Introduction to Cryptography

Lecture 2

Benny Pinkas

Perfect Cipher

- What type of security would we like to achieve?
- "Given C, the adversary has no idea what M is"
  - Impossible since adversary might have a-priori information
- In an "ideal" world, the message will be delivered in a magical way, out of the reach of the adversary
  - We would like to achieve similar security
- Definition: a *perfect cipher*
  - $Pr(\ plaintext = P \mid ciphertext = C\ ) = Pr(\ plaintext = P)$

Perfect Ciphers

- A simple criteria for perfect ciphers.
- Claim: The cipher is perfect if, and only if,

  $\forall\ m_1, m_2 \in M$, $\forall$cipher c,

  $\Pr(\mathrm{Enc}(m_1)=c) = \Pr(\mathrm{Enc}(m_2)=c)$. (homework)

- Idea: Regardless of the plaintext, the adversary sees the same distribution of ciphertexts.

- Note that the proof cannot assume that the cipher is the one-time-pad, but rather only that *Pr(plaintext = P | ciphertext = C) = Pr(plaintext = P)*

Size of key space

- Theorem: For a perfect encryption scheme, the number of keys is at least the size of the message space.

- Proof:
  - Consider ciphertext C.
  - Must be a possible encryption of any plaintext m.
  - But, need a different key per message m.

- Corollary: Key length of one-time pad is optimal ☹

Computational security

- We should only worry about polynomial adversaries
- Idea: Generate a string which “looks random” to any polynomial adversary. Use it instead of a OTP.

- Looks random?
  - Fraction of bits set to 1 is ≈ 50%
  - Longest run of 0’s is of length ≈ log(n),
  - Is that sufficient?...

- Enumerating a set of statistical tests that the string should pass is not enough.

Computational security – Pseudo-randomness

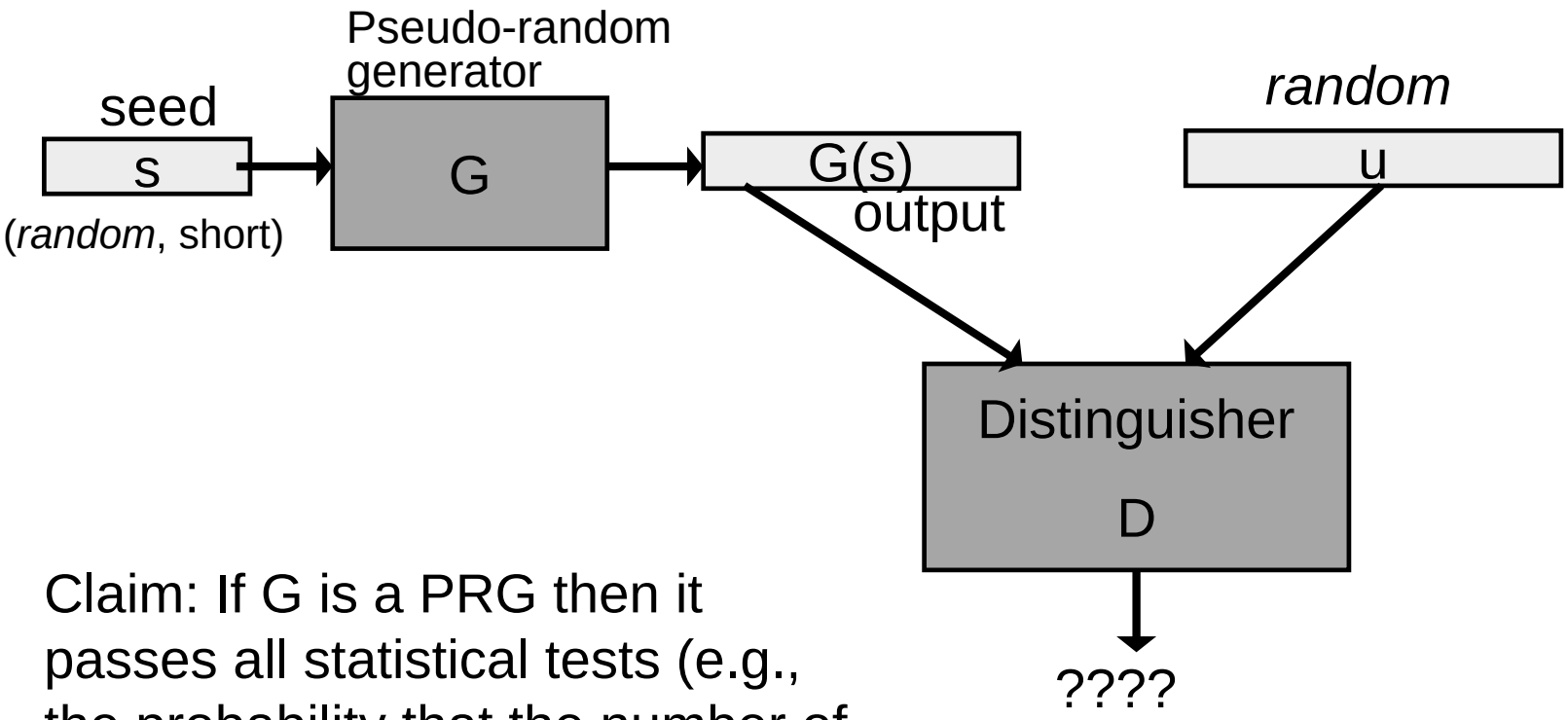
- Pseudo-random string: no *efficient* observer can *distinguish* it from a uniformly random string of the same length
- Motivation: *Indistinguishable objects are equivalent*
- The foundation of modern cryptography

- *(t,ε)*-Pseudo-random generator (PRG)
  - $G: \{0,1\}^{|k|} \Rightarrow \{0,1\}^{|m|}$ $|k|<|m|$, polynomially computable.
  - $\forall$ adversary D running in time *t*,
    for $s \in_R \{0,1\}^{|k|}$, $u \in_R \{0,1\}^{|m|}$,
    it holds that $\Pr(D(G(s)) \neq D(u) < \varepsilon$

Pseudo-random generators

- Pseudo-random generator (PRG)
  - G: $\{0,1\}^{|k|} \Rightarrow \{0,1\}^{|m|}$ $|k|<|m|$, polynomially computable.
  - $\forall$ <u>polynomial time</u> adversary D,
    for $s \in_R \{0,1\}^{|k|}$, $u \in_R \{0,1\}^{|m|}$,
    it holds that $\Pr(D(G(s)) \neq D(u)$ is <u>negligible</u>
  - *Polynomial time:* running in time $t(n)$ s.t. $\exists$polynomial $p()$ for which $t(n)<p(n)$ for all large enough n
  - *Negligible:* the difference is a function $\varepsilon(n)$ s.t. $\forall$polynomials $q()$, for all large enough n it holds that $\varepsilon(n) < q(n)$

Pseudo-random generator



Claim: If G is a PRG then it passes all statistical tests (e.g., the probability that the number of 1 bits is < |m|/3 is negligible).

Using a PRG for Encryption

- Key: a (short) random seed $s \in \{0,1\}^{|k|}$.
- Message $m = m_1,\ldots,m_{|m|}$.

- Encryption:
  - Use the output of the PRG as a one-time pad. Namely,
  - Generate $G(s) = g_1,\ldots,g_{|m|}$
  - Ciphertext $C = g_1 \oplus m_1,\ldots, g_{|m|} \oplus m_{|m|}$

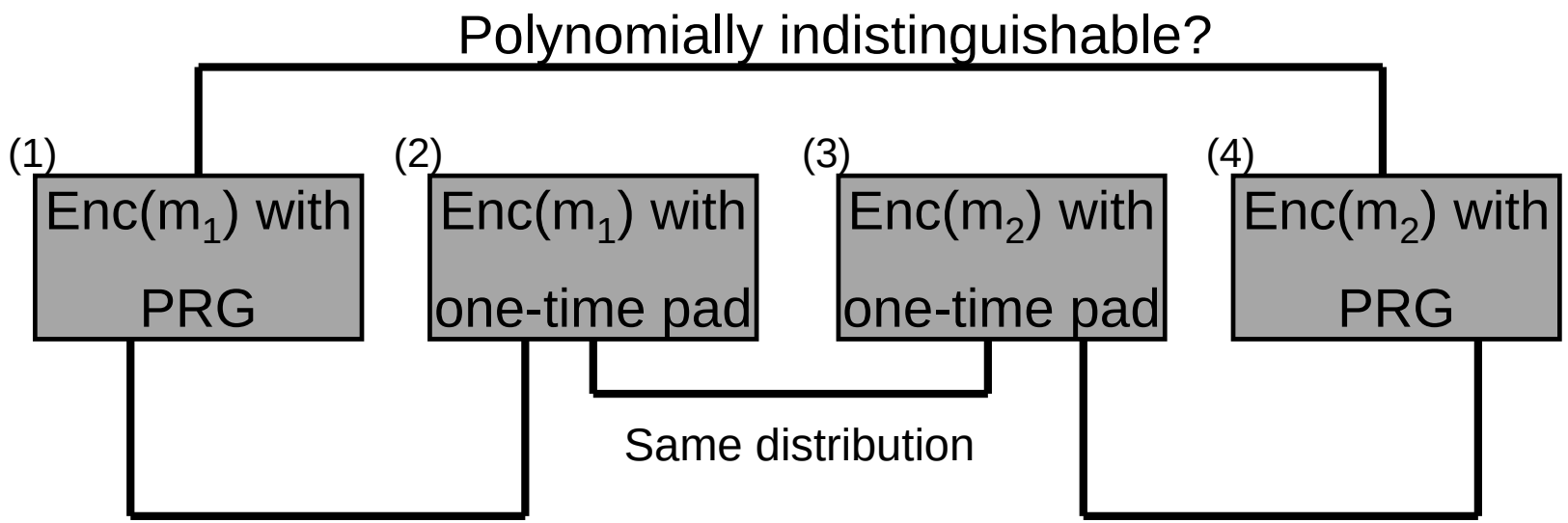
Using a PRG for Encryption: Security

- One time pad:
  - $\forall\ m_1, m_2 \in M$, $\forall c$, the probability that c is an encryption of m_1 is equal to the probability that c is an encryption of m_2.
  - I.e., $\forall\ m_1, m_2 \in M\ \forall c$, it is impossible to tell whether c is an encryption of m_1 or of m_2.

- Security of pseudo-random encryption:
  - Show that $\forall\ m_1, m_2 \in M$, no *polynomial time* adversary can distinguish between the encryptions of m_1 and of m_2.

- Proof by reduction: if one can break the security of the encryption (distinguish between encryptions of m_1 and of m_2), it can also break the security of the PRG (distinguish it from random).

Proof of Security



Indistinguishable since otherwise PRG output was distinguishable from a random string

Indistinguishable since otherwise PRG output was distinguishable from a random string

Distinguishing between (1) and (4), implies distinguishing between (1) and (2), or (2) and (3), or (3) and (4).

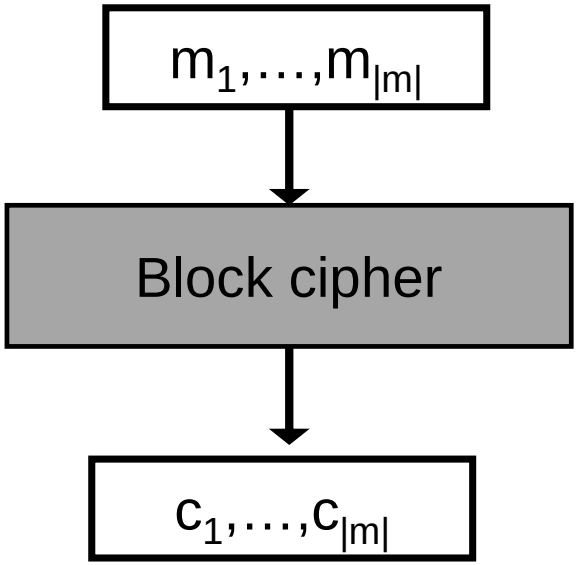
Symmetric systems used in practice

- Are not based on computational problems
- Are (usually) not proven secure by reductions
- Are designed for specific input and key lengths
- Are very efficient

- Stream ciphers
  - Meant to implement a pseudo-random generator
  - Usually used for encryption in the same way as OTP
  - Examples: A5, RC4, SEAL.
  - Require *synchronization*

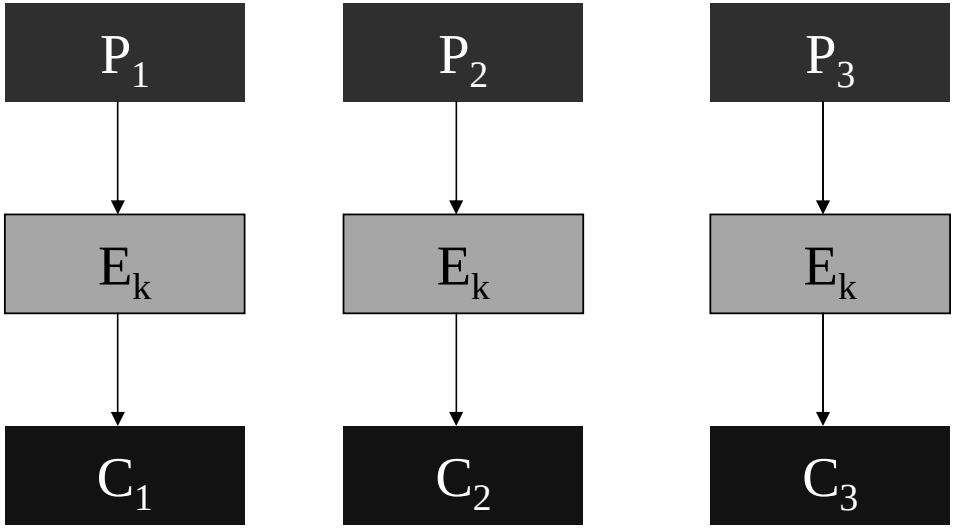
Block Ciphers

- Plaintexts, ciphertexts of fixed length, |m|. Usually, |m|=64 or |m|=128 bits.
- The encryption algorithm E_k is a *permutation* over $\{0,1\}^{|m|}$, and the decryption D_k is its inverse.
- Ideally, use a *random* permutation. Instead, use a *pseudo-random* permutation, keyed by a key k.



- Encrypt/decrypt whole blocks of bits
  - Might provide better encryption by simultaneously working on a block of bits
  - Error propagation: one error in ciphertext affects whole block
  - Delay in encryption/decryption
- Different modes of operation

ECB Encryption Mode (Electronic Code Book)

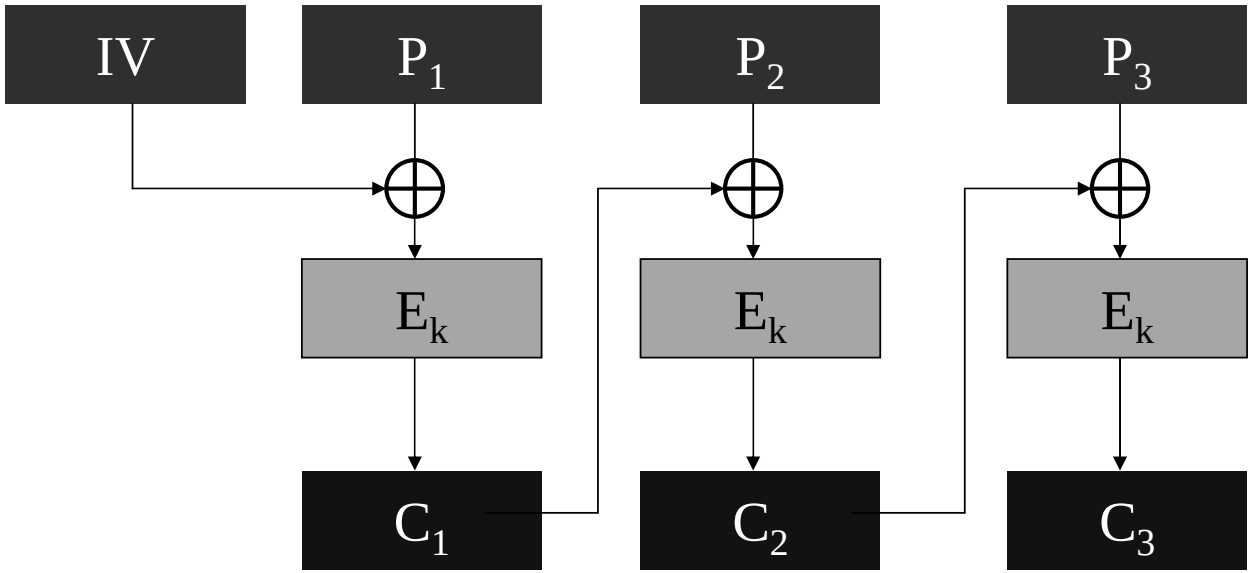


Namely, encrypt each plaintext block separately.

Properties of ECB

- Simple and efficient
- Parallel implementation is possible
- Does not conceal plaintext patterns
  - Enc(P_1, P_2, P_1, P_3)
- Active attacks are possible (plaintext can be easily manipulated by removing, repeating, or interchanging blocks).

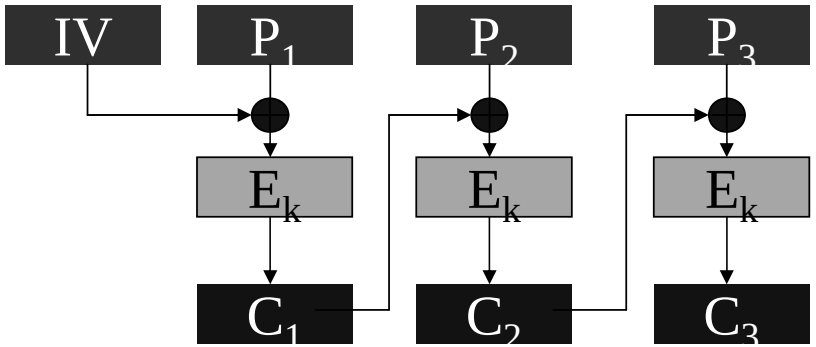
CBC Encryption Mode (Cipher Block Chaining)



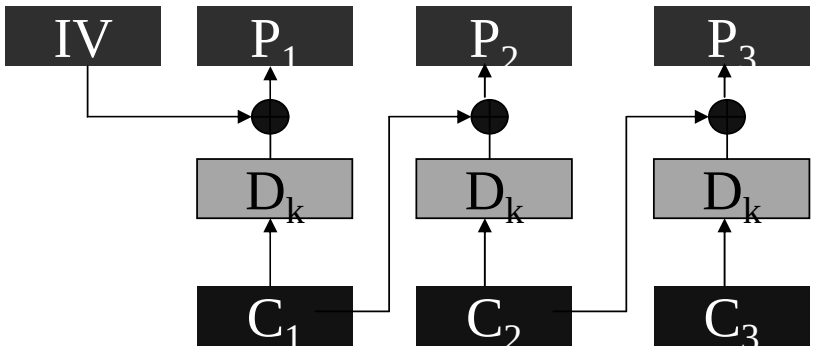
Previous *ciphertext* is XORed with current *plaintext* before encrypting current block.
An initialization vector IV is used as a “seed” for the process.
IV can be transmitted in the clear (unencrypted).

CBC Mode

Encryption:



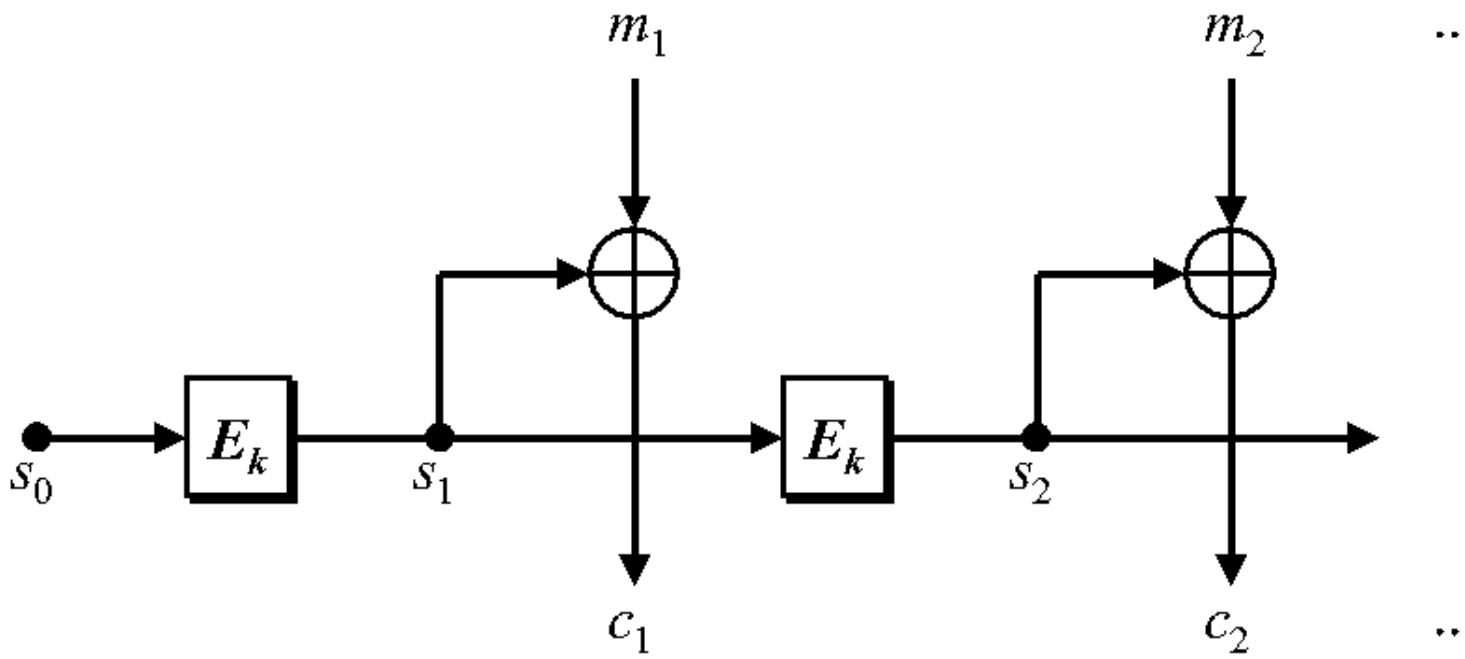
Decryption:



Properties of CBC

- Asynchronous: the receiver can start decrypting from any block in the ciphertext.
- Errors in one *ciphertext* block propagate to the decryption of the next block (but that's it).
- Conceals plaintext patterns (same block -> different ciphertext blocks)
  - But if IV is fixed, CBC does not hide not common prefixes
- No parallel implementation is known
- Plaintext cannot be easily manipulated.
- Standard in most systems: SSL, IPSec, etc.

OFB Mode (Output FeedBack)



- An initialization vector s_0 is used as a ``seed'' for generating a sequence of “pad” blocks s_i. ($s_i=E_k(s_{i-1})$)
- Essentially a stream cipher
- s_0 can be sent in the clear.

Properties of OFB

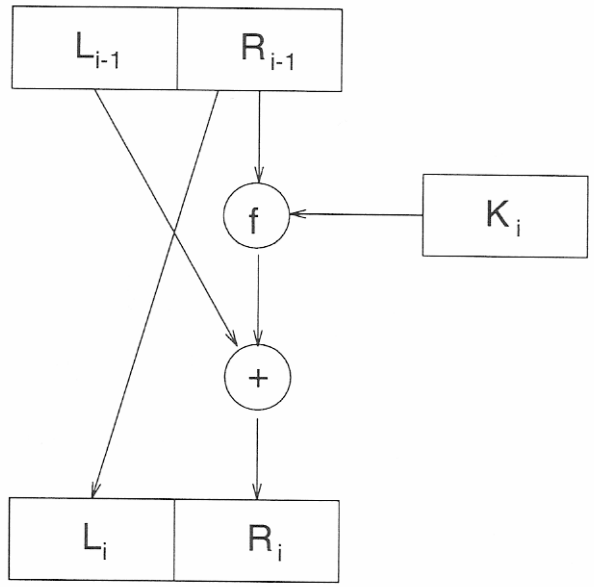
- Synchronous stream cipher. I.e., the two parties must know s_0 and the current bit position.
- The parties must synchronize the location they are encrypting/decrypting.
- Errors in ciphertext do not propagate
- Implementation:
  - Pre-processing is possible
  - No parallel implementation known
- Conceals plaintext patterns
- Active attacks (by manipulating the plaintext) are possible

Design of Block Ciphers

- More an art/engineering challenge than science. Based on experience and public scrutiny.
- *"Diffusion":* each intermediate/output bit affected by many input bits
- *"Confusion"*: avoid structural relationships between bits
- Cascaded (round) design: the encryption algorithm is composed of iterative applications of a simple round
- A common round function: Feistel network

Feistel Networks

- Encryption:
- *Input:* $P = L_{i-1} \mid R_{i-1}$. $|L_{i-1}|=|R_{i-1}|$
  - $L_i = R_{i-1}$
  - $R_i = L_{i-1} \oplus F(K_i, R_{i-1})$
- Decryption?



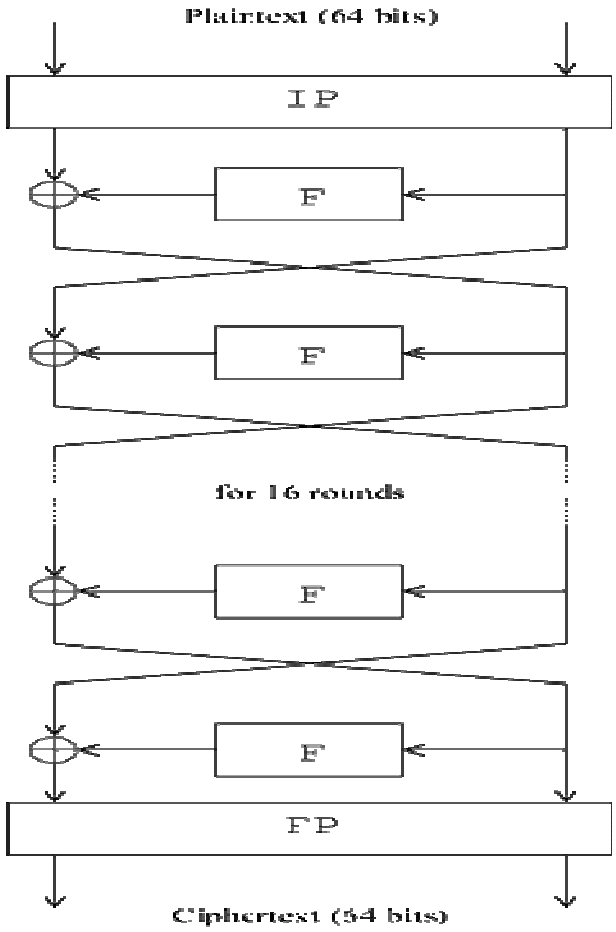
- No matter which function is used as F, we obtain a permutation (i.e., F is reversible).
- The same code/circuit, with keys is reverse order, can be used for decryption.
- Theoretical result [LubRac]: If F is a pseudo-random function then 4 rounds give a pseudo-random permutation

DES (Data Encryption Standard)

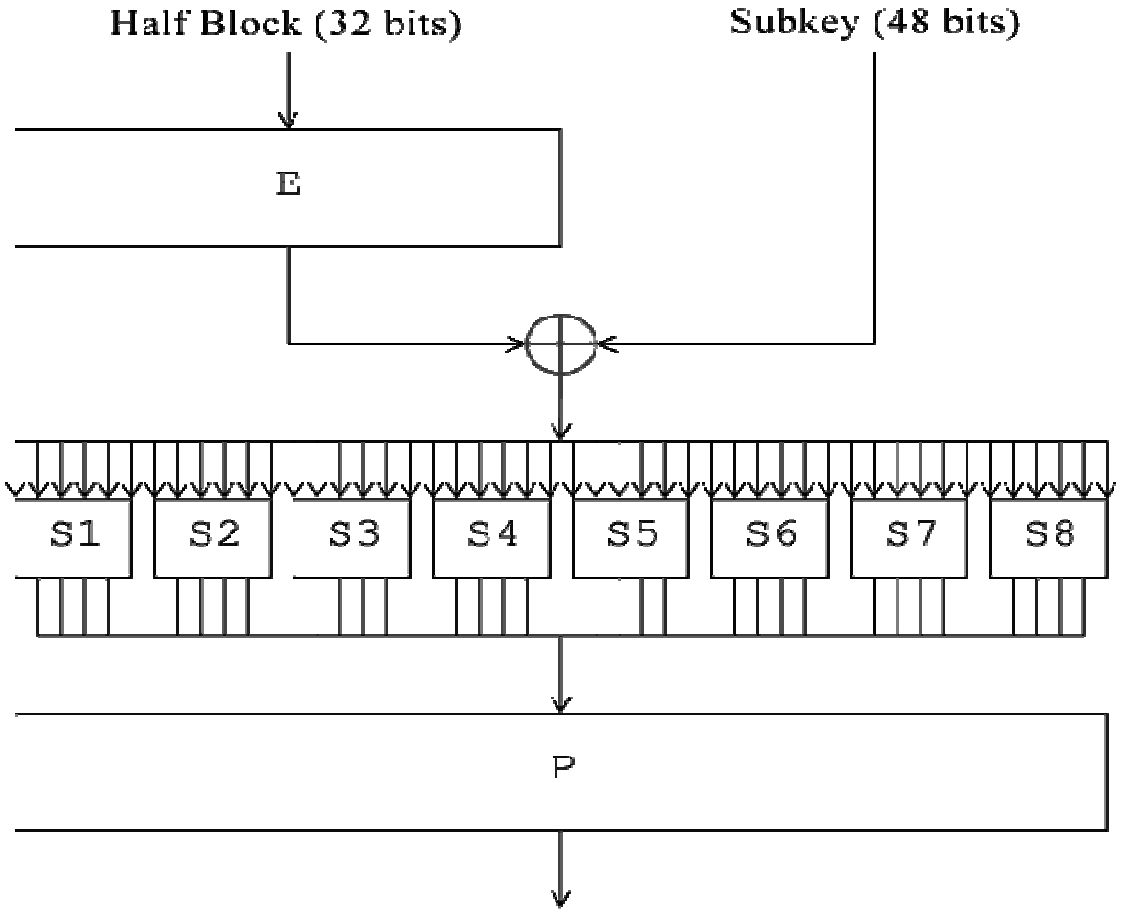
- A Feistel network encryption algorithm:
  - How many rounds?
  - How are the round keys generated?
  - What is F?

- DES (Data Encryption Standard)
  - Designed by IBM and the NSA, 1977.
  - 64 bit input and output
  - 56 bit key
  - 16 round Feistel network
  - Each round key is a 48 bit subset of the key
- Throughput ≈ software: 10Mb/sec, hardware: 1Gb/sec (in 1991!).
- Criticized for unpublished design *decisions* (designers did not want to disclose differential cryptanalysis).
- Linear cryptanalysis: about 2^{40} known plaintexts

DES diagram

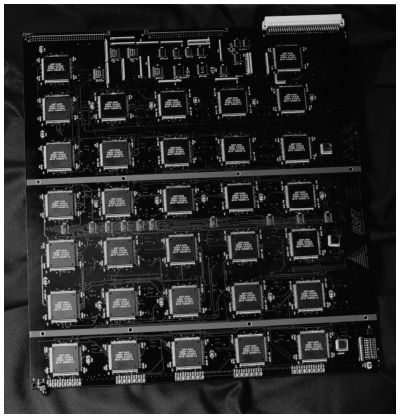

Plaintext (64 bits)
IP
F
F
for 16 rounds
F
F
FP
Ciphertext (64 bits)

DES F functions



Double DES

- DES is out of date due to brute force attacks on its short key (56 bits)



- Why not apply DES twice with two keys?
  - Double DES: $DES_{k1,k2} = E_{k2}(E_{k1}(m))$
  - Key length: 112 bits
- But, double DES is susceptible to a meet-in-the-middle attack, requiring $\approx 2^{56}$ operations and storage.
  - Compared to brute a force attack, requiring 2^{112} operations and O(1) storage.

Meet-in-the-middle attacks

- Meet-in-the-middle attack
  - $c = E_{k2}(E_{k1}(m))$
  - $D_{k2}(c) = E_{k1}(m)$

- The attack:
  - Input: *(m,c)* for which $c = E_{k2}(E_{k1}(m))$
  - For every possible value of k_1, generate and store $E_{k1}(m)$
  - For every possible value of k_2, check if $D_{k2}(c)$ is in the table
  - Might obtain several options for (k_1,k_2). Check them or repeat the process again with a new *(m,c)* pair.

- The attack is applicable to any iterated cipher

Triple DES

- $3DES_{k1,k2} = E_{k1}(D_{k2}(E_{k1}(m))$
- Why use Enc(Dec(Enc( ))) ?
  - Backward compatibility: setting $k_1=k_2$ is compatible with single key DES
- Only two keys
  - Effective key length is 112 bits
  - Why not use three keys? There is a meet-in-the-middle attack with 2^{112} operations
- Provides good security. Widely used. Less efficient.

AES (Advanced Encryption Standard)

- Design initiated in 1997 by NIST
  - Goals: improve security and software efficiency of DES
  - 15 submissions, several rounds of public analysis
  - The winning algorithm: Rijndael

- Input block length: 128 bits
- Key length: 128, 192 or 256 bits
- Multiple rounds (10, 12 or 14), but does not use a Feistel network

What we've learned today

- Perfect security implies $|M| \leq |K|$
- Computational security
- Pseudo-randomness, Pseudo-random generator
- Block ciphers
- DES, AES
- Meet in the middle attack