CS641

Modern Cryptology

Lecture 4

Building Blocks

- A linear transformation on the entire block helps mixing the information well.
- A non-linear transformation is needed to make it secure.
- Therefore, a combination of the two is desirable.
- A large number of encryption algorithms use both.

Mixing Property of Linear Transformations

Theorem

Let $u \in \mathbb{Z}^b, u \neq 0$, and $K \in \mathbb{Z}^{b \times b}$. Then, over random choices of K, $K \cdot u$ is a random vector in $\mathbb{Z}^b$.

- Given u and c with $u \neq 0$, let ith entry of u, u_i, be non-zero.
- Let K_i be the ith column of K.
- Probability that $c = K \cdot u$ equals the probability that $K_i = \frac{1}{u_i}(c - \sum_{1 \leq j \neq i \leq b} u_j K_j)$.
- Fixing all other columns of K except K_i, this is the probability that a random vector K_i equals a fixed vector.

Invertibility of Non-linear Transformations

- Since encrypted text is required to be decrypted, all transformations done during encryption need to be invertible.
- This is easy for linear transformations, but non-linear transformations are typically not invertible.
- So we have to find non-linear transformations that are invertible.
- There is a generic way of doing it given by Feistel.

Feistel Structure

- Let f be any non-linear transformation with $f : \{0,1\}^n \mapsto \{0,1\}^n$.
- Define transformation g, $g : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}^n \times \{0,1\}^n$ as:

$$g(a, b) = (b, a \oplus f(b)),$$

  where $\oplus$ is bitwise XOR.
- g is clearly a non-linear transformation, and it is also invertible:

$$g^{-1}(b, a') = (a' \oplus f(b), b).$$

- Therefore, we can use Feistel structure to ensure invertibility given any transformation f.

Encryption Using Feistel Structure

- Function g transforms only half of input text (b is present in the output).
- We use two rounds of applications of g to transform input completely:

$$g(g(a,b)) = g(b, a \oplus f(b)) = (a \oplus f(b), b \oplus f(a \oplus f(b))).$$

- We can use any number of rounds — generally, greater number of rounds provide more security.

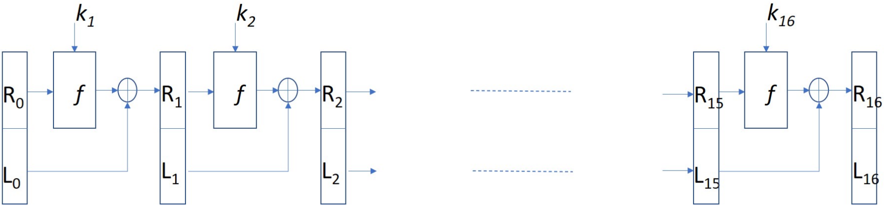
History

- Designed in 1974 by a group of IBM engineers led by Walter Tuchman.
- Adopted by National Bureau of Standards (US) in 1976 as standard and named Data Encryption Standard (DES).
- One of the most widely used encryption algorithm until 2001.

DES Parameters

- A block cipher with blocksize $= 64$ bits, or 8 bytes.
- Key size $= 56$ bits.
- This size was sufficient in 1970s to be resistant against brute-force attacks.
- Uses Feistel structure with 16 rounds.

DES Structure



- $R_{i+1} = L_i \oplus f(R_i, k_i)$ for $0 \leq i < 16$
- $L_{i+1} = R_i$ for $0 \leq i < 16$.
- Function f also depends on round key k_i.

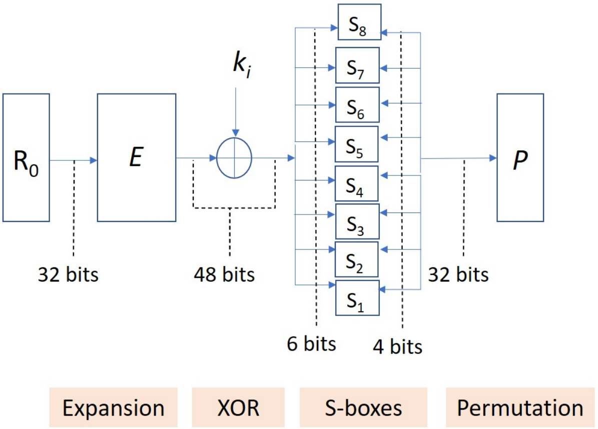
Round Operations

- Key k_i is called round key for round i, $1 \leq i \leq 16$.
- Each round key is 48 bits long.
- Each is a fixed subset of 56 bits of key
  - Subsets are different for different rounds, but fixed.
- Plaintext block is $L_0 R_0$ with $|L_0| = |R_0| = 32$ bits.
- Input to round $i+1$ is $L_i R_i$ and its output is $L_{i+1} R_{i+1}$, with $|L_{i+1}| = |R_{i+1}| = 32$ bits.
- As per Feistel structure, $L_{i+1} = R_i$ for $0 \leq i < 16$.

Function f

- Function f is a non-linear function.
- It takes as input right half of round input (of 32 bits) and round key (of 48 bits), and produces a 32 bit output.
- It can be further divided into a series of four operations, three of which are linear and one is non-linear.

Function f



- $R_{i+1} = L_i \oplus f(R_i, k_i)$ for $0 \leq i < 16$
- $L_{i+1} = R_i$ for $0 \leq i < 16$.
- Function f also depends on round key k_i.

Expansion E

- Takes 32 bit input and produces 48 bit output.
- Replicates 16 bits of input in the following way:
  - Input: $b_0 b_1 \cdots b_{31}$
  - Output: $b_{31} b_0 b_1 b_2 b_3 b_4 b_3 b_4 b_5 b_6 b_7 b_8 b_7 b_8 \cdots b_{29} b_{30} b_{31} b_0$

PERMUTATION P

- Shuffles input bits as:
  - Input: $b_0 b_1 b_2 \cdots b_{31}$
  - Output: $b_{15} b_7 b_{19} b_{20} b_{28} b_{11} b_{27} b_{16} \cdots b_{21} b_{10} b_3 b_{24}$
- Primary aim is to shuffle bits so that in all 4-bits in a block move to different blocks, for each of the eight blocks.

S-boxes

- Only nonlinear operation in entire algorithm
- There are eight S-boxes, each mapping six bits to four bits.
- Each of the eight boxes are distinct transformations.

S-1

S-1	0000	0001	0010	0011	0100	0101	0110	0111
00	14	4	13	1	2	15	11	8
01	0	15	7	4	14	2	13	1
10	4	1	14	8	13	6	2	11
11	15	12	8	2	4	9	1	7
S-1	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
00	3	10	6	12	5	9	0	7
01	10	6	12	11	9	5	3	8
10	15	12	9	7	3	10	5	0
11	5	11	3	14	10	0	6	13

- Columns indexed by middle four bits of input, and rows indexed by first and last bits of input.
- Numbers are between 0 to 15 representing four bit outputs.
- Every row has all 16 numbers occurring once.

Design Choices

- Why 56 bit key size? Why not 64 bits?
  - Key is stored in 64 bits. In each byte, msb is used to do parity check of seven bits of key.
  - To catch any error occurring in other seven bits.
- Why so small S-boxes?
  - To store S-box tables in hardware so that algorithm can be executed fast.
  - Same reason for other choices of operations.