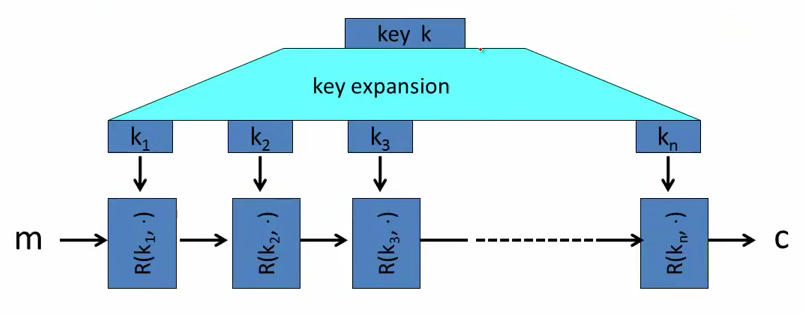
## Block ciphers

Takes a Plain Text (PT) of n bits and encrypts with a Key of k bits generating a Ciphered Text (CT) of n bits

3DES: n = 64 bits k = 168 bits

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

Performance of block cipher is worse than stream ciphers

## Pseudo Random Function (PRF) and Pseudo Random Permutation (PRP)

Pseudo Random Function (PRF)

PSF defined over (K,X,Y): F: K x X -> Y

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

Pseudo Random Permutation (PRP)

PRP defined over (K,X): E: K x X -> 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)

Real examples:

AES: K x X -> X where K=X={0,1}128

3DES: K x X -> X where X={0,1}64 K={0,1}168

Any PRP is also a PRF.

A PRP is a PRF where X=Y and is efficiently invertible.

## Secure PRFs

Let F: K x X -> Y be a PRF

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

SF = { F(k,·) s.t. k in K } contained in Funs[X,Y]

Size of Funs[X,Y] = | Y ||X|

Size of SF = | K |

A PRF is secure if

- a random function is Funs[X,Y] is indistinguishable from a random function in SF

## Application PRF -> PRG

Let F: K x {0,1}n -> {0,1}n be a secure PRF

The the following G: K -> {0,1}nt is a secure PRG:

G(k) = F(k,0) || F(k,1) || … || F(k,t)

Key property: parallelizable. i.e., the process can be split into t blocks and each block can be computed by a different processor/core.

# The Data Encryption Standard (DES)

Early 1970s: Horst Feistel designs Lucifer at IBM

key-len = 128 bits block = 128 bits

1973: NBS asks for block cipher proposals.

IBM submits variant of Lucipher

1976: NBS adopts DES as a federal standard

key-len = 56 bits block-len = 64 bits

1997: DES broken by exhaustive search

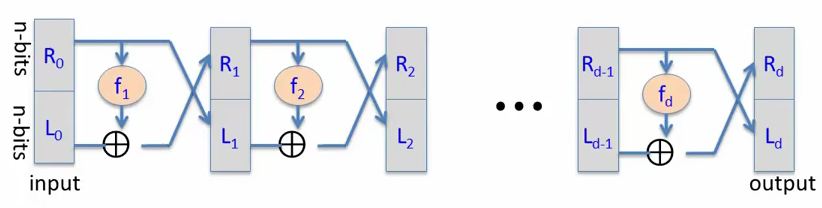
200: NIST adopts Rijndael as AES to replace DES

Widely deployed in banking (ACH) and commerce

## DES: Core idea - Fesitel Network

Given functions f1, …, fd: {0,1}n -> {0,1}n

Goal: build invertible function F: {0,1}2n -> {0,1}2n



Ri = Fi(Ri-1) xor Li

Li = Ri-1

with i = 1,....,d

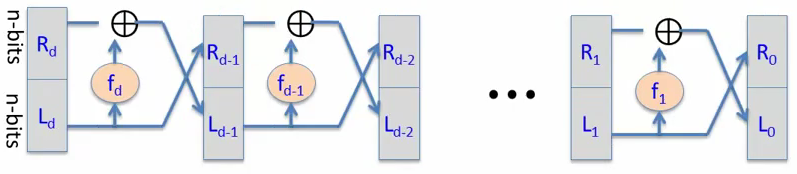
For all f1,...,fd: {0.1}n → {0,1}n the Feistel network F: {0,1}2n → {0,1}2n is invertible

The inversion of the round i+1 is:

Ri = Li+1

Li = Fi+1(Li+1) xor Ri+1

Inversion is basically the same circuit with f1,...,fd applied in reverse order:



General method for building invertible functions (block ciphers) from arbitrary functions

Used in many block ciphers, but not AES

### Theorem (Luby-Rackoff’85)

Let f: K x {0,1}n → {0,1}n a secure PRF

then a 3-round Feistel F: K3 x {0,1}2n → {0,1}2n is a secure PRP

1st round: f(k0, R0)

2nd round: f(k1, R1)

3rd round: f(k2, R2)

Three independent keys are needed (remember the key expansion) in a block cipher

## DES

DES is basically a 16 round Feistel network

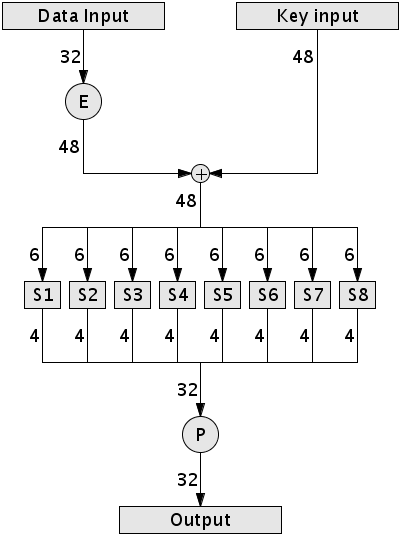
f1,...,f16: {0,1}32 → {0,1}32 , fi(x) = F(ki,x)

2n = 64 bits

ki is the round key and is derived from the key k

There is an initial permutation before the Feistel network and de inverse bit permutation at the output of the Feistel network.

A diagram of the round functions:



The boxes S are a map from 6 bits to 4 bits implemented as a look-up table.

# Exhaustive Search Attacks

Exhaustive Search for block cipher key

Goal: given a few input output pairs (mi, ci=E(k,mi) ) i=1,...,3 find key k

The key k will be unique with a probability aprox 99.5% = 1 - 1/256

For two DES pairs (m1, c1=DES(k,m1)), (m2, c2=DES(k,m2)) unicity probability aprox = 1 - 1/271

For AES-128: given two inp/out pairs, unicity prob. aprox = 1 - 1/2128

This implies that two input/output pairs are enough for exhaustive key search.

DES challenge

msg “The unk | own mess | ages is:| XXXX….”

CT = c1 c2 c3 c4

Goal: find k in {0,1}56 such that DES(k, mi) = ci 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 must not be used

### Strengthening DES against exhaustive search

Method 1: Triple-DES

Let E : K x M → M be a block cipher

Define 3E : K3 x M → M as

3E ( (k1,k2,k3), m) = E(K1, D(k2, E(k3,m) ) )

For3DES: key-size 3 x 56 = 168 bits

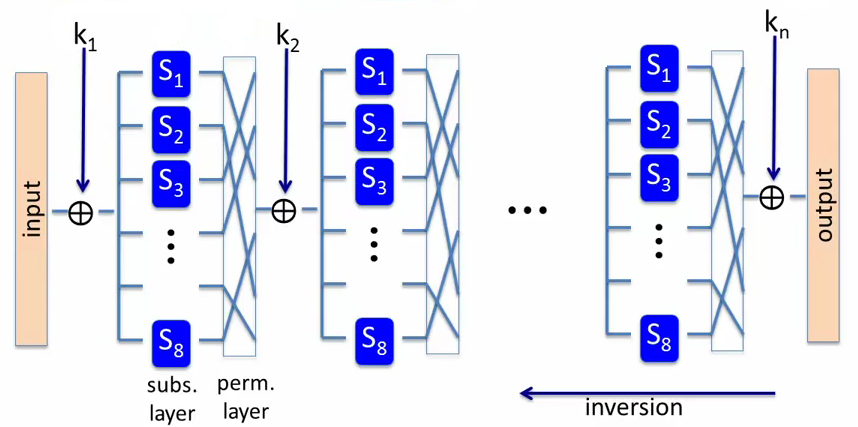
simple attack in time aprox 2118 > 290

Why not double DES?

# The AES block cipher

It is based on a Substitution-Permutation network

Each step must be reversible



# Block ciphers from PRGs

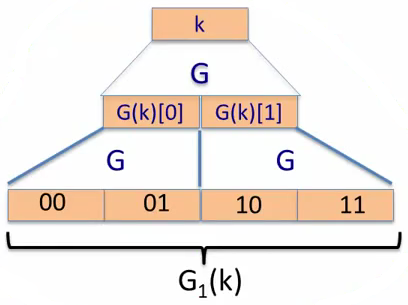
Let G : K → K2 be a secure PRG

Define 1-bit PRF F: K x {0,1} → K as

F (k, x in{0.1}) = G(k)[x]

Theorem: If G is a secure PRG then F is a secure PRF

### Extend a PRF to a larger domain

define G1: K→ K4 as G1(k) = G(G(k[0]) || G(G(k)[1])

We get a 2-bit PRF:

F(k, x in{0,1}2) = G1(k)[x]

This PRG G1 is semantically secure. At each level the result is indistinguishable from random, and each part is indistinguishable from random too.

Let G: K→ K2

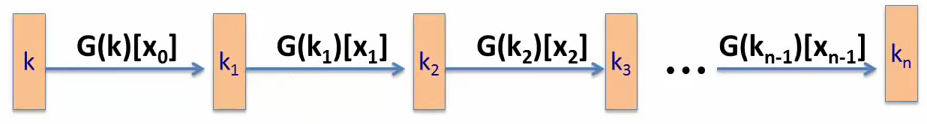
define G2: K→ K8 as G2(k) = G1(G1(k)[0])) || G1(G1(k)[1])

We get a 3-bit PRF. There is no need to compute all the output in the domain K8, we can go computing only the half that is relevant to us at each level.

### Extending even more: The GGM PRF

Let G: K→ K2, define PRF F: K x {0,1}n→ K as

For imput x=x0 x1...xn-1 in {0,1}n do:



As before, G a secure PRG ⇒ F is a secure PRF on {0,1}n

It is not used in practice due to slow performance.

Even more, a secure PRP can be built from a secure PRG just takin this GGM PRF and plug it into the Luby-Rackoff theorem (3 Feistel Network)

Lemma:

Any secure PRP is also a secure PRF if |X| is sufficiently large

Ley E be a PRP over (K,X)

Then for any q-query adversary A:

| AdvPRF[A,E] - AdvPRP[A,E] | < q2 / 2|X|

When |X| is sufficiently large the right value becomes negligible

# Modes of operation: One Time Key

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

One-time keys

Adversary sees only one ciphertext (one-time key)

Adversary’s goal is to learn info about PT from CT (semantic security)

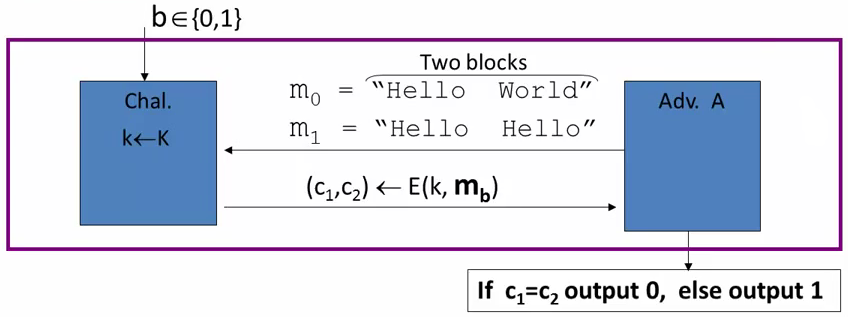
### Incorrect use of a PRP

Electronic Code Book (**ECB**)

The plain message is split in many block and each block is ciphered independently with the same key.

If two blocks are equal then the ciphered block must be equal too → Info for the attacker

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

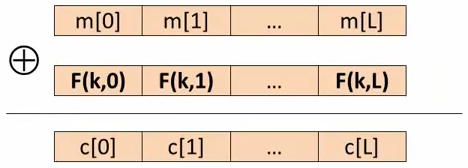


The Advss[A,ECB] = 1

## Secure construction I

Deterministic counter mode from a PRF F

EDETCTR (k,m) ⇒ Stream cipher built from a PRF (e.g. AES, 3DES)



Theorem:

For any L>0

If F is a secure PRF over (K,X,X) then EDETCTR is semantically secure cipher over (K,XL,XL).

In particular, for any efficient adversary A attacking EDETCTR there exists a n effective PRF adversary B s.t.:

AdvSS[A,EDETCTR] = 2 \* AdvPRF [B,F]

AdvPRF[B,F] is negligible (since F is a secure PRF)

Hence, Advss[A,EDETCTR] must be negligible

# Security for many-time key

Semantic security for many-time key

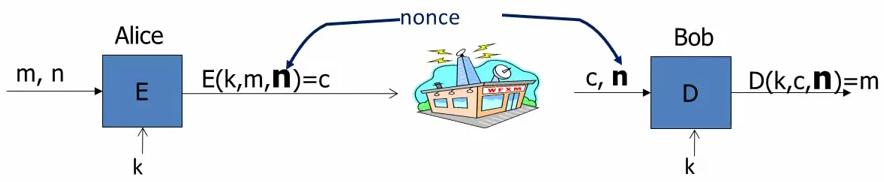
Key used more than once ⇒ Adversary sees many CTs with the same key

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 semantic security

## Solution 2: nonce-based encryption



nonce n: it’s a value that changes from msg to msg. A pair (k,n) is never used more than once

The nonce doesn’t have to be random or unknown to the adversary. The only requirement for the once is that must not be used more than once

Choosing a nonce:

method 1: nonce is a counter (e.g. packet counter)

- used when encryptor keeps state from msg to msg

- if the decryptor has the same state there’s no need to send the nonce with the CT

method 2: encryptor chooses a random nonce, n ← N

# Modes of operation: many time key (CBC)

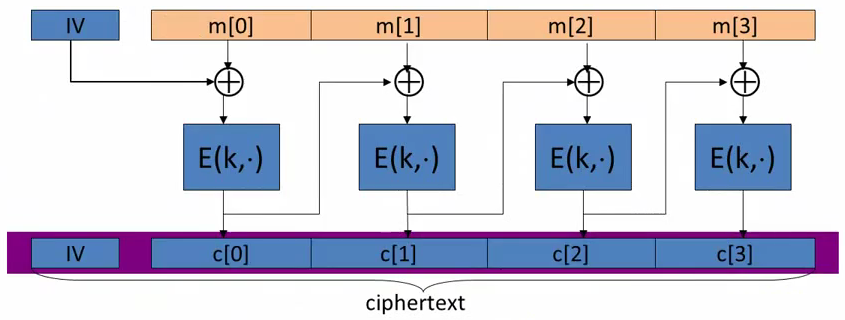
Filesystems: Same AES key used to encrypt many files

IPsec: Same AES key used to encrypt many packets

## Cipher Block Chaining

### Construction 1: CBC with random IV

Let (E,D) be a PRP. EPRP(k,m): choose random IV in X and do:



IV: Initialization Vector

c[0] = E( k, IV xor m[0] ) m[0] = IV xor D(k,c[0])

## CBC: CPA Analysis

CBC Theorem: For any L>0

If E is a secure PRP over (K,X) then ECBC is a semantically secure under CPA over (K, XL, XL+1)

In particular, for a q-query adversary A attacking ECBC there exists a PRP adversary B such that:

AdvCPA[A,ECBC] =< 2\*AdvPRP[B,E] + 2 \* q2 \* L2 / |X|

CBC is only secure as long as q2L2 << |X|

q = number os messages encrypted with k

L = length of max message

Suppose we want AdvCPA[A,ECBC] =< 1/232 ⇐ q2L2/|X| < 1/232

AES: |X| = 2128 ⇒ qL < 248

So, after 248 AES blocks the key must be changed

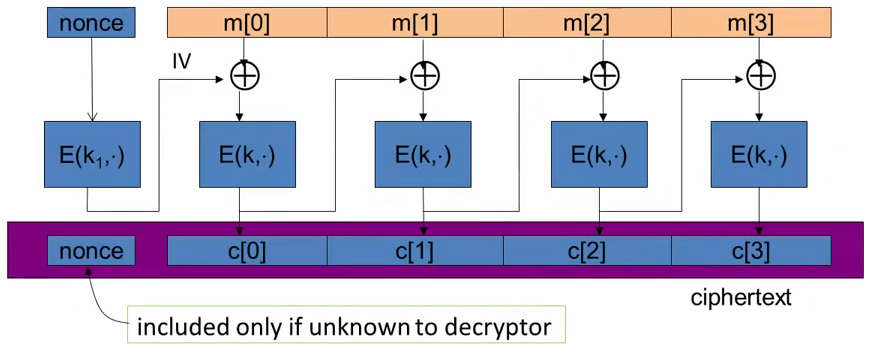
3DES: |X| = 264 ⇒ qL < 216

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

## Construction 1’: nonce-based CBC

Cipher clobk chaining with unique nonce: key = (k,k1)

Unique nonce means: (key, n) pair is used for only one message



The keys k1 and k must be different between them

CBC padding

When the last block is shorter than the size of a block a pad is append to it.

TLS: for n > 0, n byte pas is: n | n | n | … | n

if no pad needed then a dummy block is added 16 | 16 | 16 | 16 | …. | 16

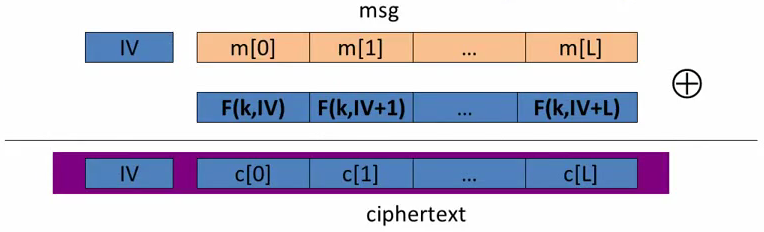
# Modes of operation Many Time Key (CTR)

Filesystems: Same AES key used to encrypt many files

IPsec: Same AES key used to encrypt many packets

## Construction 2: Randomize Counter Mode

Let F be a secure PRF. F: K x {0,1}n → {0,1}n

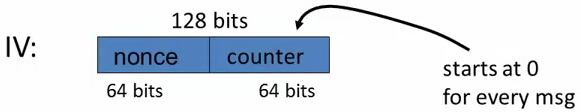


IV is chosen randomly for every message

Parallelizable (unlike CBC)

## Construction 2’: Nonce Counter Mode

To ensure F(k,x) is never used more than once, choose IV as:



## Randomize Counter Mode: CPA analysis

Counter-mode theorem: For any L>0,

If F is a secure PRF over (K,X,X) then ECTR is a semantically secure under CPA over (K,XL,XL+1)

In particular, for a q-query adversary A attacking ECTR there exists a PRF adversary B such that:

AdvCPA[A,ECTR] =< 2 AdvPRF[B,F] + 2 q2 L / |X|

Counter-mode only secure as long as q2L << |X|. Better than CBC!

### Example

Suppose we want AdvCPA[A,ECTR] =< 1 / 232 ⇐ q2L / |X| < 1/232

AES: |X| = 2128 ⇒ qL1/2 < 248

So, after 232 CTs each of len 232, must change key (total of 264 AES blocks)

# Comparison: ctr vs CBC

|  |  |  |
| --- | --- | --- |
|  | CBC | ctr mode |
| uses | PRP | PRF |
| parallel processing | No | Yes |
| Security of rand. enc | q2L2 << |X| | q2L << |X| |
| dummy padding block | Yes | No |
| 1 byte msgs (nonce-based) | 16x expansion | no expansion |

for CBC, dummy padding block can by solved using ciphertext stealing