFSR) of length 93, 84, 111
- XOR-Sum of all three NLFSR outputs generates key stream z_i
- Small in Hardware: total register count: 288; non-linearity: 3 AND-Gates; 7 XOR-Gates (4 with three inputs)



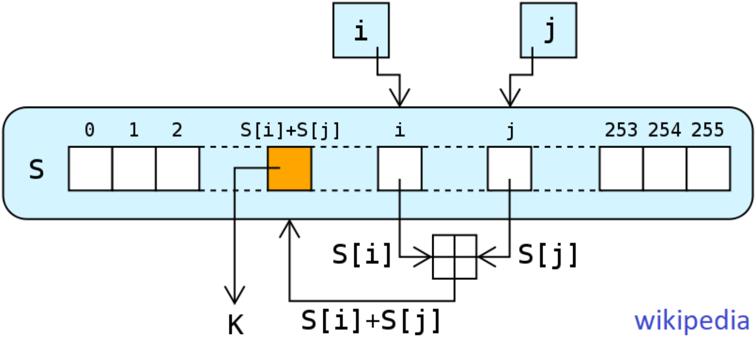
- **Initialization:**
  - $S_1 \ldots S_{80} = $ 80-bit key
  - $S_{94} \ldots S_{173} = $ 80-bit initialization vector (IV) = nonce
  - $S_{286} \ldots S_{288} = 111$
  - Other bits of S = 0
- **Warm-Up:**
  - Clock S 1152 ($= 4 \times 288$) times without generating output
- **Encryption:**
  - XOR-Sum of all three NLFSR outputs generates key stream Z_i

RC4 stream cipher (Rivest Cipher 4)

- The RC4 stream cipher was designed by Ron Rivest for RSA Data Security in 1987
- Algorithm had been a trade secret; allegedly revealed on the Internet in 1994
  - "RC4" is a trademark and cannot be used to refer to an implementation of the algorithm
- The design of RC4 `avoids the use of LFSRs` and is ideal for software implementation
- RC4 generates a keystream (pseudorandom stream of bits), that is used for encryption by combining it with the plaintext using XOR gate (similar to the Vernam cipher)

RC4 stream cipher (Rivest Cipher 4)

- Input: key of length keylength (typically from 40 to 2048 bits)
- Heart: S-box – a permutation of all 256 possible bytes
- Output: a pseudo-random keystream `bytes`
- The RC4 cipher has two components
  - Key Scheduling Algorithm (KSA)
  - Pseudo-Random Generation Algorithm (PRGA)



wikipedia

RC4 stream cipher (Rivest Cipher 4)

- RC4 Key Schedule Algorithm
  The permutation is initialized with a variable length key

Initialization:

for i from 0 to $2^n - 1 = 255$

```
   S[i] := i
endfor
j := 0
```

Scrambling:

```
 for i from 0 to 255
   j := (j + S[i] + key[i mod keylength]) mod 256
   swap values of S[i] and S[j]
endfor
```

RC4 stream cipher (Rivest Cipher 4)

- RC4 Pseudo Random Generation Algorithm

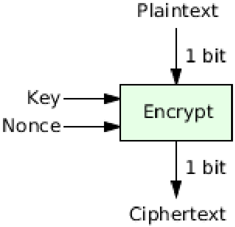
Initialization:

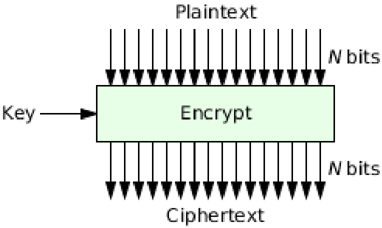
```
i:=0
j := 0
```

Generation Loop:

```
 while GeneratingOutput:
   i := (i + 1) mod 256
   j := (j + S[i]) mod 256
   swap values of S[i] and S[j]
   K := S[(S[i] + S[j]) mod 256]
   output K
endwhile
```

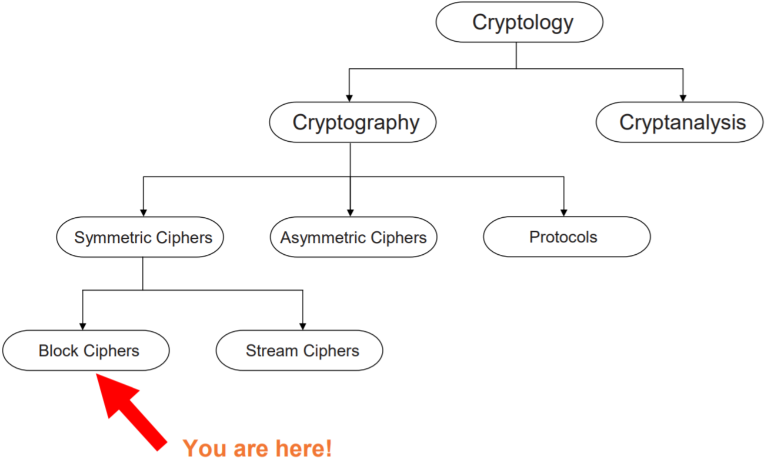
Block Cipher vs Stream cipher





Based on notes by Alan Kaminsky: https://www.cs.rit.edu/ ark/462/module03/notes.shtml

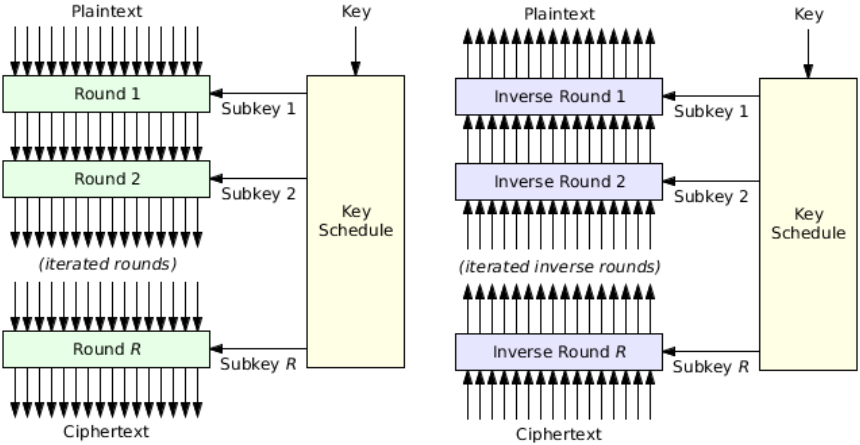
Block Cipher



Block Cipher

- A stream cipher algorithm defines how to encrypt/decrypt arbitrary-length messages
- A block cipher algorithm `does not define` how to encrypt/decrypt arbitrary-length messages
  - You can only encrypt/decrypt N bits, no more, no less
  - Arbitrary-length messages are handled by a separate algorithm called a `block cipher mode of operation` (ECB)
- A stream cipher's encryption and decryption operations are the same
- A block cipher's encryption and decryption operations are `different`
  - To decrypt, run the encryption operation backwards
  - The encryption operation must be invertible

Block Cipher Architecture



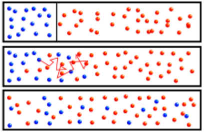
Based on notes by Alan Kaminsky: https://www.cs.rit.edu/ ark/462/module03/notes.shtml

Block Cipher Primitives: Confusion and Diffusion

Claude Shannon: There are two primitive operations with which strong encryption algorithms can be built:

1. `Confusion`: An encryption operation where the relationship between key and ciphertext is obscured.
   Today, a common element for achieving confusion is substitution, which is found in both AES and DES.

2. `Diffusion`: An encryption operation where the influence of one plaintext symbol is spread over many ciphertext symbols with the goal of hiding statistical properties of the plaintext.

   A simple diffusion element is the bit permutation, which is frequently used within DES.

Block Cipher Primitives: Confusion and Diffusion

`Claude Shannon:` Both operations by themselves cannot provide security. The idea is to concatenate confusion and diffusion elements to build so called `product ciphers`.

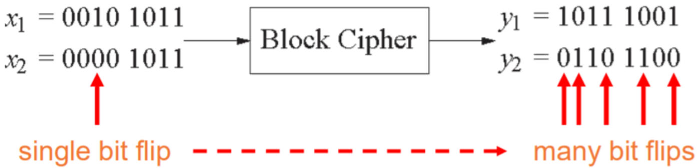

Photo: Alfred Eisenstaedt/The LIFE Picture Collection/Getty Images

- Substitutions
  - Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements
- Permutation
  - No elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed

Product Ciphers

- Most of today's block ciphers are product ciphers as they consist of rounds which are applied repeatedly to the data.
- Can reach excellent diffusion: changing of one bit of plaintext results on average in the change of half the output bits.

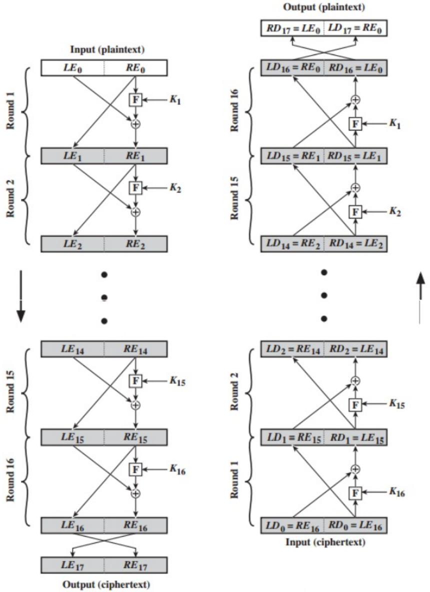
Example:



Feistel Cipher (Feistel network)

- Horst Feistel (IBM) proposed the use of a cipher that alternates substitutions and permutations
- Is a practical application of a proposal by Claude Shannon to develop a product cipher
- A large set of block ciphers use the scheme (it is a design model from which many different block ciphers are derived), including the Data Encryption Standard (DES). `DES is just one example of a Feistel Cipher`
- The ciphertext is calculated from the plaintext by repeated application of the same transformation or round function

Feistel Cipher (Feistel network)



Cryptography Network Security: Principal and Practice by William Stallings

- Input: plaintext block of length $2w$ bits and a key K
- The plaintext block is divided into two halves, L_0 and R_0
- The two halves of the data pass through n rounds of processing and then combine to produce the ciphertext block
- Each round i has as inputs L_{i-1} and R_{i-1} derived from the previous round, as well as a subkey K_i derived from the overall K
- All rounds have the same structure
- A substitution is performed on the left half of the data. This is done by applying a round function F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data
- A Feistel network is a way to turn a one-way (noninvertible) function F into a two-way (invertible) round function

- Advantage: encryption and decryption differ only in keyschedule
- Rounds
  - Plaintext is split into halves L_i and R_i
  - R_i is fed into the function F, the output of which is then XORed with L_i
  - Left and right half are swapped
- Rounds can be expressed as:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \otimes f(R_{i-1}, K_i)$$

Thanks for Your attention.