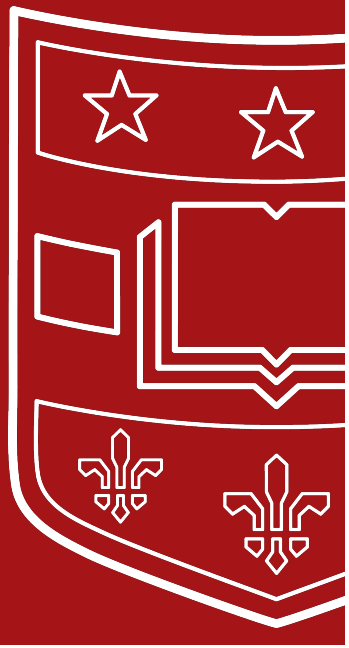
CSE 433S: Introduction to Computer Security

Block Cipher



Washington University in St. Louis

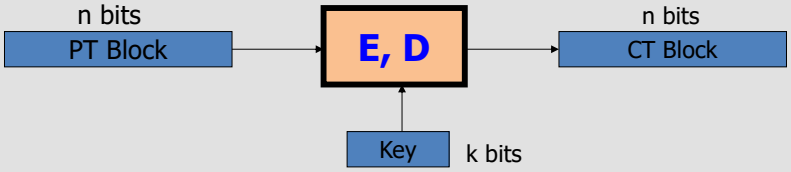
Previously in CSE 433 – Classic and Stream Cipher

- Caesar Cipher, Substitution Cipher
- Frequency attack
- Rotor Machine, Egnima Machine
- XOR with uniform random variable
- Perfect Secrecy - One time pad
- Attack on Stream Cipher
  - Two time pad
  - Integrity attack

Quiz Questions

- Cryptography is all about the algorithm, therefore, as long as we use the right cryptographic tool, and random keys, the system is secure.
- What is the key space of substitution cipher for English alphabet, how would you crack this?
- What is the intuitive idea / formal definition of Shanon's idea of perfect secrecy?
- What is OTP, what's its limitation, and what mathematical property gives it perfect secrecy intuitive?
- What is the definition of semantic security? Why is it a weaker notion of perfect secrecy?
- Can stream cipher have perfect secrecy?
- Name two attacks against stream cipher? What can an adversary achieve with these two attacks?

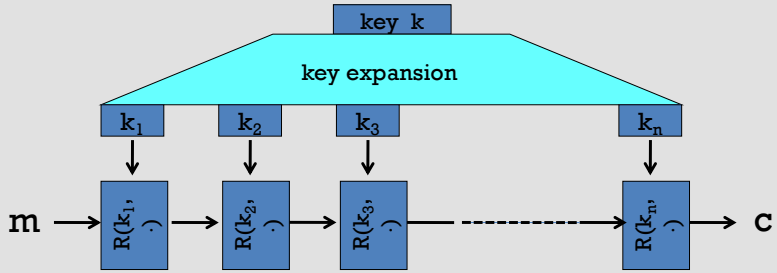
Block ciphers: crypto work horse



Canonical examples:

1. 3DES: n= 64 bits, k = 168 bits
2. AES: n=128 bits, k = 128, 192, 256 bits

Block Ciphers Built by Iteration



R(k,m) is called a round function

for 3DES (n=48), for AES-128 (n=10)

When and how do we use block ciphers? And how to use it correctly.

Performance: Crypto++ 5.6.0 [Wei Dai]

AMD Opteron, 2.2 GHz (Linux)

	Cipher	Block/key size	Speed (MB/sec)
stream	RC4		126
	Salsa20/12		643
	Sosemanuk		727
block	3DES	64/168	13
	AES-128	128/128	109

Pseudo Random Function

Pseudo Random Function (PRF) defined over (K,X,Y):

$$F: K \times X \rightarrow Y$$

such that exists “efficient” algorithm to evaluate F(k,x)

Pseudo Random Permutation

- Pseudo Random Permutation (**PRP**) defined over (K,X):

  $E: K \times X \rightarrow X$

  such that:

  1. Exists "efficient" deterministic algorithm to evaluate E(k,x)
  2. The function E(k, ·) is one-to-one
  3. Exists "efficient" inversion algorithm D(k,y)

Running example

- Example PRPs: 3DES, AES, …

  AES: $K \times X \rightarrow X$ where $K = X = \{0,1\}^{128}$

  3DES: $K \times X \rightarrow X$ where $X = \{0,1\}^{64}$, $K = \{0,1\}^{168}$

- Functionally, any PRP is also a PRF.
  - A PRP is a PRF where X=Y and is efficiently invertible.

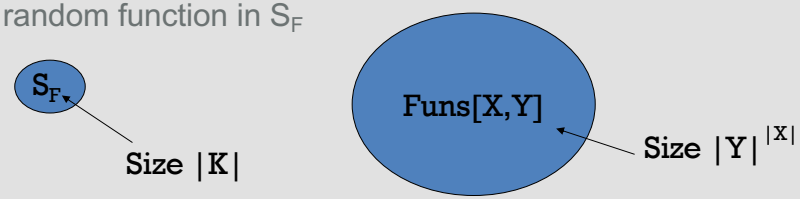
Secure PRFs – Why block ciphers are secure

- Let $F: K \times X \rightarrow Y$ be a PRF

  Funs[X,Y]: the set of **all** functions from X to Y

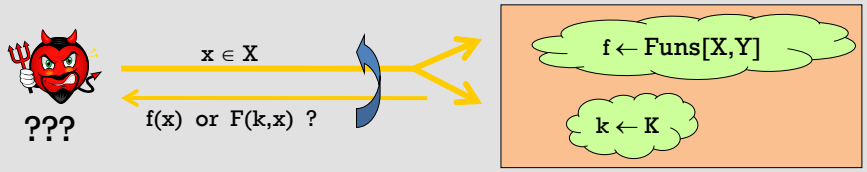
  $S_F = \{ F(k,\cdot) \text{ s.t. } k \in K \} \subseteq$ Funs[X,Y]

- Intuition: a PRF is **secure** if a random function in Funs[X,Y] is indistinguishable from a random function in S_F



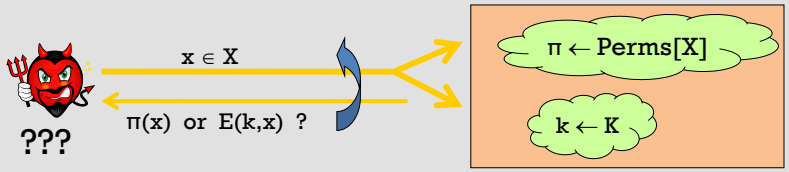
Secure PRFs

- Intuition: a PRF is secure if a random function in Funs[X,Y] is indistinguishable from a random function in SF



Secure PRPs - secure block cipher

- Intuition: a PRP is secure if a random function in Perms[X] is indistinguishable from a random function in SF



Let $F: K \times X \rightarrow \{0,1\}^{128}$ be a secure PRF.

Is the following G a secure PRF?

$$G(k, x) = \begin{cases} 0^{128} & \text{if } x=0 \\ F(k,x) & \text{otherwise} \end{cases}$$

No, it is easy to distinguish G from a random function

Yes, an attack on G would also break F

It depends on F

An easy application: PRF ⇒ PRG

Let $F: K \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a secure PRF.

Then the following $G: K \rightarrow \{0,1\}^{nt}$ is a secure PRG:

$$G(k) = F(k,0) \,||\, F(k,1) \,||\, \cdots \,||\, F(k,t-1)$$

Key property: parallelizable

Security from PRF property: $F(k, \cdot)$ indistinguishable from random function $f(\cdot)$, therefore the output G(k) is also indistinguishable as well

Block Cipher – Design Principles

General Design Principles

- Confusion - Relationship between plain & cipher text is obscured.
  - Ex. Substitution table
- Diffusion – the influence of each plaintext bit is spread over many cipher text (avalanche effect)
  - Ex permutation, let's see a brief example
- Non-linearity – resistant to linear cryptanalysis
- Combine into multiple rounds

The Data Encryption Standard (DES)

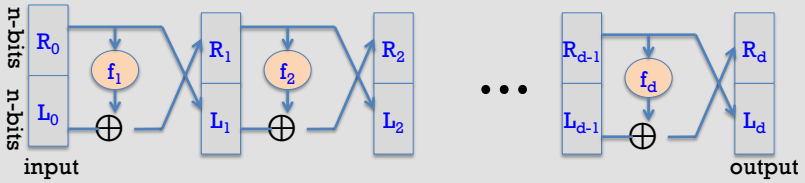
- Early 1970s: Horst Feistel designs Lucifer at IBM
  key-len = 128 bits ; block-len = 128 bits
- 1973: NBS asks for block cipher proposals.
  IBM submits variant of Lucifer.
- 1976: NBS adopts DES as a federal standard
  key-len = 56 bits ; block-len = 64 bits
- 1997: DES broken by exhaustive search
- 2000: NIST adopts Rijndael as AES to replace DES

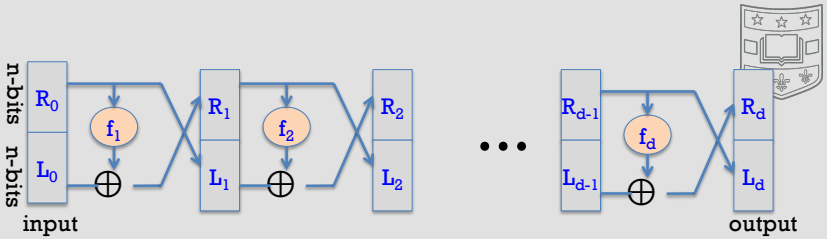
Widely deployed in banking (ACH) and commerce

DES: core idea – Feistel Network

Given functions $f_1, \ldots, f_d$: $\{0,1\}^n \longrightarrow \{0,1\}^n$

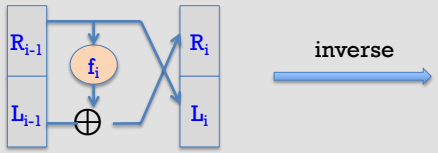
Goal: build invertible function $F: \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$





Claim: for all $f_1, \ldots, f_d$: $\{0,1\}^n \longrightarrow \{0,1\}^n$
Feistel network $F: \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$ is invertible

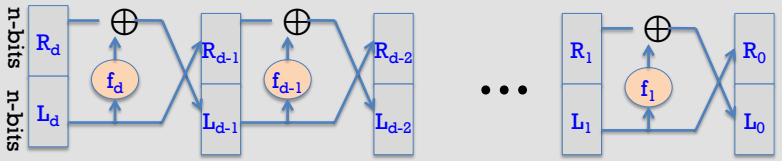
Proof: construct inverse



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

$$L_{i-1} = f_i(L_i) \oplus R_i$$

Decryption circuit

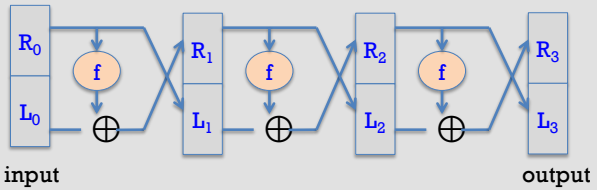


- Inversion is basically the same circuit, with $f_1, \ldots, f_d$ applied in reverse order
- General method for building invertible functions (block ciphers) from arbitrary functions.
- Used in many block ciphers … but not AES

"Thm:" (Luby-Rackoff '85): Convert PRF to PRP

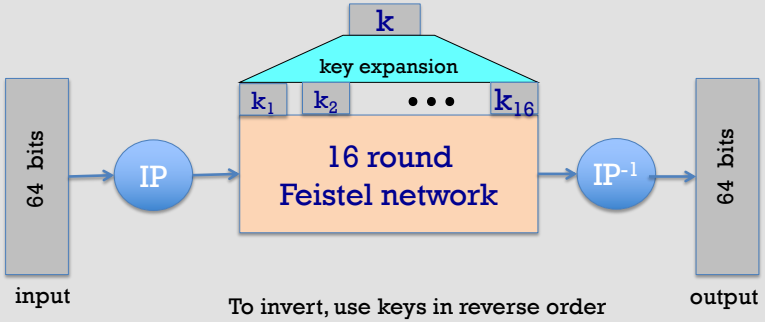
f: $K \times \{0,1\}^n \longrightarrow \{0,1\}^n$ a secure PRF

$\Rightarrow$ 3-round Feistel F: $K^3 \times \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$ a secure PRP

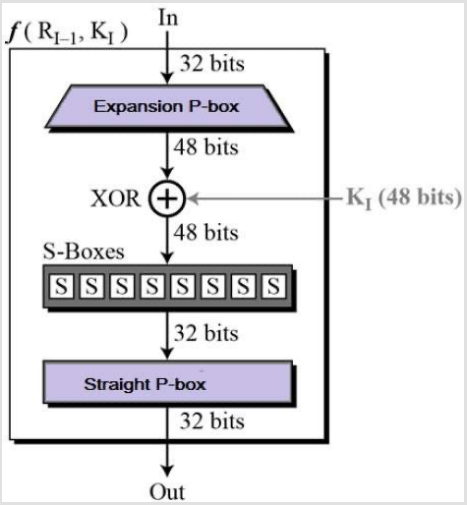


DES: 16 round Feistel network

$f_1, \ldots, f_{16}$: $\{0,1\}^{32} \longrightarrow \{0,1\}^{32}$, $f_i(x) = \mathbf{F}(k_i, x)$



The function $F(k_i, x)$



The S-boxes

$S_i: \{0,1\}^6 \longrightarrow \{0,1\}^4$

S_5		Middle 4 bits of input															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Example: a bad S-box choice

Suppose:

$S_i(x_1, x_2, \ldots, x_6) = (x_2 \oplus x_3,\ x_1 \oplus x_4 \oplus x_5,\ x_1 \oplus x_6,\ x_2 \oplus x_3 \oplus x_6)$

or written equivalently: $S_i(\mathbf{x}) = A_i \cdot \mathbf{x} \pmod 2$

$$\begin{bmatrix} 0&1&1&0&0&0 \\ 1&0&0&1&1&0 \\ 1&0&0&0&0&1 \\ 0&1&1&0&0&1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} x_2 \oplus x_3 \\ x_1 \oplus x_4 \oplus x_5 \\ x_1 \oplus x_6 \\ x_2 \oplus x_3 \oplus x_6 \end{bmatrix}$$

We say that S_i is a linear function.

Choosing the S-boxes and P-box

Choosing the S-boxes and P-box at random would result in an insecure block cipher (key recovery after $\approx 2^{24}$ outputs) [BS'89]

Several rules used in choice of S and P boxes:

- **No output bit should be close to a linear func. of the input bits**

⋮

DES challenge

msg = "The unknown messages is: XXXX … "

CT = c_1 c_2 c_3 c_4

Goal: find $k \in \{0,1\}^{56}$ s.t. $DES(k, m_i) = c_i$ for i=1,2,3

1997: Internet search -- **3 months**
1998: EFF machine (deep crack) -- **3 days** (250K $)
1999: combined search -- **22 hours**
2006: COPACOBANA (120 FPGAs) -- **7 days** (10K $)

⇒ 56-bit ciphers should not be used !!

Strengthening DES against ex. search

Method 1: **Triple-DES**

- Let $E : K \times M \longrightarrow M$ be a block cipher
- Define **3E**: $K^3 \times M \longrightarrow M$ as

$$\mathbf{3E}((k_1,k_2,k_3), m) = E(k_1, D(k_2, E(k_3, m)))$$

For 3DES: key-size = 3×56 = 168 bits. 3×slower than DES.

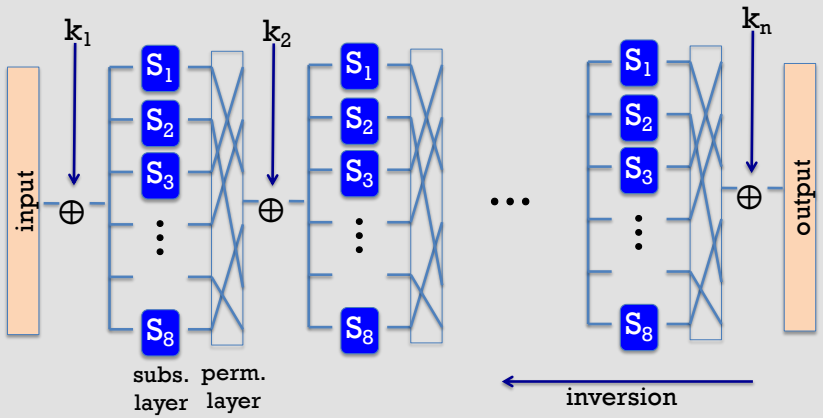
(simple attack in time $\approx 2^{118}$)

The AES process

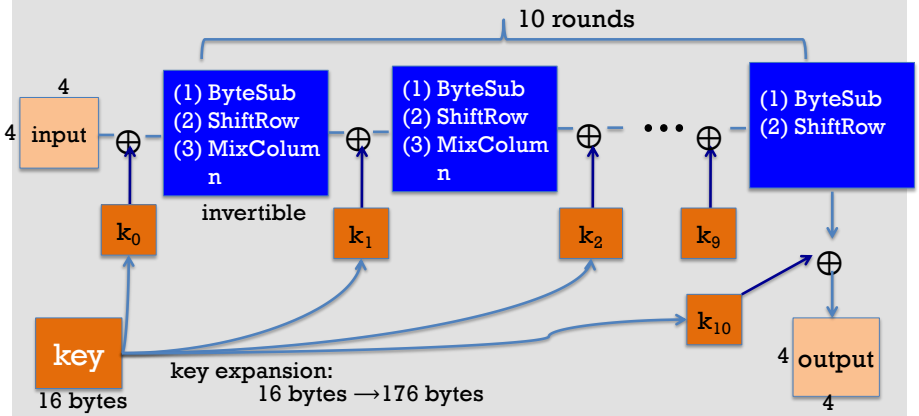
- 1997: NIST publishes request for proposal
- 1998: 15 submissions. Five claimed attacks.
- 1999: NIST chooses 5 finalists
- 2000: NIST chooses Rijndael as AES (designed in Belgium)

Key sizes: 128, 192, 256 bits. Block size: 128 bits

AES is a Subs-Perm network (not Feistel)

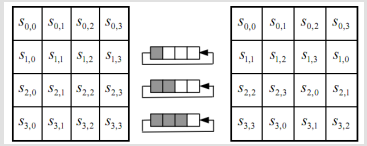


AES-128 schematic

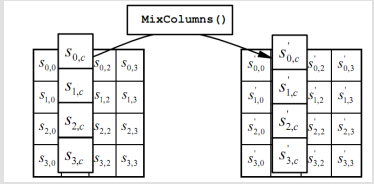


The round function

- **ByteSub**: a 1 byte S-box. 256 byte table [4 x 4 table], input is used as index
- **ShiftRows**:



- **MixColumns**:



Code size/performance tradeoff

	Code size	Performance
Pre-compute round functions (24KB or 4KB)	largest	fastest: table lookups and xors
Pre-compute S-box only (256 bytes)	smaller	slower
No pre-computation	smallest	slowest

AES in hardware

AES instructions in Intel Westmere:

- **aesenc, aesenclast**: do one round of AES
  128-bit registers: xmm1=state, xmm2=round key
  aesenc xmm1, xmm2 ; puts result in xmm1
- **aeskeygenassist**: performs AES key expansion
- Claim 14 x speed-up over OpenSSL on same hardware

Similar instructions on AMD Bulldozer

Attacks

Best key recovery attack:
four times better than ex. search [BKR'11]

Related key attack on AES-256: [BK'09]
Given 2^{99} inp/out pairs from **four related keys** in AES-256, can recover keys in time $\approx 2^{99}$

Also many side channel attacks

Summary

- Design principles of block cipher
  - Confusion
  - Diffusion
  - Non-linear
  - Multiple rounds
- Luby-Rackoff, Feistel Network generating PRP from PRF
- Overview of DES, AES
- Attacks
  - Exhaustive search, meet in the middle, linear and differential cryptanalysis, side channel attack

Block Cipher - Modes of Operation

How do we encrypt very long messages, using AES and DES?

Let's forget about how DES and AES work, but think about a block cipher as any generic PRP

Using PRPs and PRFs

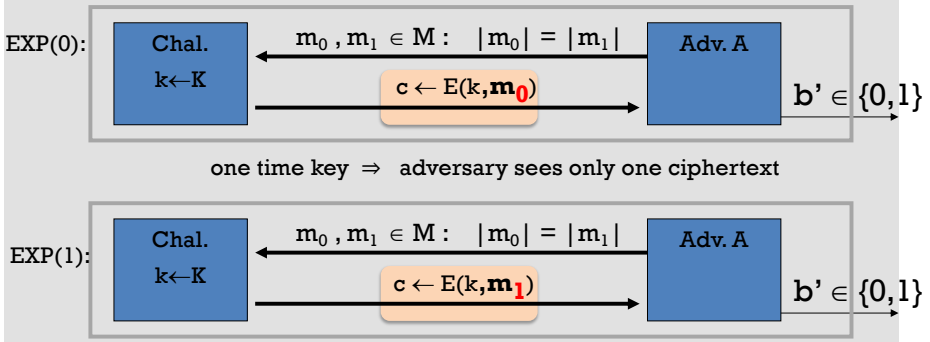
Goal: build "secure" encryption from a secure PRP (e.g. AES).

This segment: **one-time keys**

1. Adversary's power:
   Adv sees only one ciphertext (one-time key)
2. Adversary's goal:
   Learn info about PT from CT (semantic security)

There is also many-time keys (a.k.a chosen-plaintext security)

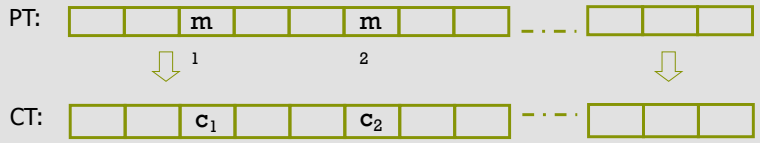
Semantic Security Model for one time key (one-time key)



$Adv_{SS}[A,OTP] = \left| Pr[\textbf{EXP(0)}=1] - Pr[\textbf{EXP(1)}=1] \right|$ should be "neg."

Incorrect use of a PRP

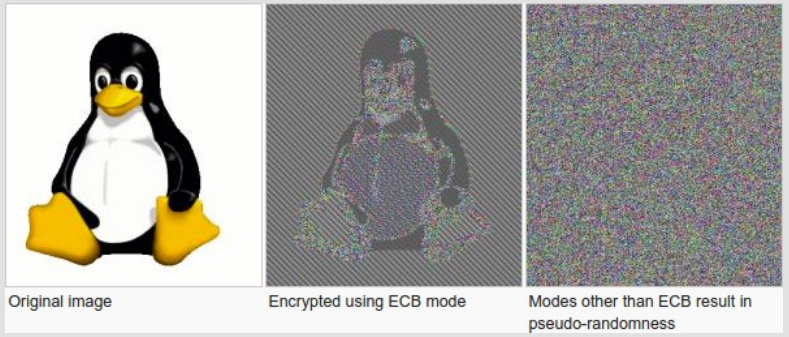
Electronic Code Book (ECB):



Problem:

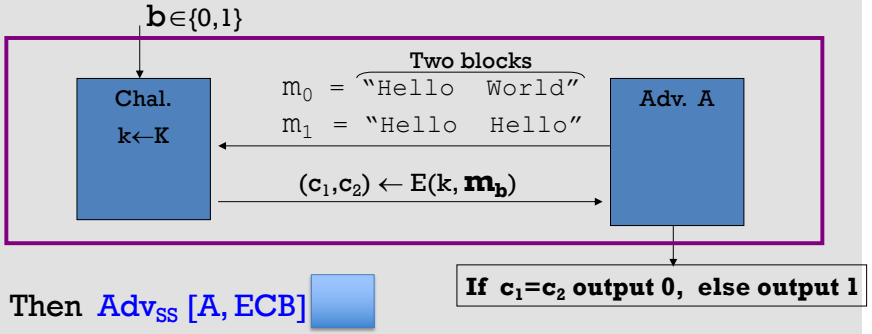
– if $m_1=m_2$ then $c_1=c_2$

Is it secure ?



ECB is not Semantically Secure

ECB is not semantically secure for messages that contain more than one block.



Then $Adv_{SS}[A, ECB]$ ▭

Semantic Security for many-time key

Key used more than once

⇒ adv. sees many CTs with same key

Adversary's power: chosen-plaintext attack (CPA)

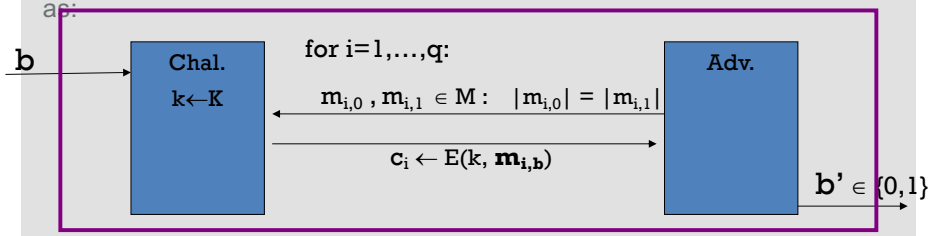
- Can obtain the encryption of arbitrary messages of his choice (conservative modeling of real life)

Adversary's goal: Break sematic security

Semantic Security for many-time key

(CPA security)

$E = (E,D)$ a cipher defined over (K,M,C). For b=0,1 define EXP(b) as:



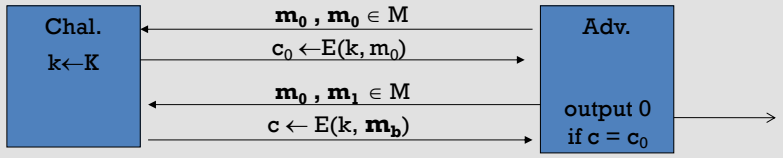
if adv. wants $c = E(k, m)$ it queries with $m_{j,0} = m_{j,1} = m$

Def: E is sem. sec. under CPA if for all "efficient" A:

$Adv_{CPA}[A,E] = |Pr[EXP(0)=1] - Pr[EXP(1)=1]|$ is "negligible."

Ciphers insecure under CPA

Suppose E(k,m) always outputs same ciphertext for msg m. Then:

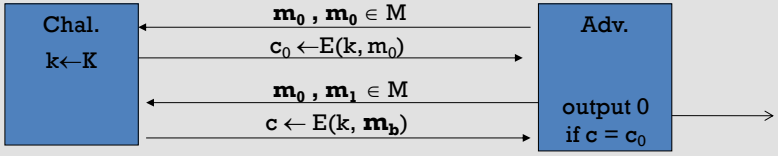


So what? For example, **An attacker can learn that two encrypted files are the same, two encrypted packets are the same, etc.**

- Leads to significant attacks when message space M is small

Ciphers insecure under CPA

Suppose E(k,m) always outputs same ciphertext for msg m. Then:

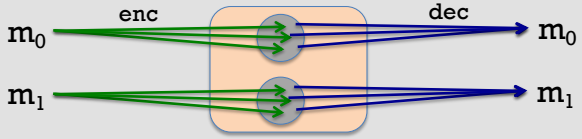


If secret key is to be used multiple times ⇒

given the same plaintext message twice, encryption must produce different outputs.

Solution 1: randomized encryption

- E(k,m) is a randomized algorithm:

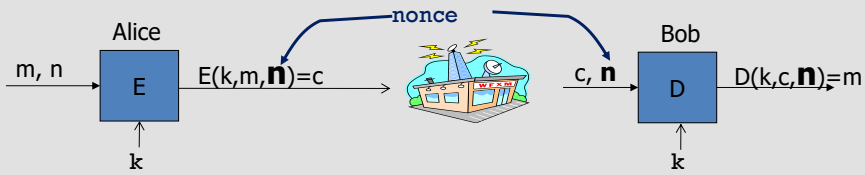


⇒ encrypting same msg twice gives different ciphertexts (whp)

⇒ ciphertext must be longer than plaintext

Roughly speaking: CT-size = PT-size + "# random bits"

Solution 2: nonce-based Encryption

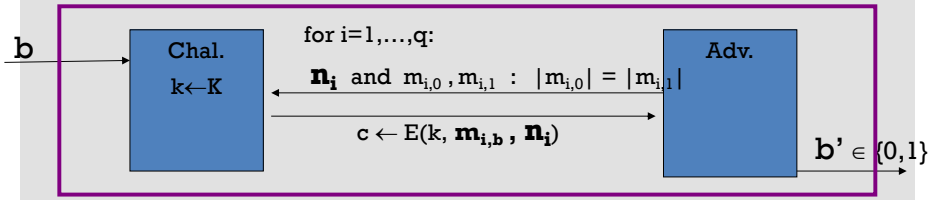


- nonce n: a value that changes from msg to msg.
  (k,n) pair never used more than once
- method 1: nonce is a **counter** (e.g. packet counter)
  - used when encryptor keeps state from msg to msg
  - if decryptor has same state, need not send nonce with CT
- method 2: encryptor chooses a **random nonce**, $n \leftarrow \mathcal{N}$

  Essentially it is randomized encryption

CPA security for nonce-based encryption

System should be secure when nonces are chosen adversarially.



All nonces $\{n_1, \ldots, n_q\}$ must be distinct.

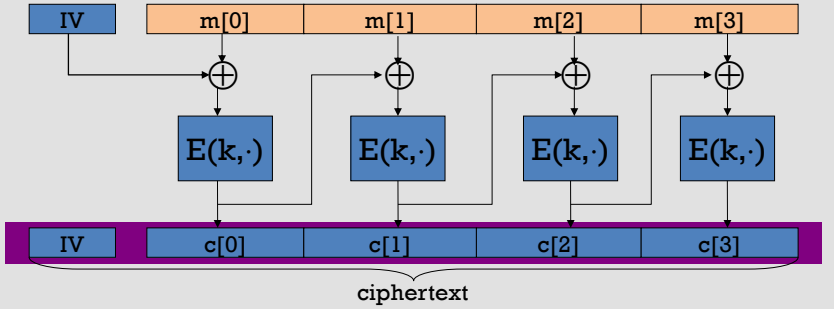
Def: nonce-based $\mathbb{E}$ is sem. sec. under CPA if for all "efficient" A:

$Adv_{nCPA}[A,\mathbb{E}] = |Pr[EXP(0)=1] - Pr[EXP(1)=1]|$ is "negligible."

Take another step back, and think about how to implement the ideas we talk about, nonce based encryption or randomized encryption

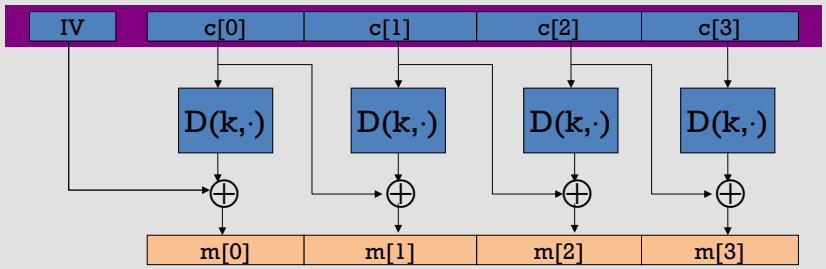
Construction 1: CBC with random IV

Let (E,D) be a PRP. $E_{CBC}(k,m)$: choose **random** IV∈X and do:



Decryption circuit

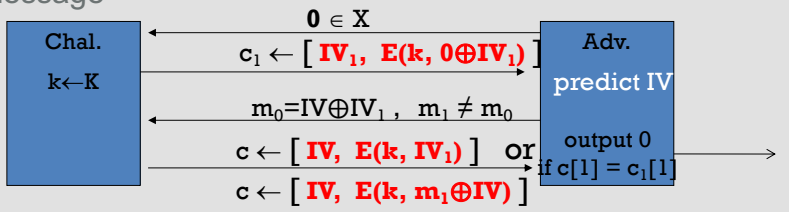
In symbols: $c[0] = E(k, IV \oplus m[0]) \Rightarrow m[0] = D(k, c[0]) \oplus IV$



Warning: an attack on CBC with rand. IV

CBC where attacker can predict the IV is not CPA-secure !!

Suppose given $c \leftarrow E_{CBC}(k,m)$ can predict IV for next message



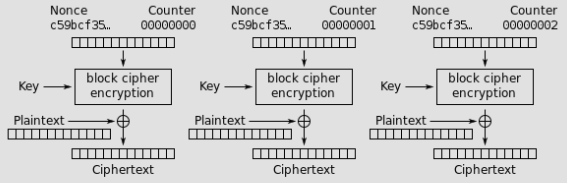
Bug in SSL/TLS 1.0: IV for record #i is last CT block of record #(i-1)

An example Crypto API (OpenSSL)

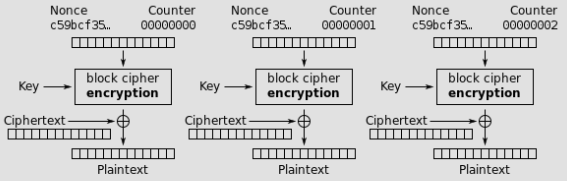
```
void AES_cbc_encrypt(
    const unsigned char *in,
    unsigned char *out,
    size_t length,
    const AES_KEY *key,
    unsigned char *ivec,      ← user supplies IV
    AES_ENCRYPT or AES_DECRYPT);
```

When nonce is non random need to encrypt it before use

Couter Mode Summary



Counter (CTR) mode encryption



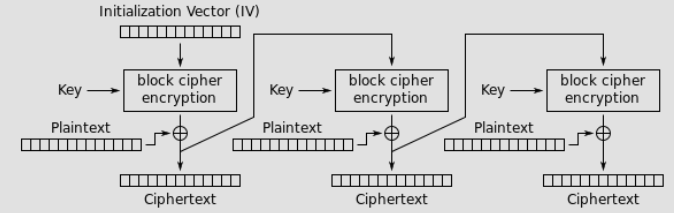
Counter (CTR) mode decryption

Comparison: CTR vs. CBC

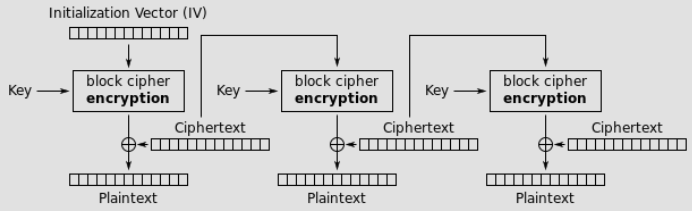
	CBC	CTR
uses	PRP	PRF
parallel processing	No	Yes
Security of rand. enc.	q^2 L^2 << |X|	q^2 L << |X|
dummy padding block	Yes	No
1 byte msgs (nonce-based)	16x expansion	no expansion

(for CBC, dummy padding block can be solved using ciphertext stealing)

CFB Mode

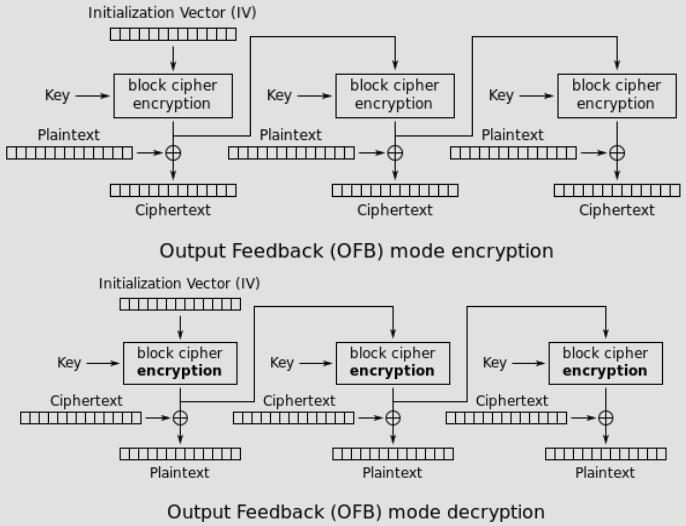


Cipher Feedback (CFB) mode encryption



Cipher Feedback (CFB) mode decryption

OFB



Summary

- Design principles of block cipher
- Techniques to prove semantic security of block cipher
- Luby-Rackoff, Feistel Network generating PRP from PRF
- One time key vs Many time key
- Ciphertext only attack vs Chosen-plaintext attack
- Randomized encryption vs counter-based encryption
- Modes of operations – CBC, CTR, OFB, CFB
- Padding, Predictable IV
- For many time keys, randomized CBC or CTR is semantically secure