ENEE 457: Computer Systems Security
09/12/16

Lecture 4
Symmetric Key Encryption II: Security Definitions and Practical Constructions

Charalampos (Babis) Papamanthou

Department of Electrical and Computer Engineering
University of Maryland, College Park

Announcements

- HW1 is due Wed
- Lab 1 is due Sat
- For Bitcoin Research: Teams must form by Wed and I expect progress by next Monday

Recall the definition of PRP's

- We say that a length-preserving keyed function F: $\{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$, is a keyed permutation if and only if each F_k is a bijection
- Also, **for security** an adversary could not distinguish between the following two worlds with probability more than ½+2^{-k}
  - He sends *x* to World1, World1 chooses a random permutation A and returns A[x]
  - He sends *x* to World2, World2 chooses a random key k and returns $F_k(x)$
- How do we encrypt using PRPs a message m of n bits?
  - **Enc**$_k$(m): $c := \langle r, F_k(r) \oplus m \rangle$
    - where $r \leftarrow \{0,1\}^n$ is chosen at uniform random
  - **Dec**$_k$(c): given $c=\langle r, s \rangle$, $m := F_k(r) \oplus s$
- Let's call the above scheme **`First_Symmetric`**

Question 2

- Why **`First_Symmetric`** is secure?

Intuitively this is secure: so long as r is not used for different messages, $F_k(r)$ should look completely random

- But this is just intuition

Semantic security (CPA)

- I give you a symmetric encryption scheme (Enc,Dec,K)
- What do you need to prove in order to say that it is secure?
- A strong notion used is “semantic security”
- We are going to define it as an interaction between the adversary **A** and a trusted party **T** that has the secret key.
- Informally:
  1. **T** picks a random secret key
  2. **A** picks messages m_i and receives ciphertexts Enc_K(m_i) from **T**.
  3. **A** picks message m_0 and m_1 and sends them to **T**.
  4. **T** flips a coin b and computes t_b=Enc_K(m_b).
  5. **T** sends t_b to the **A**.
- The scheme is secure if **A** has no better chance of finding whether t_b corresponds to m_0 or m_1 than ½+2^{-k}
- This should hold even if it is repeated many (polynomial) times

Question 3

- What behavior of the adversary does this definition model?

- Think emails…

Question 4

- Why **`First_Symmetric`** without randomness r is **not** semantically secure?

- Provide an attack where the adversary's chance of finding where t_b corresponds to is 1.

Task 1

- Prove `First_Symmetric` is semantically secure
  - Suppose it is not. That means that the adversary A, given
    - m_0 and m_1
    - $c\_b = F_k(r) \oplus m\_b$ (where b = 0 or b = 1)

  can figure out whether b = 0 or b = 1. We distinguish two cases:

  1. If m_b was chosen before, due to the "random" r and the "randomness" of $F_k(r)$, $F_k(r)$ appears "random" (cannot be distinguished from a truly random permutation) , so $F_k(r) \oplus m\_b$ appears "random" and does not give any information about m_b, a contradiction.
  2. If not, due to the "randomness" of $F_k(r)$, $F_k(r)$ appears "random", so $F_k(r) \oplus m\_b$ appears "random" " and does not give any information about m_b, a contradiction.

- So in both cases we reach a contradiction

More advanced security (CCA)

- Informally:
  - **T** picks a random secret key
  - **A** picks messages m_i and receives ciphertexts Enc_K(m_i) from **T**.
  - **A** picks message m_0 and m_1 and sends them to **T**.
  - **T** flips a coin b and computes t_b=Enc_K(m_b).
  - **T** sends t_b to the **A**.
  - **A** sends a ciphertext of its choice**, different than t_b**, for decryption
  - The scheme is secure if **A** has no better chance of finding whether t_b corresponds to m_0 or m_1 than ½+2^{-k}
- This should hold even if it is repeated many (polynomial) times

Question 5

- What behavior of the attacker does this model?
- Lunch-time attacks…

Is `First_Symmetric` CCA-secure?

- Ask encryption for $m_0 = 0000\ldots00$ and $m_1 = 1111\ldots11$
- You get $c_b = < s_b, r_b >$, where $s_b = F_k(r_b) \oplus m_b$
- How to find b is you are allowed to send decryption queries?
- Construct new new ciphertext
  - $c = < s_b \oplus 1000\ldots00, r_b > = < F_k(r_b) \oplus m_b \oplus 1000\ldots00, r_b >$
  - Decryption of this will give $m_b \oplus 1000\ldots00$
    - $1000\ldots00$, if s_b was encryption of $m_0 = 0000\ldots00$
    - $01111\ldots1$, if s_b was encryption of $m_1 = 1111111....1111$
- So we can distinguish!
- Conclusion: `First_Symmetric` is not CCA-secure.

How do we construct a PRP in practice?

- What is the main property we want?
  - Even a single bit change in the input should yield a completely independent result
- This implies that
  - Every bit of the input should affect every bit of the output…
  - Or…every change in an input bit should change each output bit with probability roughly ½
- This takes some work…

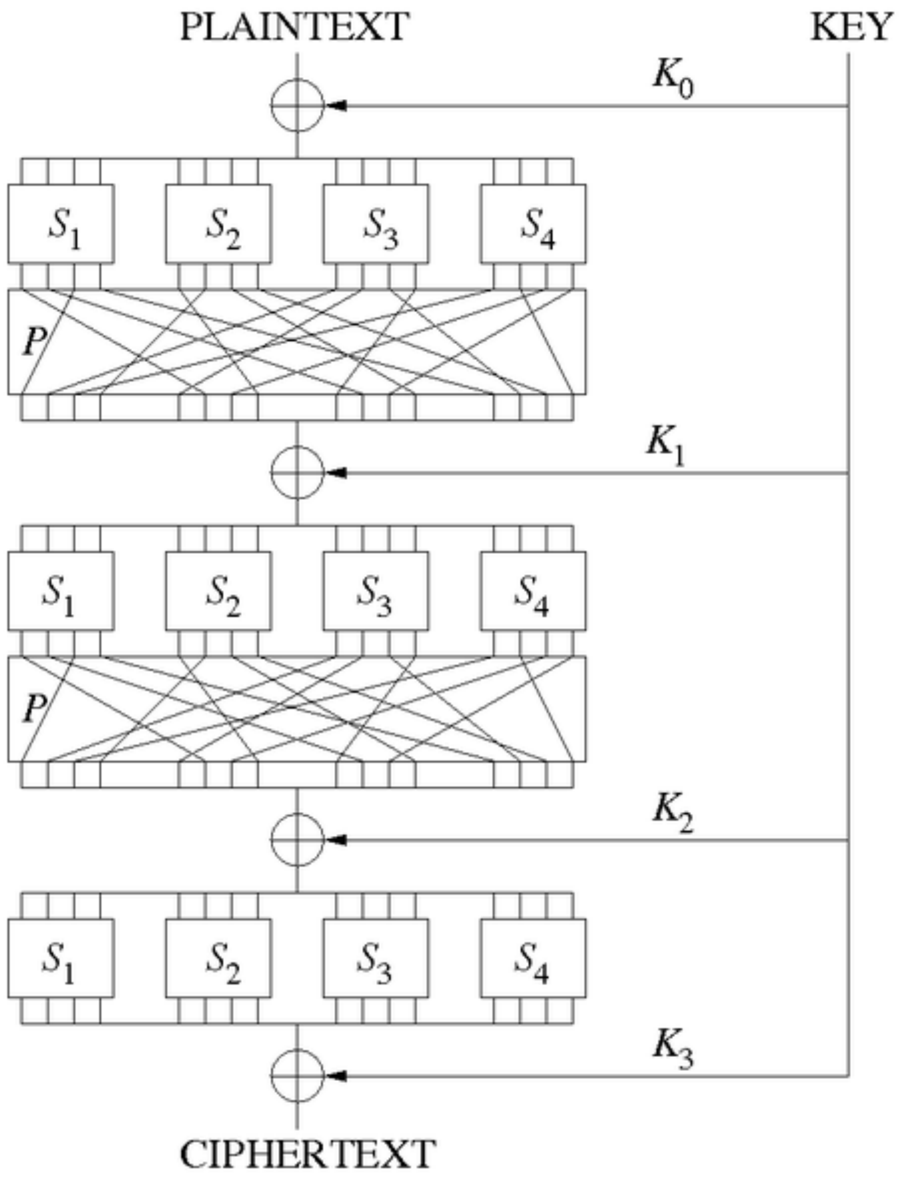
A first idea (Shannon)

- Construct block cipher from many smaller random (or random-looking) permutations
- **Confusion:** e.g., for block size 128, uses 16 8-bit random permutation
  - $F_k(x) = f_1(x_1) \ldots f_{16}(x_{16})$
  - Where key k selects 16 8-bit random permutation.
  - Does $F_k(\cdot)$ look like a random permutation?
- **Diffusion:** bits of $F_k(x)$ are permuted (re-ordered)
- Multiple rounds of confusion and diffusion are used.

Substitution-Permutation Networks

- A variant of the Confusion-Diffusion Paradigm
  - $\{f_i\}$ are fixed and are called s-boxes
  - Sub-keys are XORed with intermediate result
    - Sub-keys are generated from the master key according to a key schedule
- Each round has three steps
  - Message XORed with sub-key
  - Message divided and went through s-boxes
  - Message goes through a mixing permutation (bits reordered)

Substitution-Permutation Networks



Design Principles:

---A single-bit difference in each s-box results in changes in at least two bits in output

---The mixing permutation distributes the output bits of any s-box into multiple s-boxes

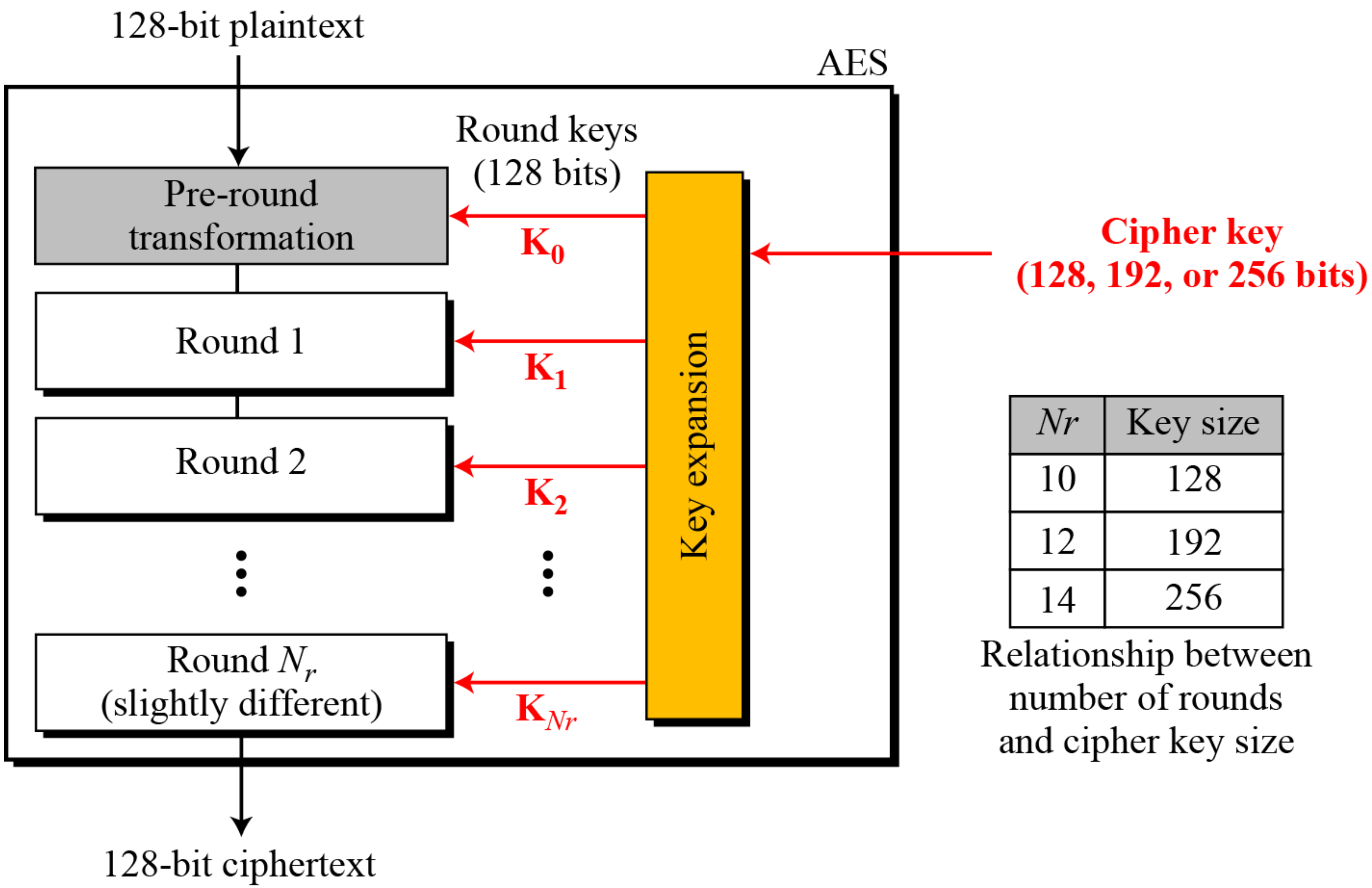
The above, with sufficient number of rounds, achieves the avalanche effect.

AES encryption, the algorithm of choice in today's Internet communications is using the above framework

Question 6

- How can you attack one round?
- How can you attack two rounds?

AES structure

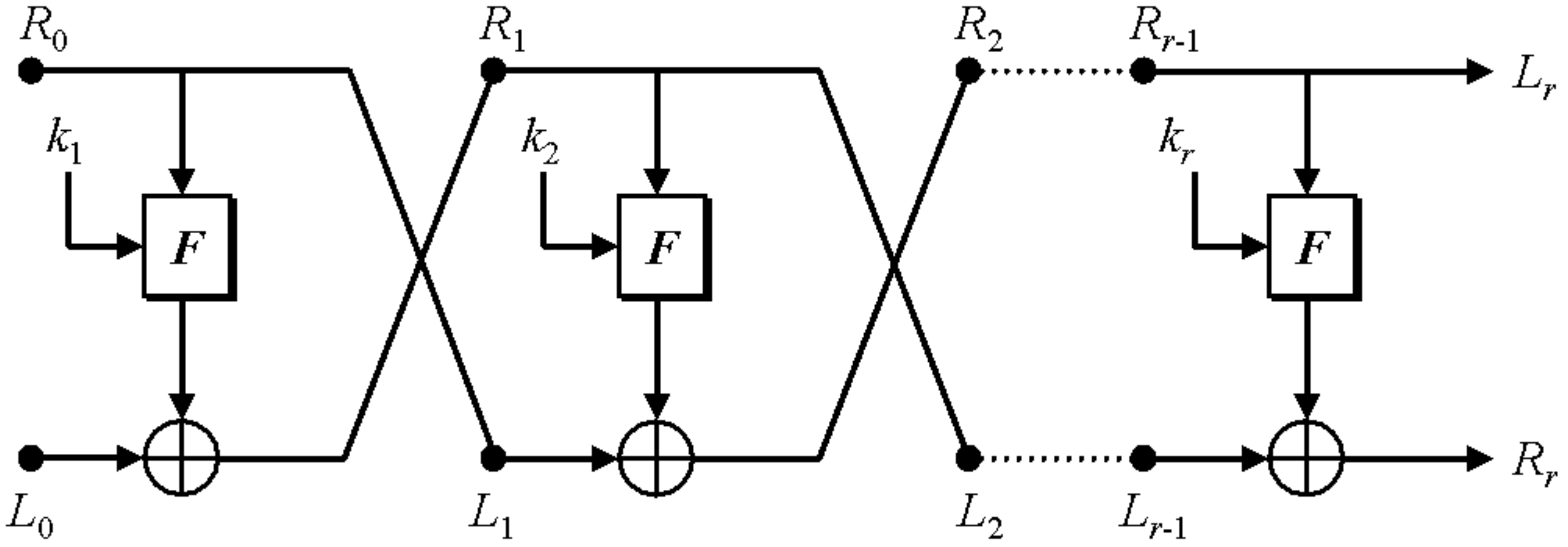


Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

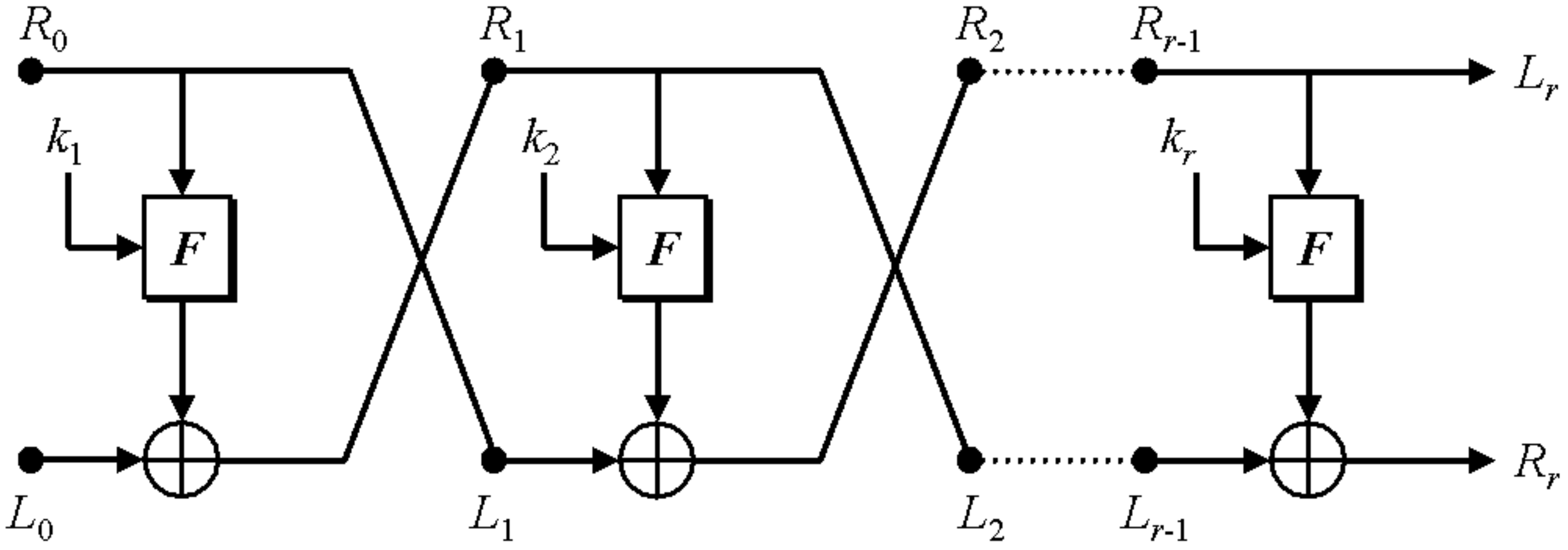
Second approach: Feistel Network

- Feistel Networks



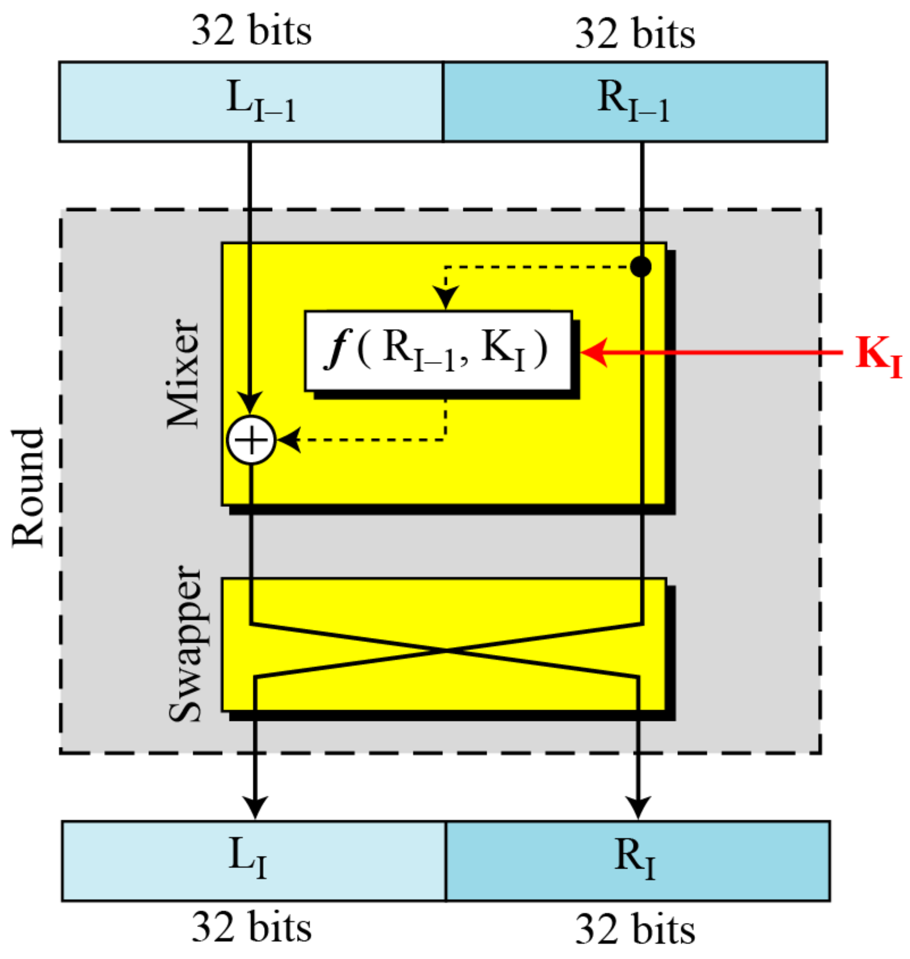
Feistel Network

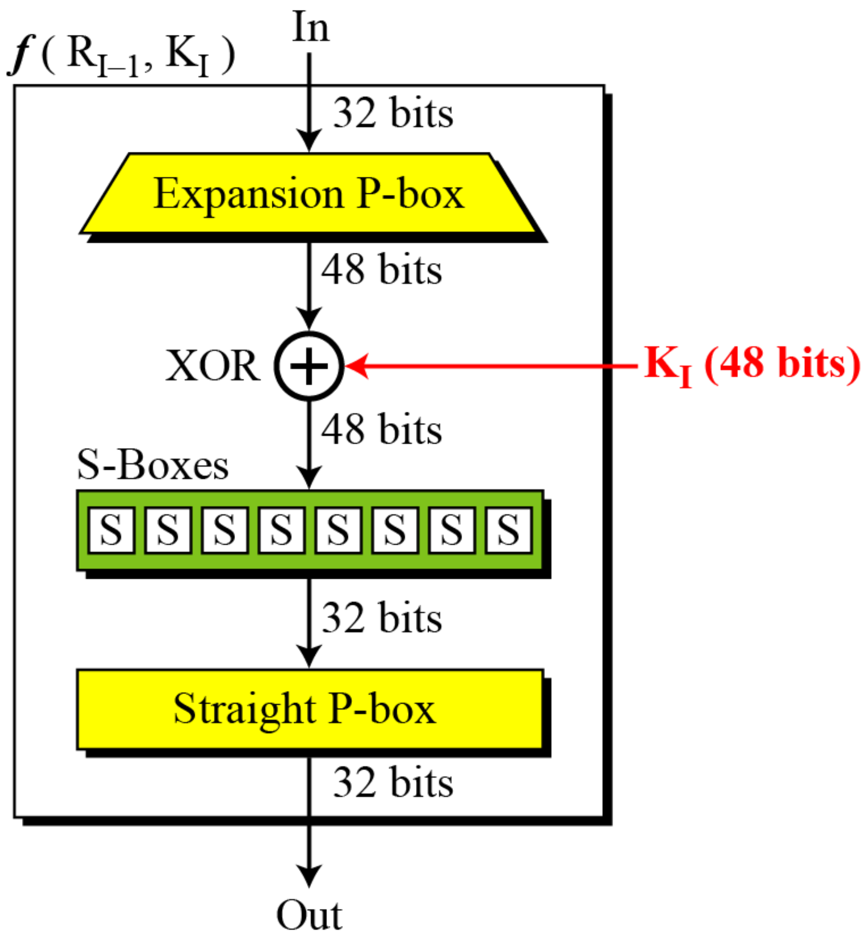
- Main difference: F does not have to be invertible
- In practice: It is a Substitution-permutation network
- DES was based on that (broken, not because of bad design, but due to the size of the key)



DES function

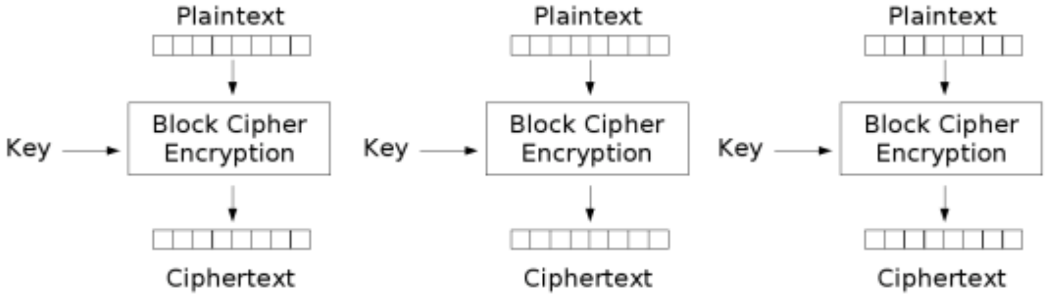
The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output





Block Cipher Modes

- So far we have described how to encrypt a string of fixed length
- How do we encrypt a 4GB file?
- Electronic Code Book (ECB) Mode (is the simplest):
  - Block P[i] encrypted into ciphertext block C[i] = E_K(P[i])
  - Block C[i] decrypted into plaintext block M[i] = D_K(C[i])



Electronic Codebook (ECB) mode encryption

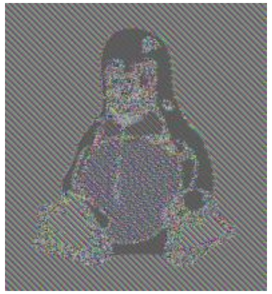
Public domain images from http://en.wikipedia.org/wiki/File:Ecb_encryption.png and http://en.wikipedia.org/wiki/File:Ecb_decryption.png

Strengths and Weaknesses of ECB

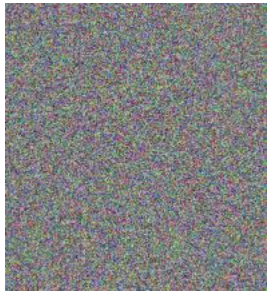
- Strengths:
  - Is very simple
  - Allows for parallel encryptions of the blocks of a plaintext
  - Can tolerate the loss or damage of a block

- Weakness:
  - Documents and images are not suitable for ECB encryption since patterns in the plaintext are repeated in the ciphertext:

ECB

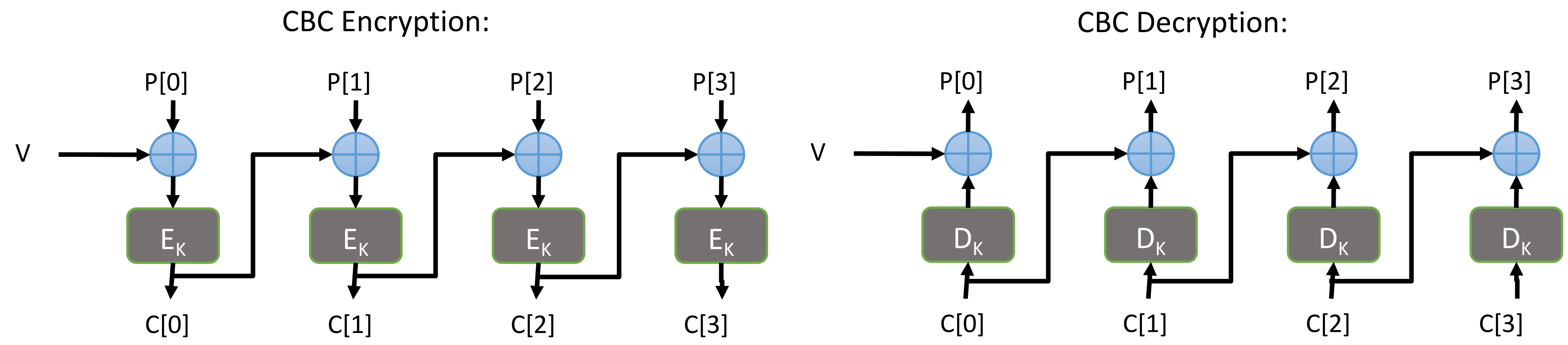


CBC



Cipher Block Chaining (CBC) Mode

- In Cipher Block Chaining (CBC) Mode
  - The previous ciphertext block is combined with the current plaintext block $C[i] = E_K (C[i-1] \oplus P[i])$
  - $C[-1] = V$, a random block separately transmitted encrypted (known as the initialization vector)
  - Decryption: $P[i] = C[i-1] \oplus D_K (C[i])$



Question 7

- Is CBC encryption parallelizable?
- Is CBC decryption parallelizable?

OpenSSL encryption decryption

- openssl aes-256-cbc -a -in plaintext.txt -out ciphertext.txt

- openssl aes-256-cbc -a -d -in ciphertext.txt -out plaintext.txt