TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Security and Cryptography 1

Stefan Köpsell, Thorsten Strufe

Module 5: Pseudo Random Permutations and Block Ciphers

Disclaimer: large parts from Mark Manulis and Dan Boneh

Dresden, WS 17/18

Reprise from the last modules

You know ***CIA***, perfect secrecy and semantic security

You know different classes of cryptographic algorithms

You can explain (and show) CTO, KPA, IND-CPA and IND-CCA ***adversary models***

You can prove that the OTP has perfect secrecy

You understand when PRGs are secure, and you can explain stream ciphers

You can explain how semantic security of stream ciphers is proven

Module Outline

Mini function theory refresher

(Trapdoor) One-way functions

Pseudo Random Functions

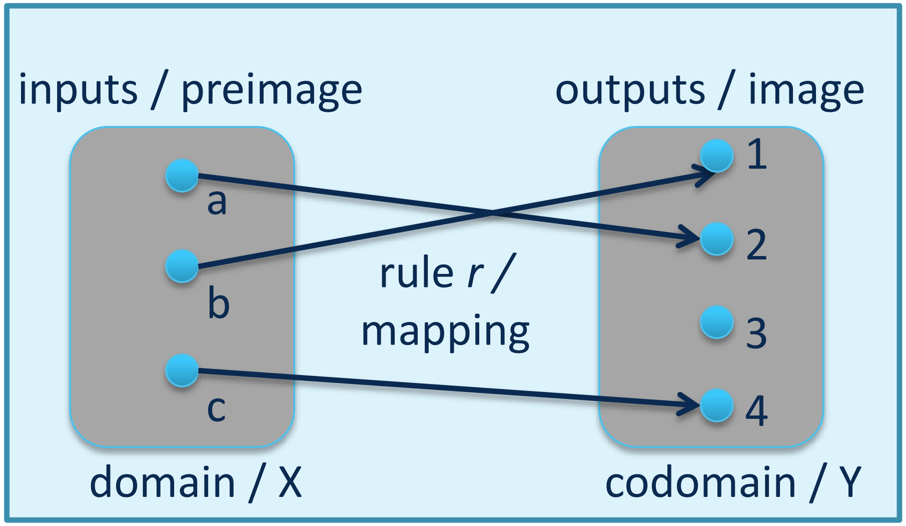
Pseudo Random Permutations

Building PRPs:
Confusion – Diffusion Paradigm / Subst-Perm Networks
Feistel Networks and DES / 3DES
AES

Making it work: Modes of operation

Properties of Functions

A little refresher on functions…



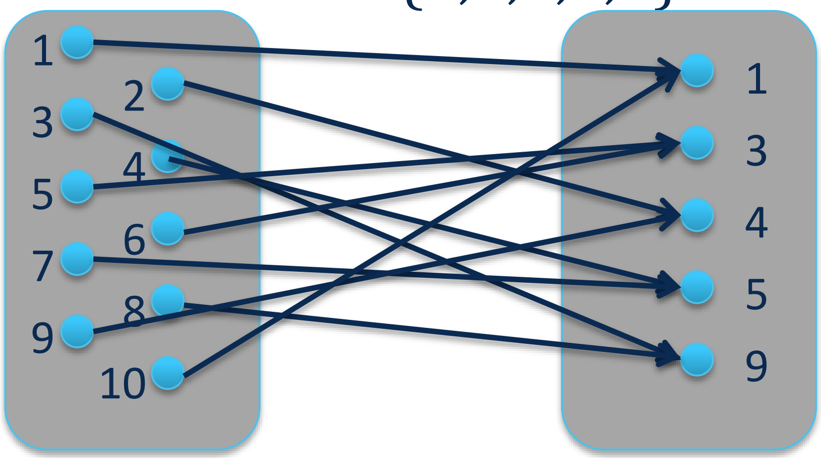
$f: X \longrightarrow Y$ $y=f(x)$

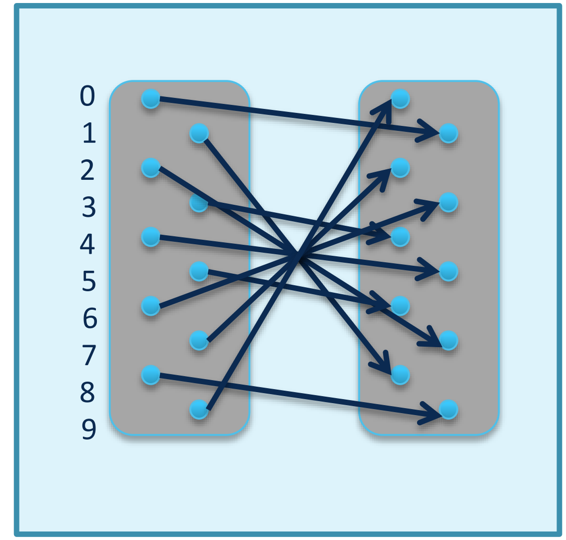
$X = \{a,b,c\}$ $Y=\{1,2,3,4\}$ $Im(f) = \{1,2,4\}$

Functions, Functions, Functions

$X = \{1,2,3,..10\}$ $\qquad f(x) = x^2 \bmod 11$

$f: X \longrightarrow Y$ $\qquad Y = \{1,3,4,5,9\}$





f is called

"onto" (surjective): $Y = Im(f)$ or: $\forall y \in Y \quad \exists x \in X: y = f(x)$

"one-to-one" (injective): $\forall x1, x2 \in X : f(x1) = f(x2) \Rightarrow x1 = x2$

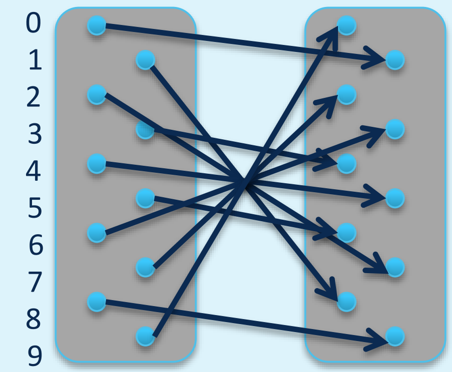
bijection: *f(x) is 1 – 1 and Im(f) = Y*

For *bijection f* there is an *inverse:* $g = f^{-1}$: $g(y) = x \;\; (= f(g(x))\,)$

Hint: (Trapdoor) One-way Functions

$X = \{0,\dots,9\}$, $Y = \{0,\dots,9\}$

$f(x) = (x+1)^3 \bmod 10$



Finding the inverse f^{-1} is not always „easy“

One way functions:

A function $f\colon X \longrightarrow Y$ is called a *one-way-function*, if $f(x)$ is „easy“ to compute for all $x \in X$, but for “essentially all” elements $y \in \mathrm{Im}(f)$ it is computationally hard to find the preimage x.

Trapdoor one-way functions:

A *trapdoor one-way function* is a one-way function that, given some additional *trapdoor information,* is feasible to invert.

(Pseudo Random) Permutations, Involutions

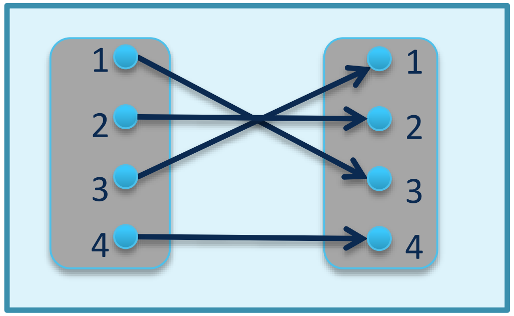
Permutations and Involutions:

A *permutation* π is a *bijective function* from a domain to itself:

$$\pi: X \longrightarrow X \qquad\qquad \mathrm{Im}(f) = X$$

A permutation π with: $\pi = \pi^{-1}$ (or: $\pi(\pi(x)) = x$)

is called an *involution*.



Pseudo Random Functions (PRF):

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

on „domain“ X and „range“ K, with „efficient“ algorithm to evaluate F(k,x)

Pseudo Random Permutation (PRP):

Permutation $\quad E: K \times X \longrightarrow X$

has efficient deterministic algorithm to evaluate $E(k,x)$ ***and***

efficient inversion algorithm $D = E^{-1}$

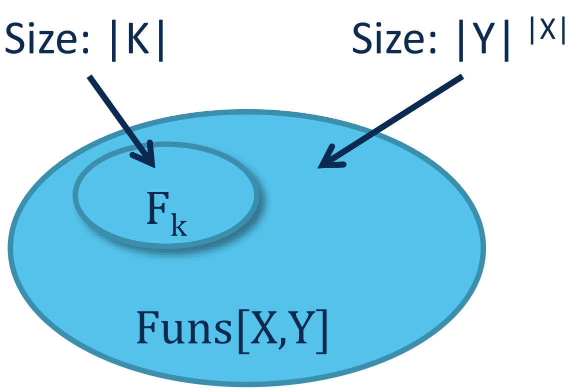
Security of PRFs

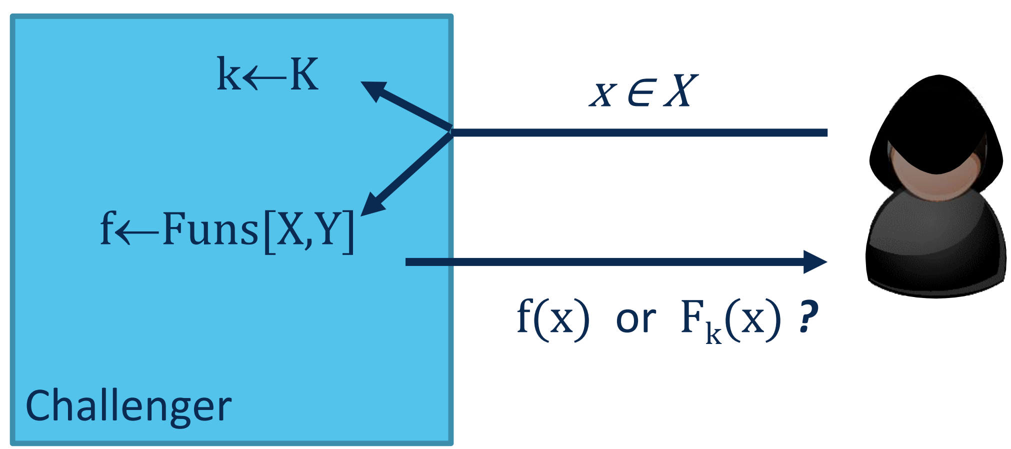
A PRF is secure, if it is indistinguishable from a random function:

Consider

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

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





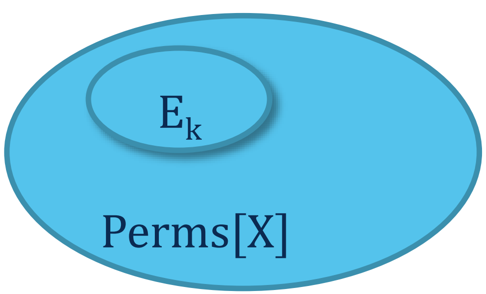
Security of PRPs

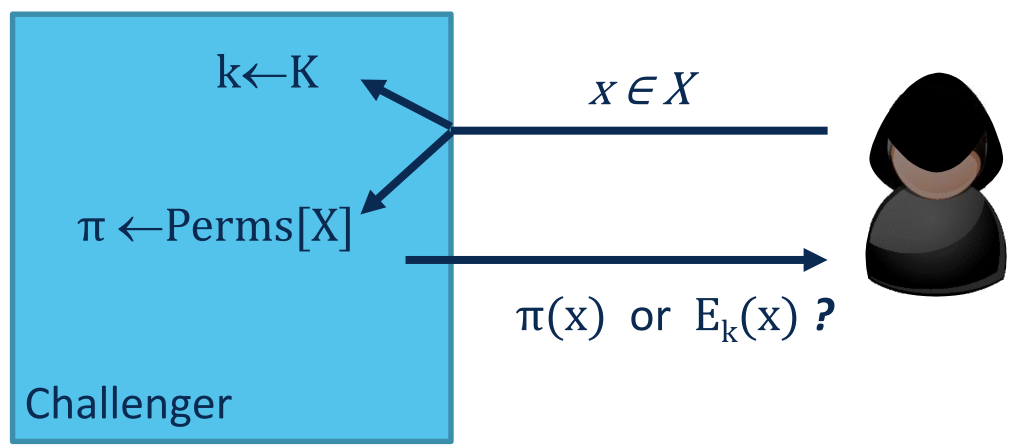
A PRP is secure, if it is indistinguishable from a random permutation:

Consider

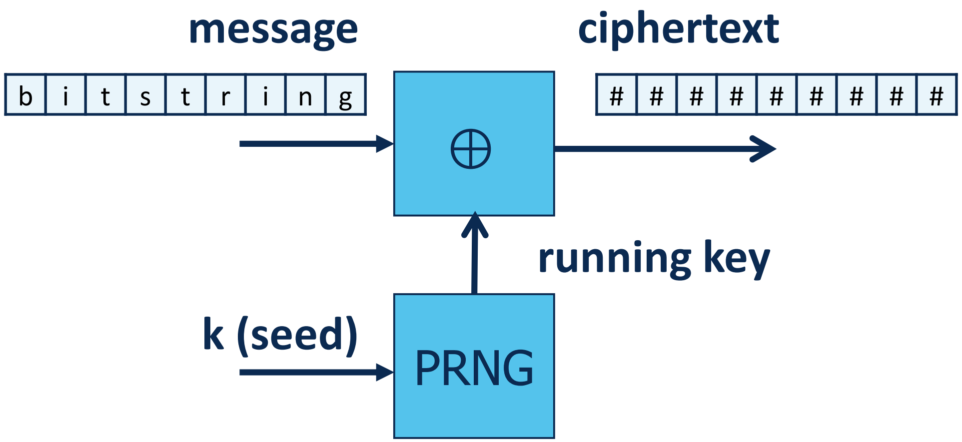
Perms[X]: the set of ***all*** one-to-one functions from X to X

PRP $E_k = \{E(k, \cdot) \ \text{s.t.}\ k \in K\} \subseteq \text{Perms}[X]$



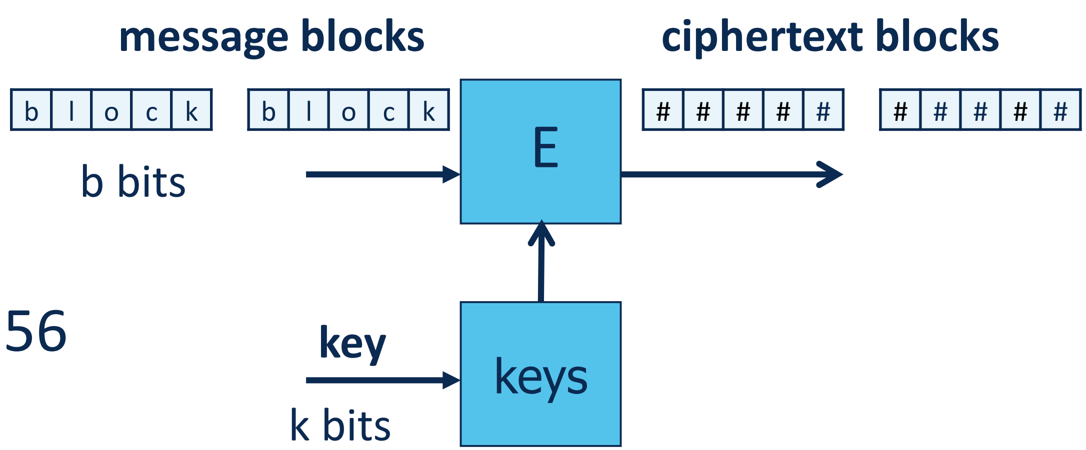


Stream Ciphers and Block Ciphers



Goal:

Build a secure PRP for b-bit blocks



Examples:

3DES: b= 64; k = 168

AES: b= 128; k = 128,192,256

Confusion – Diffusion Paradigm

Original idea:

Construct a random-looking permutation F with large block size using random-looking permutations $\{f_i\}$ with smaller block sizes

Shannon, 1949

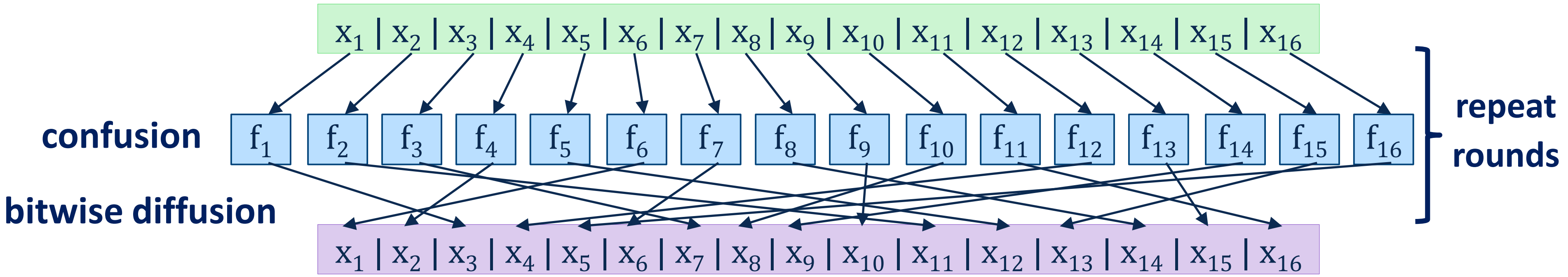
Create product cipher with confusion step (hide relation between CT and k) and diffusion step (distribute redundancy of PT)

Construction:

Lets construct $F_k : \{0,1\}^{128} \longrightarrow \{0,1\}^{128}$:

Combine $f_1,\ldots,f_{16}$ random-looking permutations $f_i : \{0,1\}^8 \longrightarrow \{0,1\}^8$,

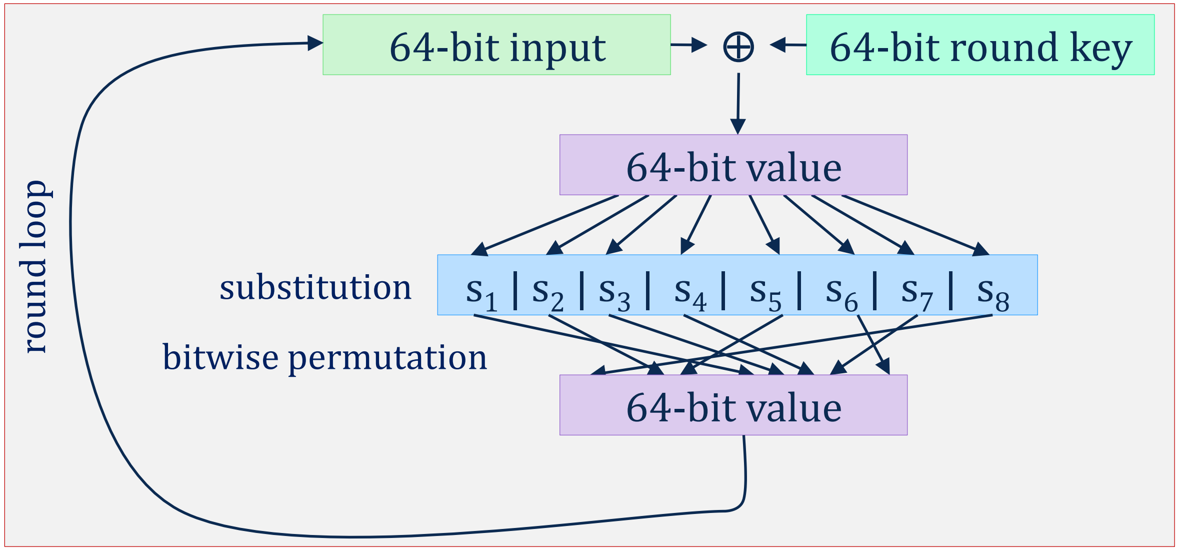
defined by random keys k_i derived from k



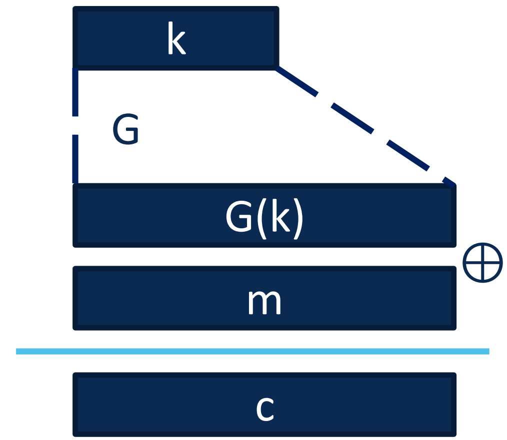
Interlude: Substitution – Permutation Nets

SPN implement the Confusion – Diffusion Paradigm:

- Round keys k_i are derived from k, then usually ⊕-ed with intermediate round output
- round functions f_i are *fixed, invertible* substitution boxes (S-Box)



Rounds and Round Keys: Key Expansion



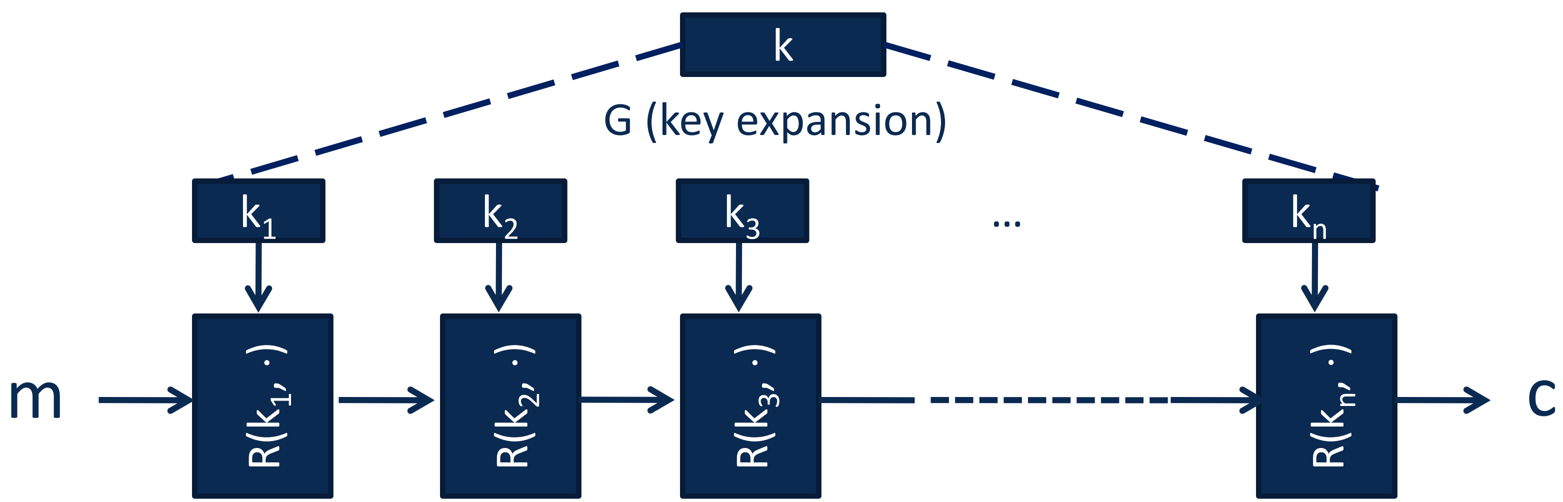
Recall from stream ciphers:

Short key expanded to encrypt bitstream

Idea:

Perform several keyed permutations in rounds

Expand key to round keys as parameters for random permutations

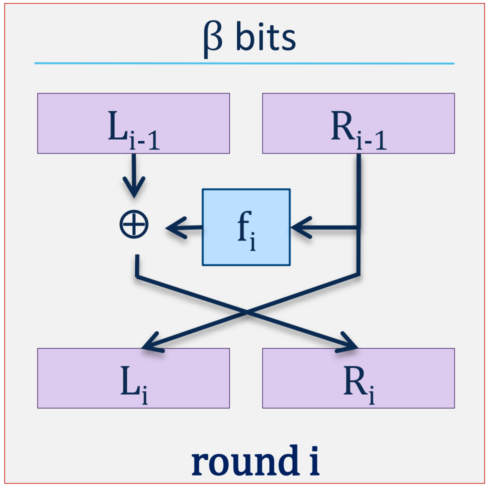


Feistel Networks

Horst Feistel

Goal:

Create a PRP from arbitrary (non-invertible) functions



Idea:

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

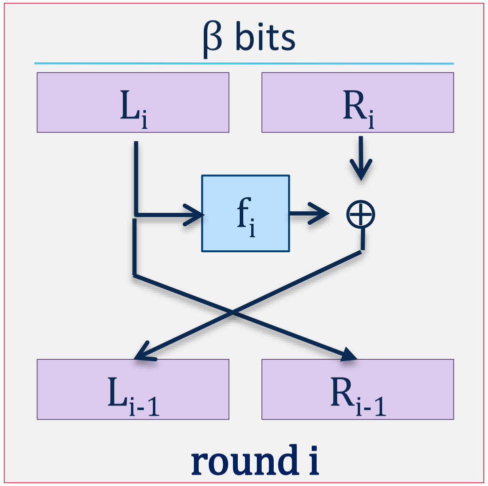
with round function f_i (possibly non-invertible),

keyed with round key k_i

Inverting is easy (basically identical, f_1 to f_d reversed):

$R_{i-1} = L_i$

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



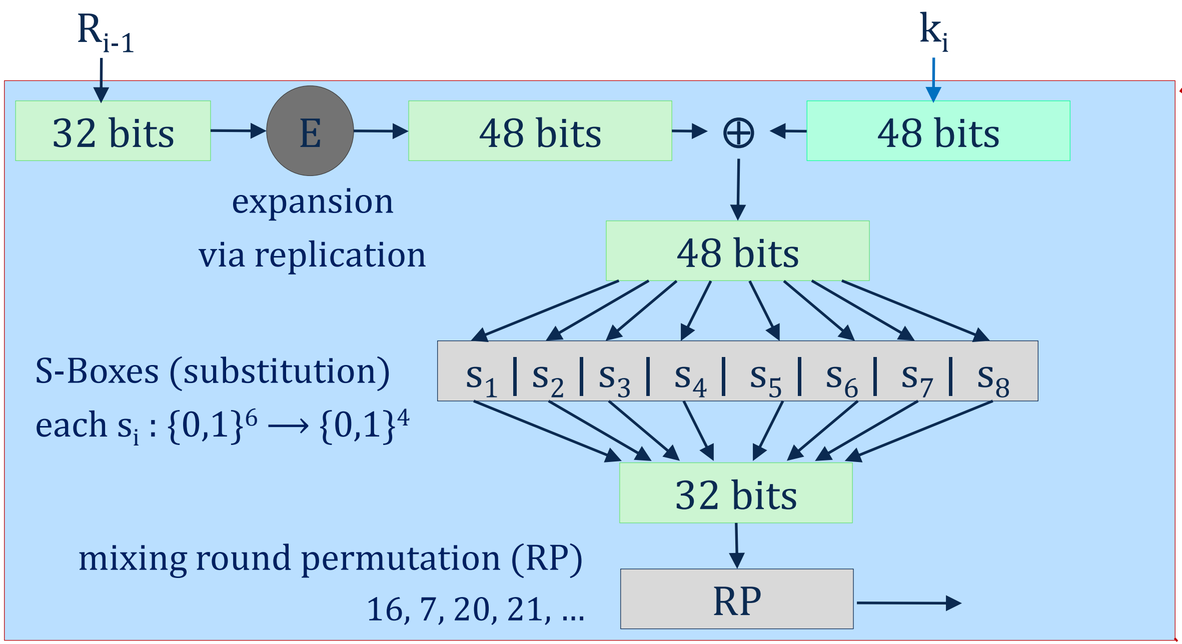
Luby-Rackoff ’85: a 3 round Feistel-Network

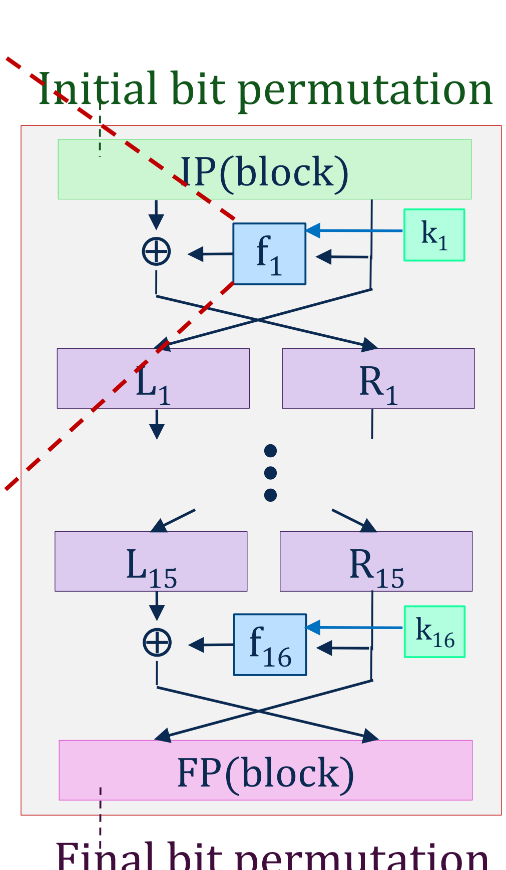
F: $K^3 \times \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$, built using PRF, is a PRP

Lucifer and the Data Encryption Standard

„Lucifer“ at DES challenge (16 rounds; b,k = 128 bit FN, IBM)

Standardized as DES after adaptation (b=64, k = 56,..., due to NSA)





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

Exhaustive Search

Given a few input-output pairs $\big(m_i,\ c_i = E(k, m_i)\big)$ i=1,..,3 , find key k.

DES challenge:

msg = "`The unknown message is:` XXXX … "

CT = c_1 c_2 c_3 c_4 …

DES broken by exhaustive search (DESCHALL) in 96 days in 1997

„The unknown message is: It's time to move to a longer key length."

distributed.net: 39 days in 1998

„The secret message is: Many hands make light work."

EFF „deep crack" (250k$) breaks DES in 56h in 1998

„The secret message is: It's time for those 128-, 192-, and 256-bit keys."

Combined search: 22h in 1999

„See you in Rome (second AES Conference, March 22-23, 1999)"

From DES to 3DES

Goal:

Strengthen DES by increasing key length

Let $E : K \times M \longrightarrow M$ be a block cipher (DES)

Define $\mathbf{3E}: K^3 \times M \longrightarrow M$ as

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

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

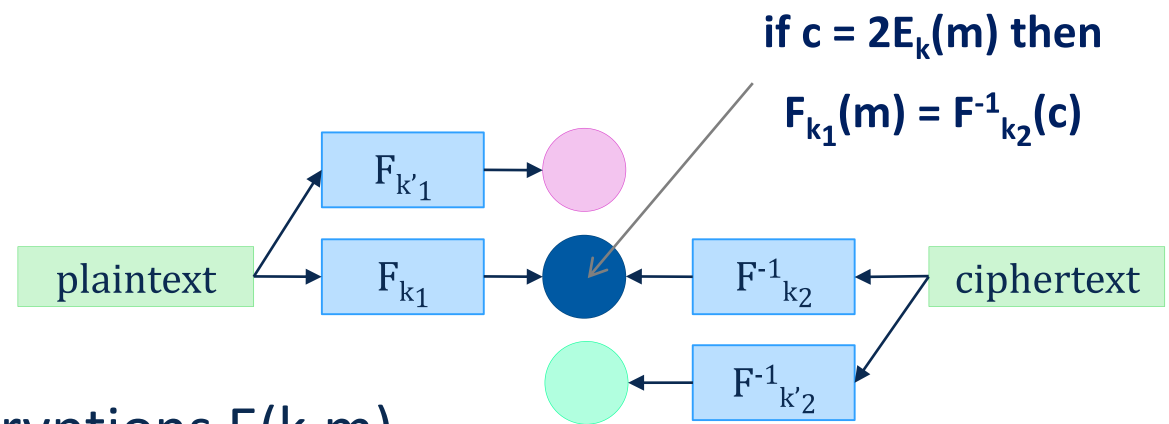
Why not E(E(E(m)))? … What if: $k_1 = k_2 = k_3$?

Simple attack feasible in time $\approx 2^{118}$

Meet-in-the-middle attack (no double DES?)

Define $2E\big((k_1,k_2), m\big) = E\big(k_1, E(k_2, m)\big)$

Idea: test if E(m) = D(c)



Step 1: build table of encryptions E(k,m)

Step 2: for all $k \in \{0,1\}^{56}$ do:

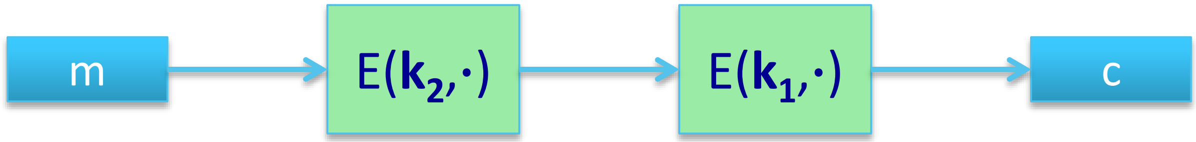
test if D(k, c) is in 2^{nd} column.

k^0 = 00…00	$E(k^0, M)$
k^1 = 00…01	$E(k^1, M)$
k^2 = 00…10	$E(k^2, M)$
⋮	⋮
k^N = 11…11	$E(k^N, M)$

2^{56} entries

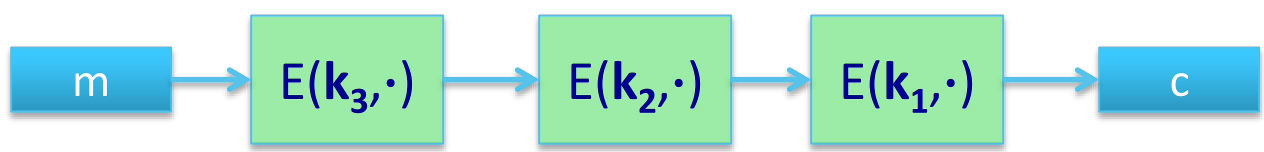
k^0 = 00…00	$E(k^0, M)$
k^1 = 00…01	$E(k^1, M)$
k^2 = 00…10	$E(k^2, M)$
⋮	⋮
k^N = 11…11	$E(k^N, M)$

Complexity of Meet-in-the-Middle



Time = $2^{56}\log(2^{56}) + 2^{56}\log(2^{56}) < 2^{63} \ll 2^{112}$, space $\approx 2^{56}$

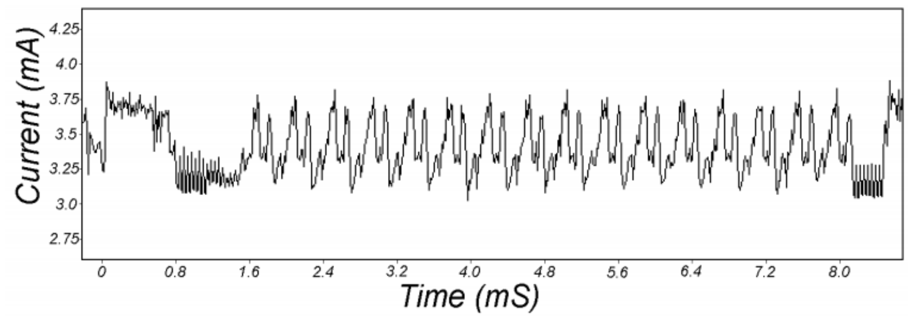
Same attack on 3DES: Time = 2^{118} , space $\approx 2^{56}$



Attacking the Implementation

1. Side channel attacks:
   - Measure **time** to do enc/dec, measure **power** for enc/dec

smartcard



[Kocher, Jaffe, Jun, 1998]

2. Fault attacks:
   - Computing errors in the last round expose the secret key k

Quantum Attacks on Block Ciphers

Generic search problem:

Let $f: X \longrightarrow \{0,1\}$ be a function.

Goal: find $x \in X$ s.t. $f(x)=1$.

Classical computer: best generic algorithm time = $O(|X|)$

Quantum Algorithm (Grover):

Given m, $c=E(k,m)$ define

$$f(k) = \begin{cases} 1 & \text{if } E(k,m) = c \\ 0 & \text{otherwise} \end{cases}$$

Quantum computer can find k in time $O(|K|^{1/2})$

DES: time $\approx 2^{28}$ (btw: ***AES-128: time*** $\boldsymbol{\approx 2^{64}}$)

Quantum adversary: 256-bits key ciphers (e.g. AES-256)

Some Requirements for S-Boxes

When designing ciphers, we require four properties (of the S-Box):

- *Completeness*: each output bit has to depend on each input bit
- *Avalanche*: Changing one input bit should effect half of the output bits
- *Correlation immunity*: output should be statistically independent from input
- *Non-linearity*: No output bit should be linear dependent on any input bit

To avoid the analyst to learn anything (easily) about

- *Plaintext*
- *Key*

Some Words on the S-Boxes

Why do we require the S-Boxes to be non-linear?

- What is a bitwise permutation?

$$Ax = y$$

- What does it mean for a transformation to be linear?

$$f_1(x) = A_1 x$$
$$f_1(\alpha x) = \alpha x \text{ (homogeneity)}$$
$$f_1(x+y) = f_1(x) + f_1(y) \text{ (additivity)}$$

- What happens, when I combine linear transformations?

$$f_1(f_2(x)) = A_1 A_2 x$$

Some Words on S-Boxes (ctd.)

- How do you multiply (matrices) in {0,1}?

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

Let's put this together:

- One round of DES (simplified), is:

$$f_i(k_i,x) := \pi(\mathrm{Subst}(x \oplus k_i))$$

- 16 rounds are

$$F(k,m) := f_{16}\,(k_{16},(f_{15}\,(k_{15},f_{14}(k_{14},f_{13}(\ldots f_1(k_1,m))..)$$

Linear S-Boxes...

Factoring out the constants:

$$64\ \overset{832=|m|+16\cdot|k|}{\boxed{\quad B \quad}} \cdot \begin{pmatrix} m \\ k_1 \\ k_2 \\ \vdots \\ k_{16} \end{pmatrix} = \boxed{c} \qquad \pmod 2$$

Then:

$$F(k,m_1) \oplus F(k,m_2) \oplus F(k,m_3)$$

$$B^{m_1}_{k} \oplus B^{m_2}_{k} \oplus B^{m_3}_{k} = B^{m_1 \oplus m_2 \oplus m_3}_{k \oplus k \oplus k}$$

$$= F(k, m_1 \oplus m_2 \oplus m_3)$$

Some Last Words on S-Boxes

S-Boxes shall not be linear transformations

They should not even be similar to linear transformations

(if you chose them at random, they would be too easy to break -> key recovery after ≈2^{24} outputs)   [BS'89]

They should also not be "easy to analyze"

-> as close to a PRF as possible

(equal output probabilities -> 4-to-1 maps (6->4bits))

etc....

(message: do not invent or implement crypto…)

The Advanced Encryption Standard

Joan Daemen / Vincent Rijmen

1997: NIST publishes request for proposal

1998: 15 submissions

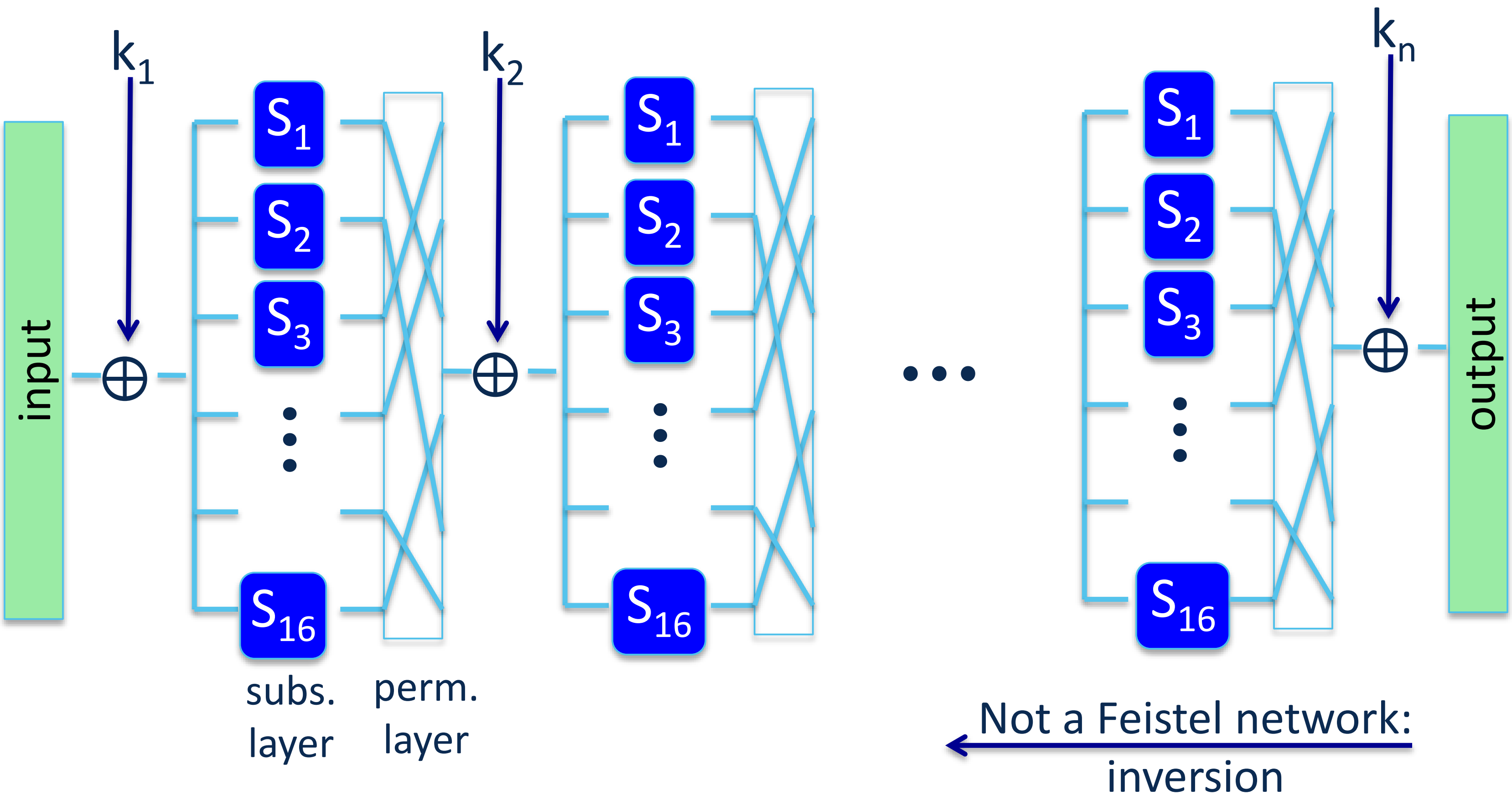
1999: NIST chooses 5 finalists
(Mars: IBM, RC6: RSA, Rijndael: Rijmen/Daemen – Belgium, Serpent: Anderson/Biham/Knudsen, Twofish: Bruce Schneier et al.)

2000: NIST chooses Rijndael as AES

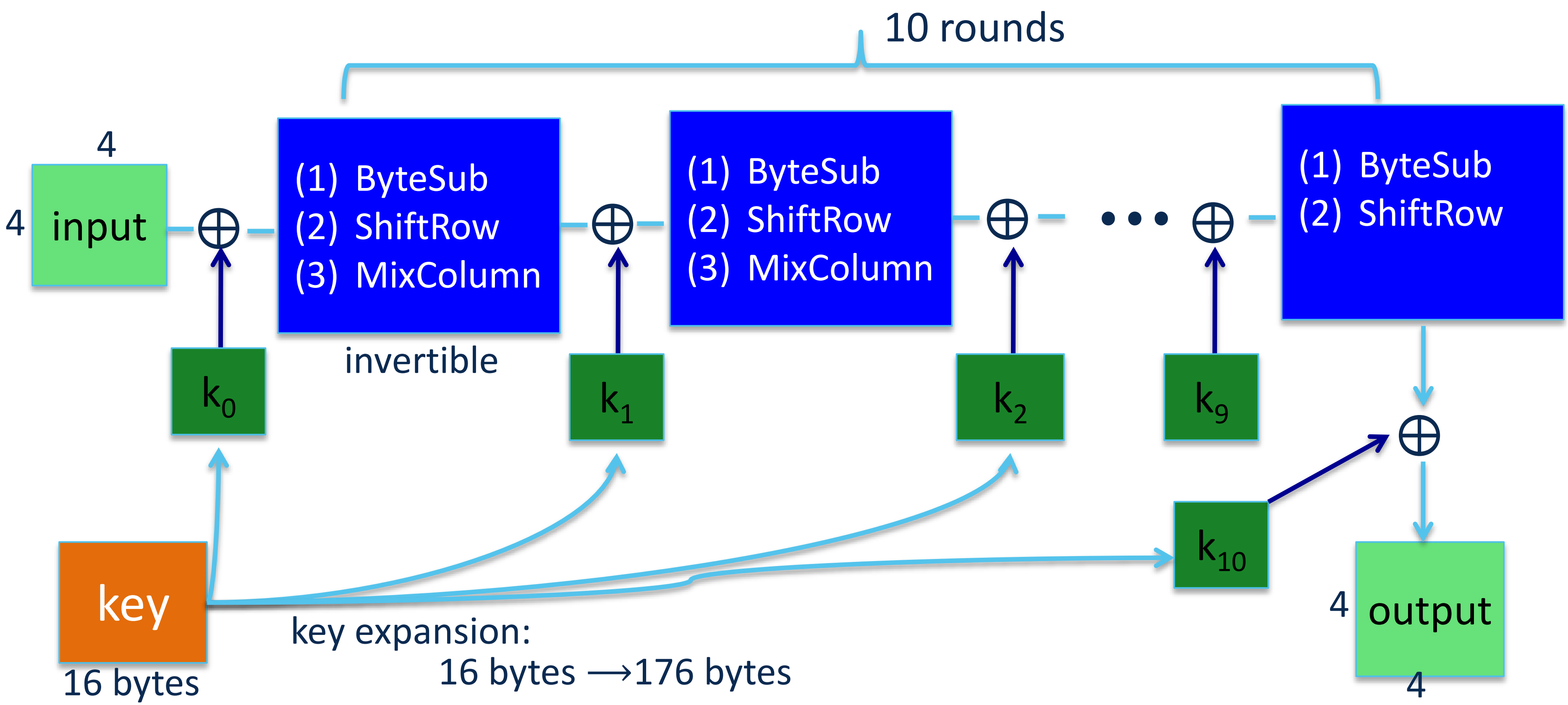
Key sizes: 128, 192, 256 bits Block size: 128 bits

Best known (theoretical) attacks in time $\approx 2^{99}$

AES Substitution-Permutation Network



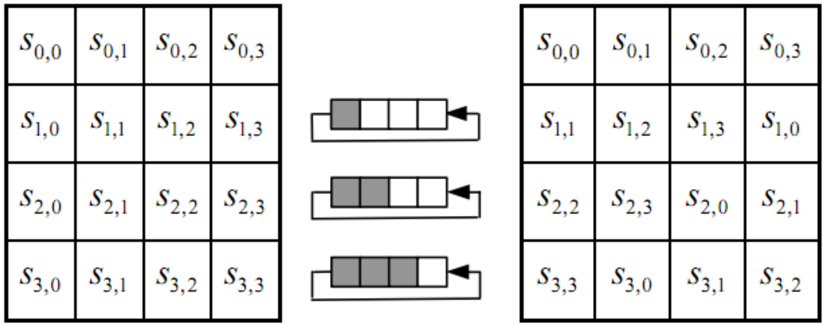
AES-128 scheme



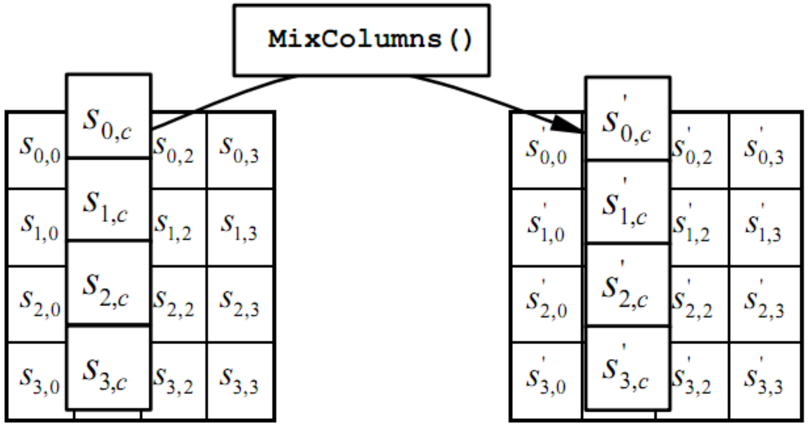
AES round functions

ByteSub: a 1 byte S-box. 256 byte table (easily computable)

ShiftRows:



MixColumns:



AES is Secure and *Efficient*

For the Web:

- JavaScript implementation (6.4KB)
- ByteSub tables not transmitted, but precomputed on client

Implementation in Hardware (Intel, similar on AMD)

- **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

Performance of Ciphers

Block ciphers are much than stream ciphers *(Why?)*

Comparison (AMD Opteron, 2.2 GHz, Linux, Crypto++ 5.6.0)

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

Performance of Ciphers

Block ciphers are much slower than stream ciphers *(Why?)*

Comparison (AMD Opteron, 2.2 GHz, Linux, Crypto++ 5.6.0)

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

Intermediate question:

Considering (E,D), PRF and PRP, what are AES/3DES?

Building Block Ciphers (Modes of Operation)

So far we have seen PRFs and PRPs (3DES, AES)

Goal:

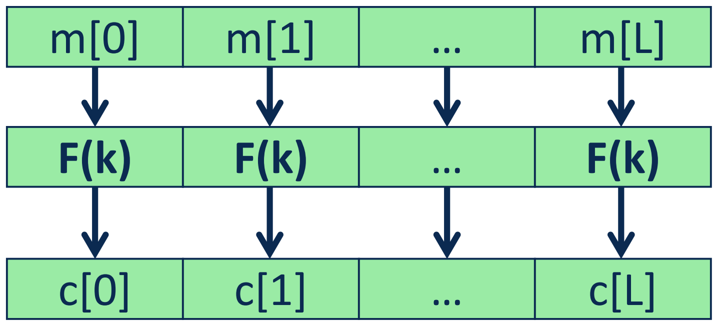
Build „secure“ encryption from secure PRPs

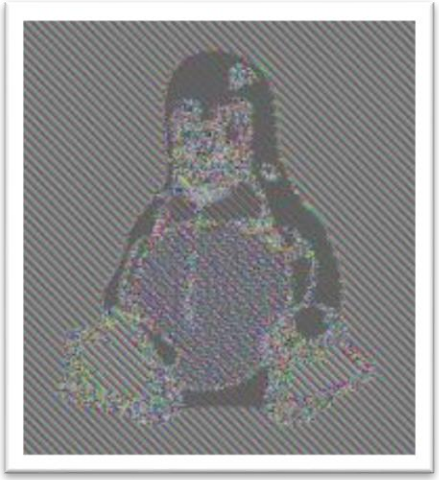
Only one-time keys for the moment:

- Adversary can submit only two messages
- Sees only one ciphertext
- Aims at learning about PT/k from CT (semantic security)

Electronic Code Book Mode (insecure!)

Encrypt each block with the keyed PRP:





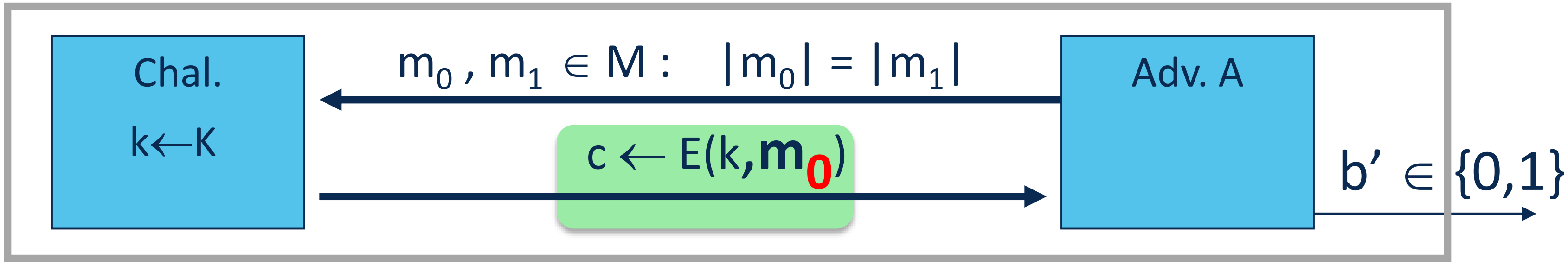
ECB encryption is *deterministic*

⇒ identical PT is encrypted to identical CT:

Is this "secure" (how)?

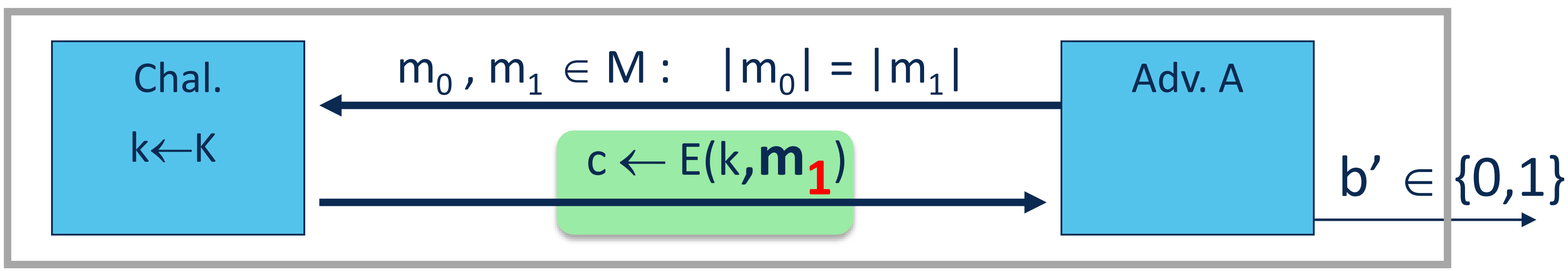
Semantic Security (One Time Key)

EXP(0):



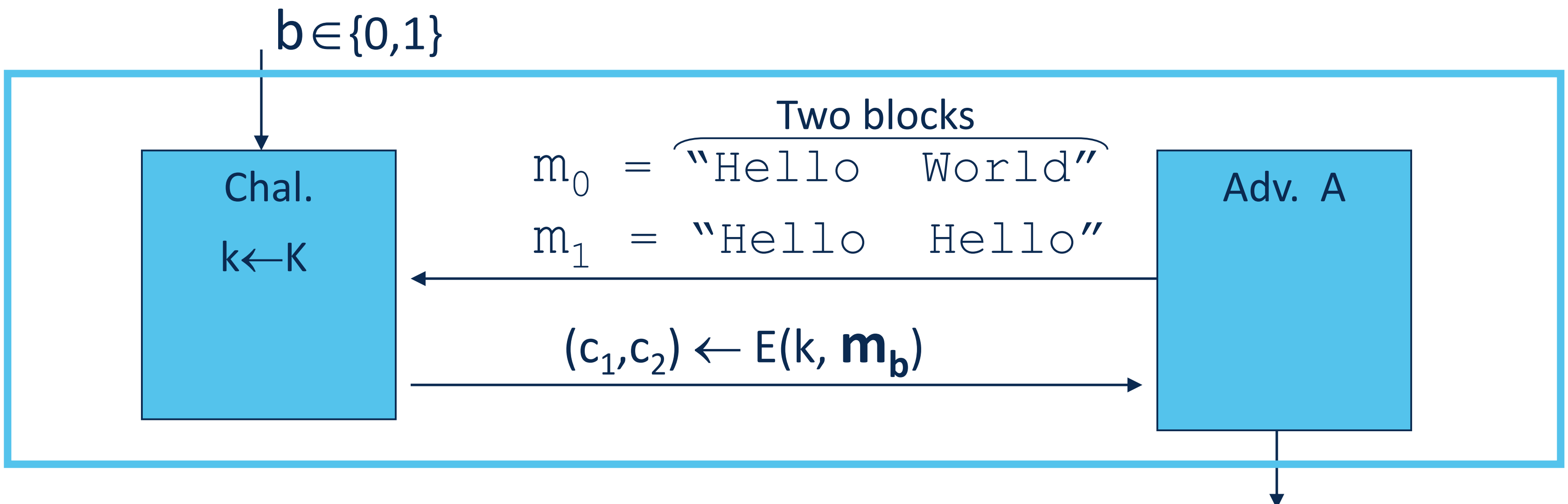
one time key $\Rightarrow$ adversary sees only one ciphertext

EXP(1):



$Adv_{SS}[A,ECB] = \Big| \Pr[\,\mathbf{EXP(0)}=1\,] - \Pr[\,\mathbf{EXP(1)}=1\,] \Big|$ should be “neg.”

ECB not semantically secure

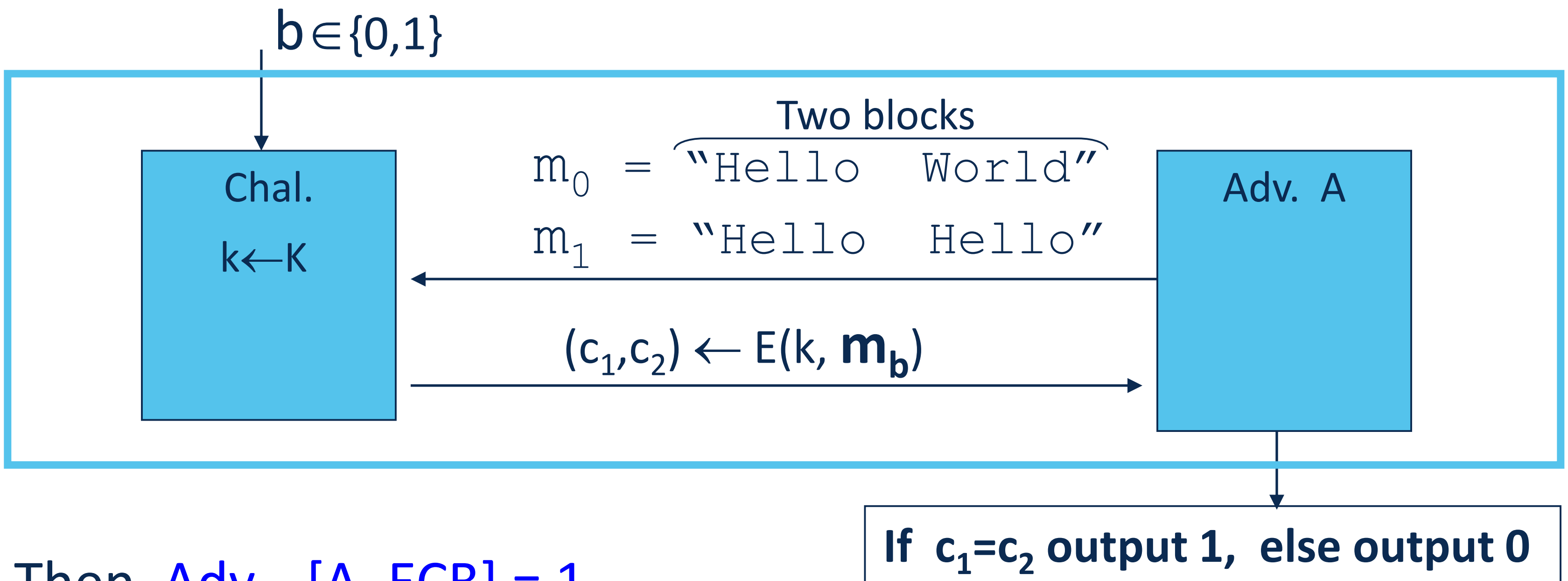


If $c_1=c_2$ output 1, else output 0

Then $\mathrm{Adv}_{SS}[A, ECB] =$

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

ECB not semantically secure



Then $Adv_{SS}[A, ECB] = 1$

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

Deterministic Counter Mode

XOR each block with changing keys:

Encrypt key and counter, take a PRF $F: K \times \{0,1\}^n \longrightarrow \{0,1\}^n$

$E_{DETCTR}(k,m) =$

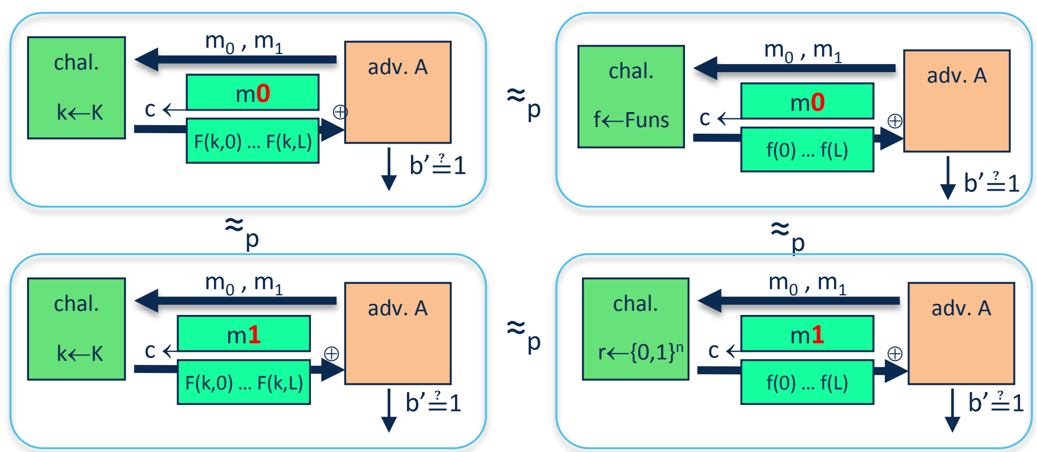
	m[0]	m[1]	…	m[L]
$\oplus$	**F(k,0)**	**F(k,1)**	…	**F(k,L)**
	c[0]	c[1]	…	c[L]

$D_{DETCTR}(k,m) = ?$

Proof of semantic security:

$Adv_{SS}[A, E_{DETCTR}] \leq \varepsilon$

Note: PRF (pot. non-invertible)



CPA and Many Time Keys

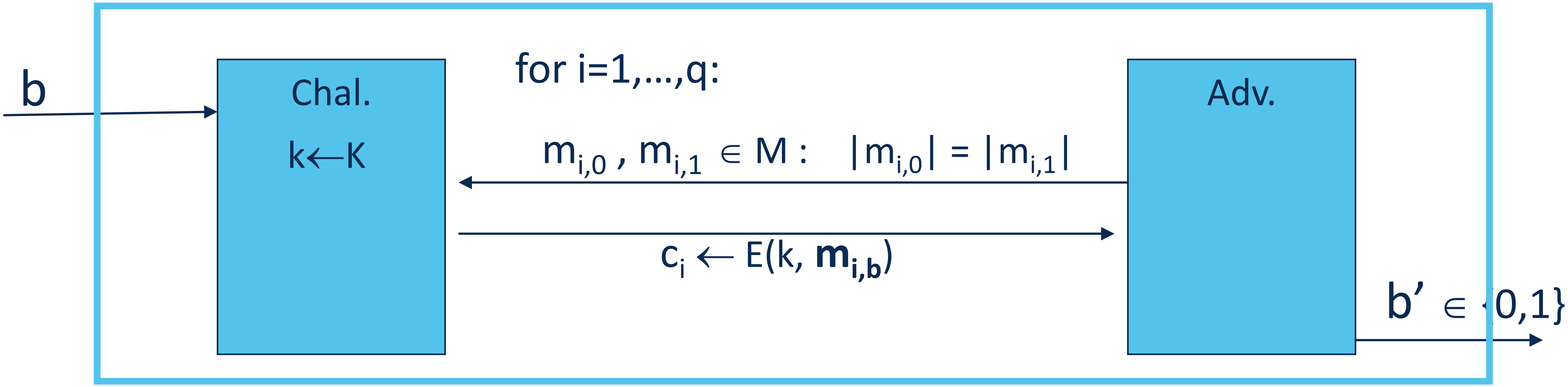
Assumption:

Keys are used more than once ⇒ adv. sees many CTs with same key

Extension to one-time key:

- Adversary can obtain the encryption of arbitrary messages of his choice *(conservative modeling of real life)*
- Aims at breaking semantic security (learn anything about PT)

CPA Security (SS for many time key)



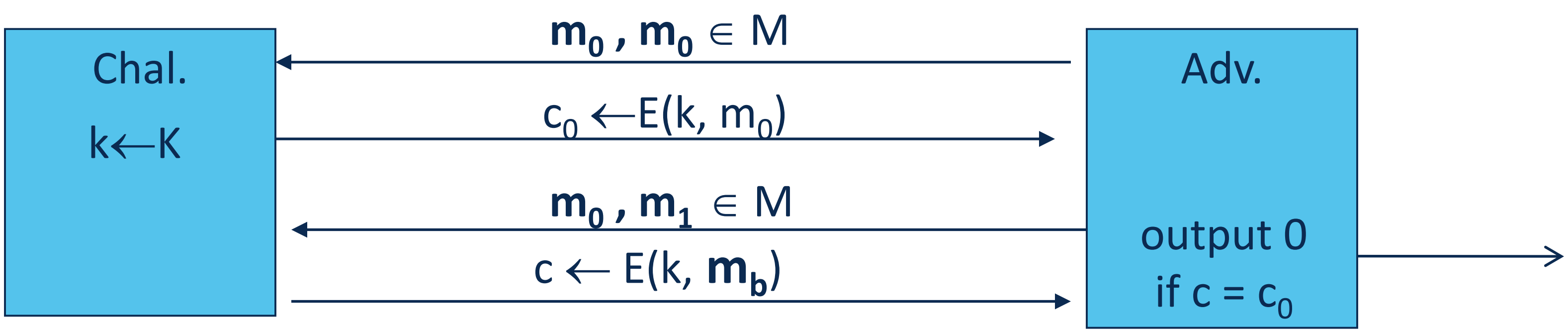
$\mathbb{E} = (E,D)$ a cipher defined over (K,M,C).

$\mathbb{E}$ is sem. sec. under CPA if for all “efficient” A:

$$\mathrm{Adv}_{\mathrm{CPA}}[A,\mathbb{E}] = |\Pr[\mathrm{EXP}(0)=1] - \Pr[\mathrm{EXP}(1)=1]| \leq \varepsilon$$

Deterministic Ciphers aren‵t IND-CPA

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



An attacker can learn that two encrypted files are the same, two encrypted packets are the same, etc.

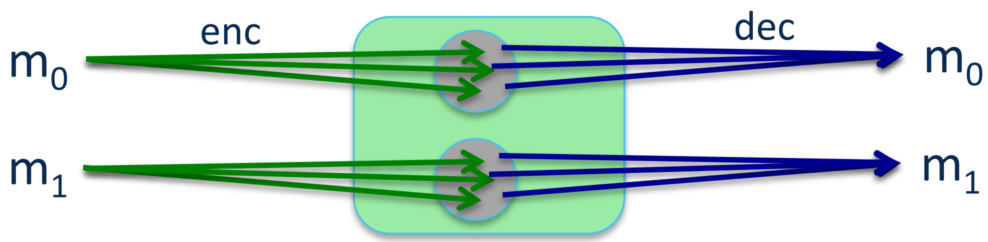
⇒ Leads to significant attacks when message space M is small

If secret key is to be used multiple times:

If repetition of plaintext possible, E must produce different outputs!

Random / Nonce-based Encryption

E(k,m) maps identical m to different c:



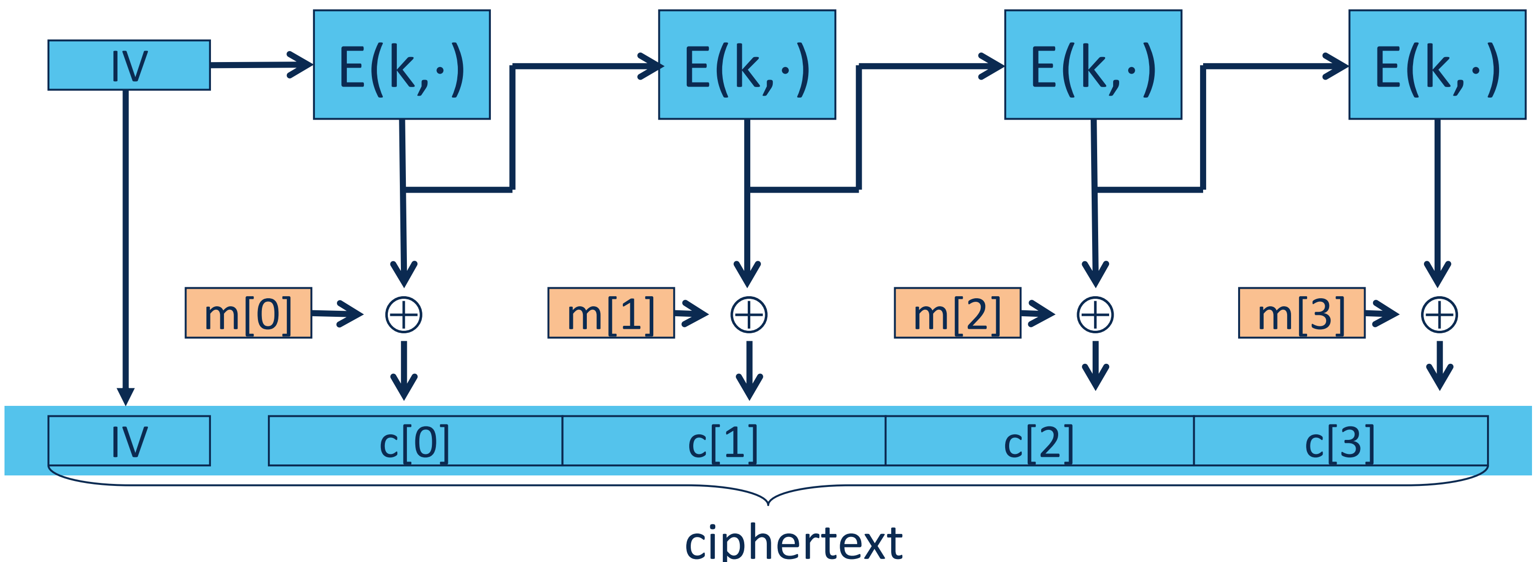
$E_i\,(k,m_l) = E_j\,(k,m_l) \Rightarrow i{=}j$

Transmission is longer than plaintext (transmit nonce)

- Use counter as nonce (shared state, never same nonce and k!)
- Choose random nonce $n \leftarrow \mathcal{N}$

Output Feedback Mode

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



Remarks:

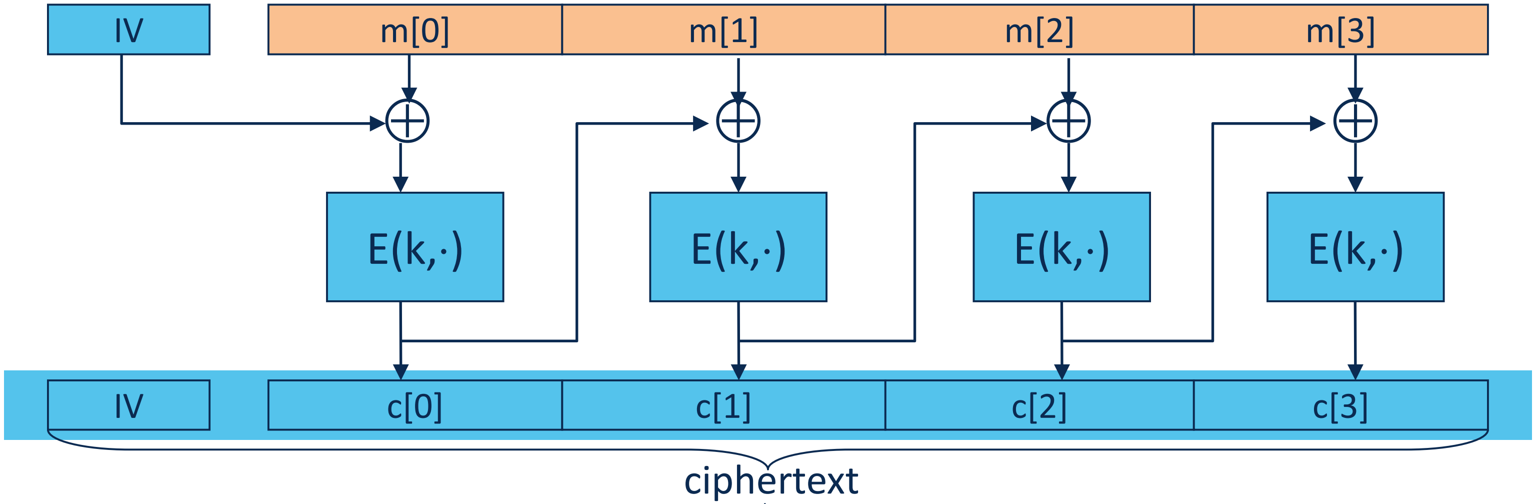
Dependency between subsequent packets

E and D cannot be parallelized, ***but***

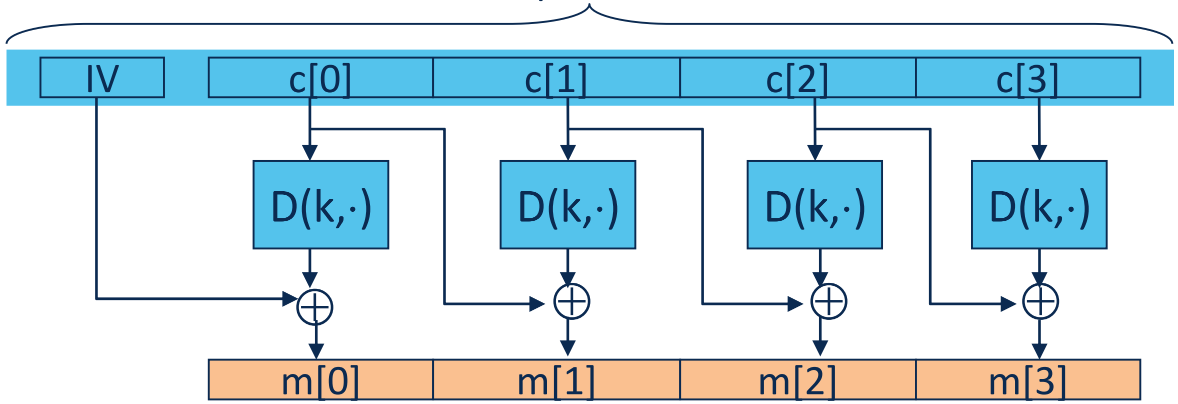
E(k,E(k,E(k,…E(k,IV))) can be precomputed

CBC with random IV

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



Decrypt?

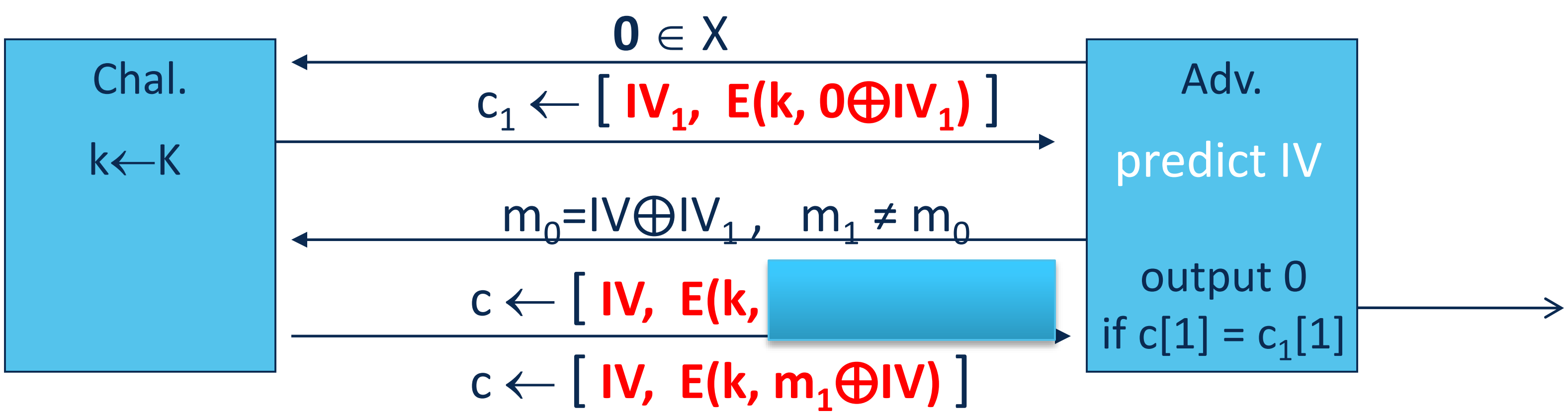


Dependency between subsequent packets, D can be parallelized

Warning: Attacking CBC with Random IV

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

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

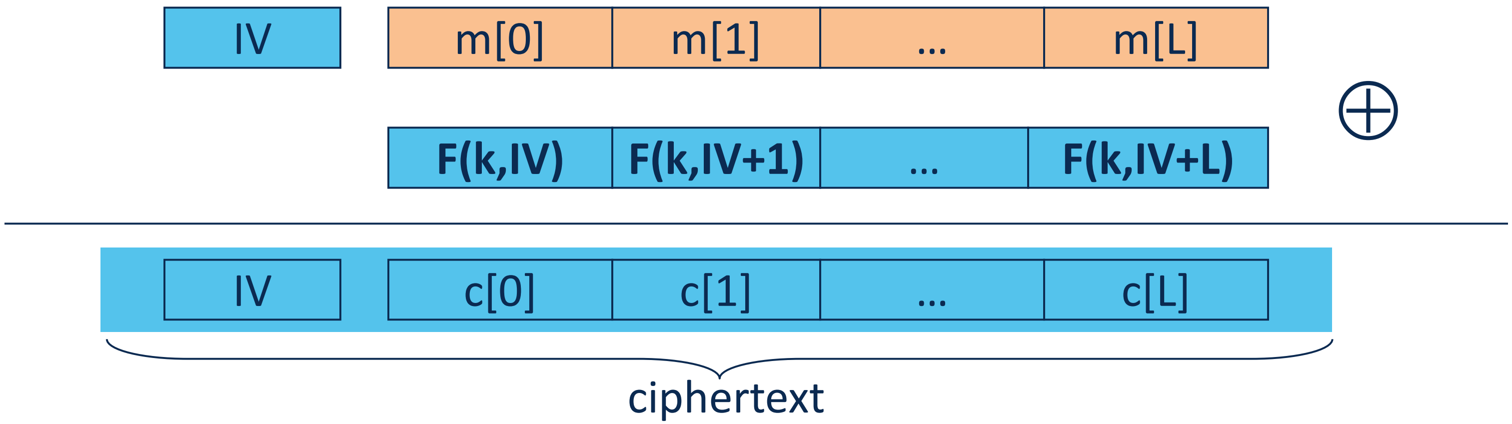


Bug in SSL/TLS 1.0: IV for record #i is last CT block of record #(i-1)

Randomized Counter Mode R-CTR

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

E(k,m): choose a random IV $\in \{0,1\}^n$ and do:



Variation: Choose 128 bit IV as: nonce || counter

Remarks:

E, D can be parallelized and F(k,IV+i) can be precomputed

R-CTR allows random access, any block can be decrypted on its own

Again: F can be any PRF, no need to invert

Summary

You recall properties of functions

You can explain what (Trapdoor) One-way functions are

You know what PRFs and PRPs are

You can also show against what they are secure

You can explain Feistel Networks and DES/3DES

You can break DES (and you can explain how it‘s done)

You know about Substitution Permutation Networks and AES

You saw different modes of operation and know their properties