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 16/17

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

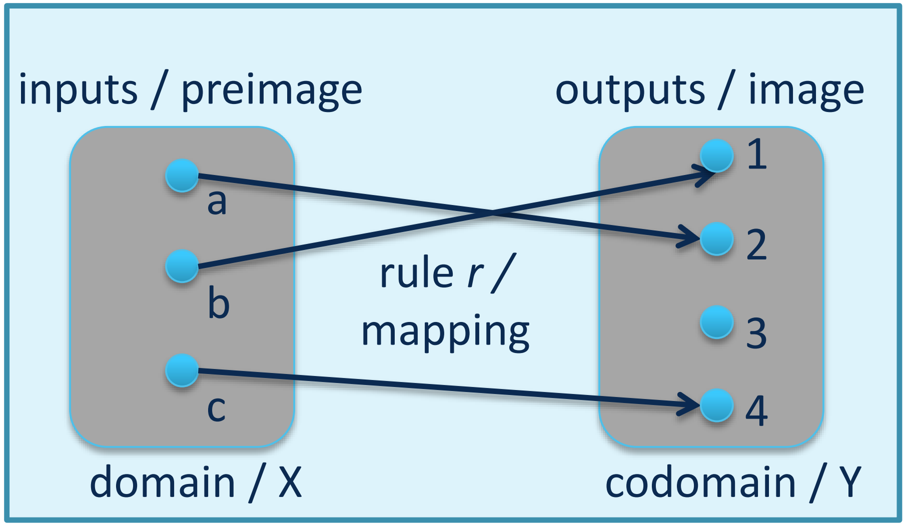
Pseudo Random Permutations

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…



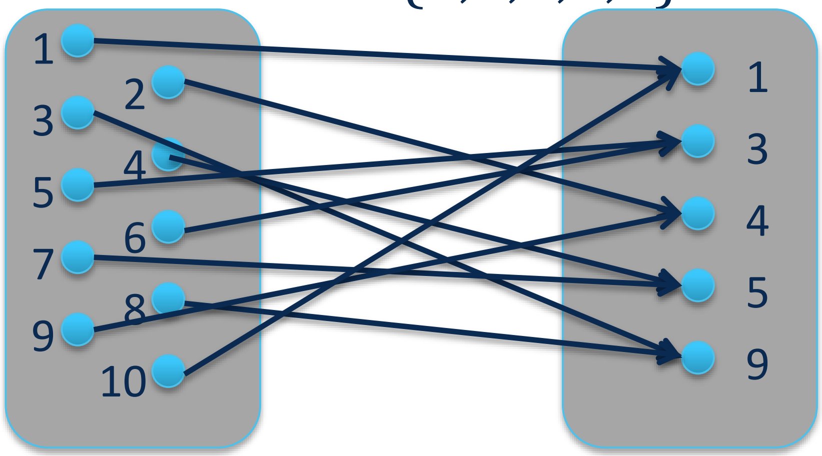
$f: X \longrightarrow Y$ $\qquad$ $y = f(x)$

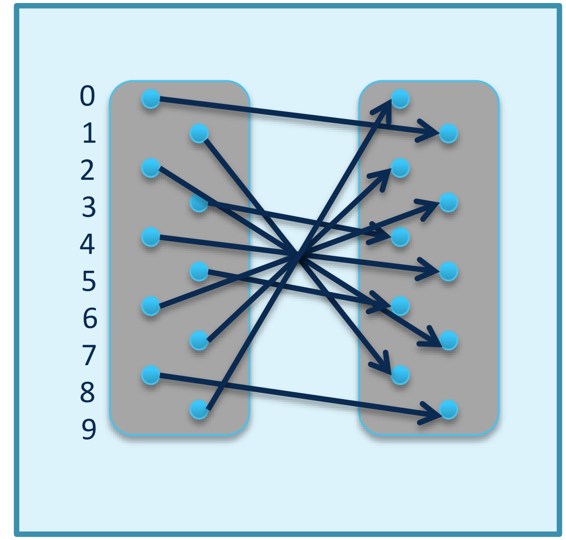
$X = \{a,b,c\}$ $\qquad$ $Y = \{1,2,3,4\}$ $\qquad$ $Im(f) = \{1,2,4\}$

Functions, Functions, Functions

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

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





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): $\forall x1, x2 \in X: f(x1) = f(x2) \Rightarrow x1 = x2$

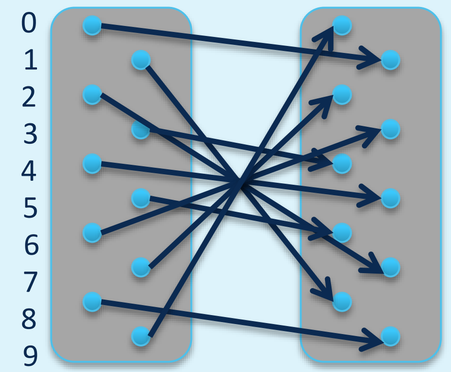
bijection: $f(x)$ *is* $1 - 1$ *and* $Im(f) = Y$

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

Hint: (Trapdoor) One-way Functions

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

$X = \{0,\ldots,9\}$, $Y = \{0,\ldots,9\}$
$f(x) = (x+1)^3 \bmod 10$



One way functions:

A function $f: X \longrightarrow Y$ 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:

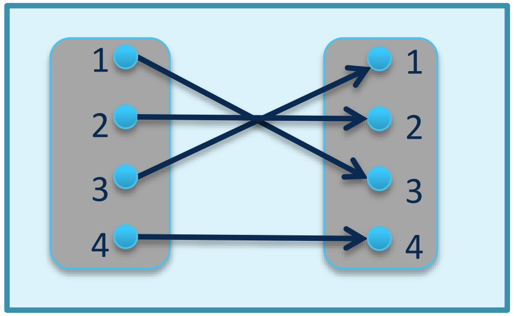
A *trapdoor one-way function* is a one-way function that, given some additional *trapdoor information,* is feasible to invert.

(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 $D = E^{-1}$

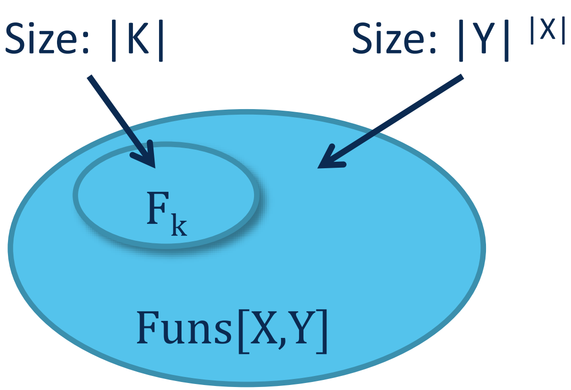
Security of PRFs

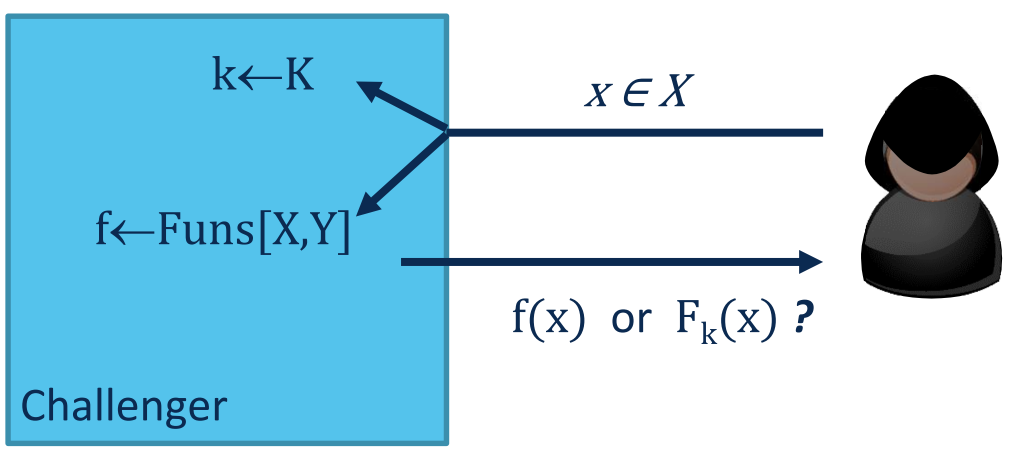
A PRF is secure, if it is indistinguishable from a random function:

Consider

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

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





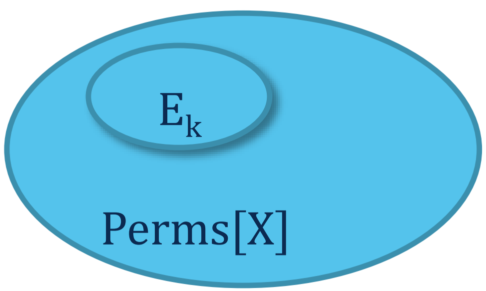
Security of PRPs

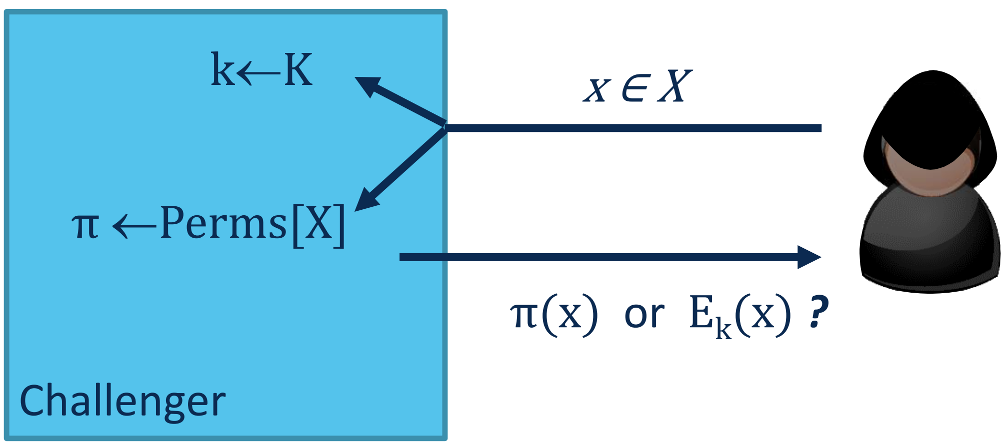
A PRP is secure, if it is indistinguishable from a random permutation:

Consider

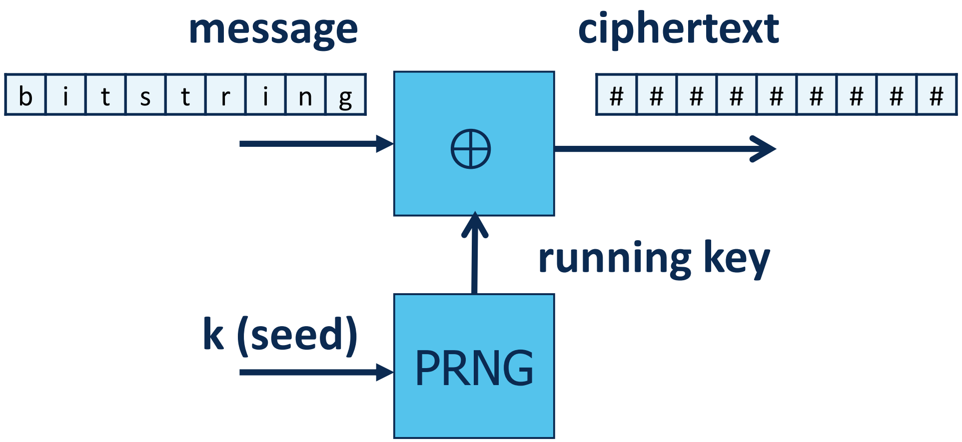
Perms[X]: the set of ***all*** one-to-one functions from X to X

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





Stream Ciphers and Block Ciphers



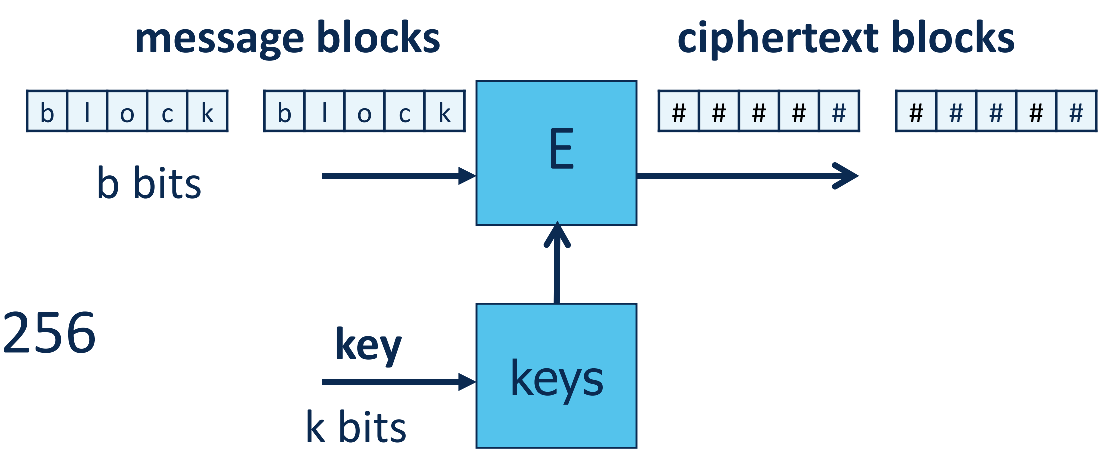
Goal:

Build a secure PRP for b-bit blocks

Examples:

3DES: $n = 64$, $k = 168$

AES: $n = 128$, $k = 128,192,256$



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

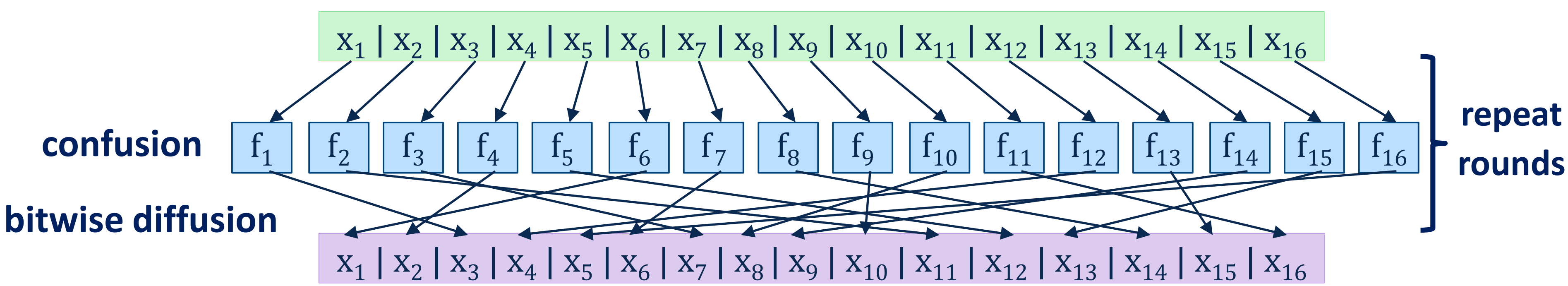
Create product cipher with confusion step (hide relation between CT and k) and diffusion step (distribute redundancy of PT)

Construction:

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

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

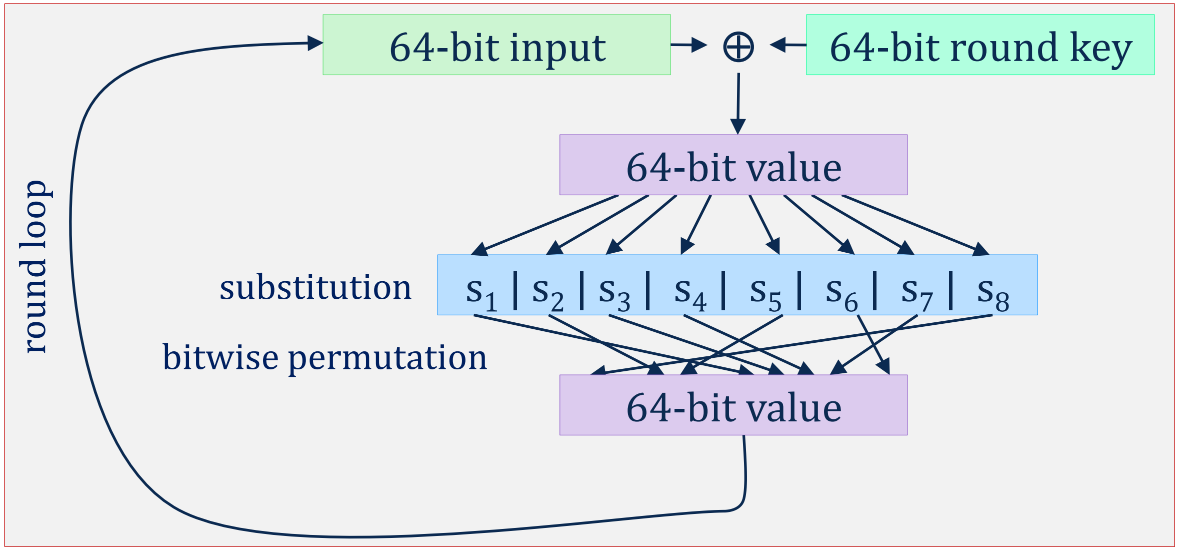
defined by random keys k_i derived from k



Interlude: Substitution – Permutation Nets

SPN implement the Confusion – Diffusion Paradigm:

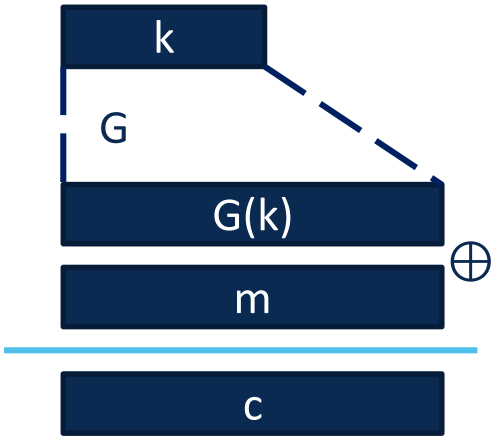
- Round keys k_i are derived from k, then usually ⊕-ed with intermediate round output
- round functions f_i are *fixed, invertible* substitution boxes (S-Box)



Rounds and Round Keys: Key Expansion

Recall from stream ciphers:

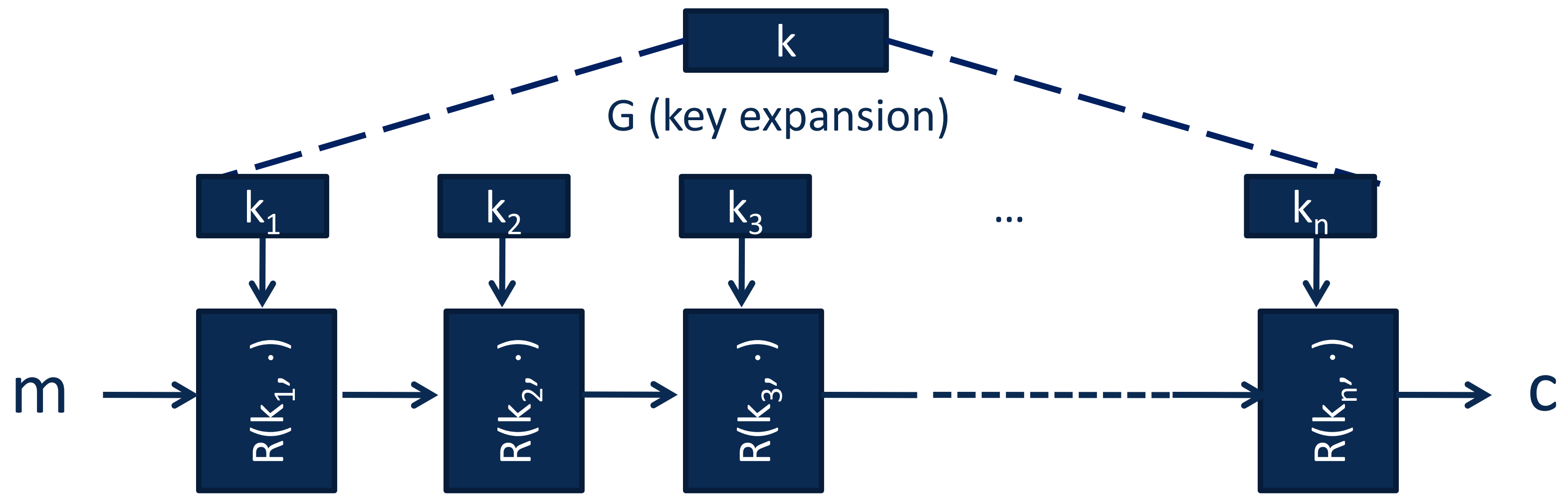
Short key expanded to encrypt bitstream



Idea:

Perform several keyed permutations in rounds

Expand key to round keys as parameters for random permutations

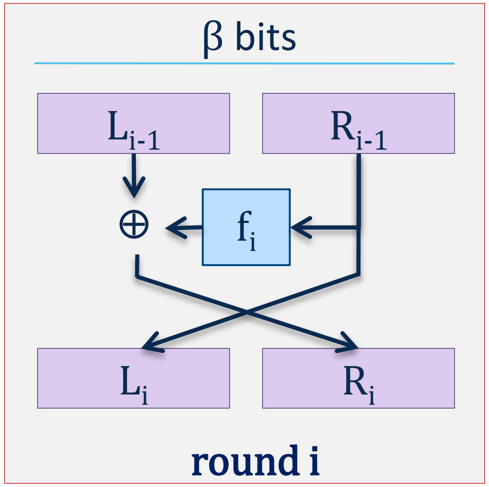


Feistel Networks

Horst Feistel

Goal:

Create a PRP from arbitrary (non-invertible) functions



Idea:

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

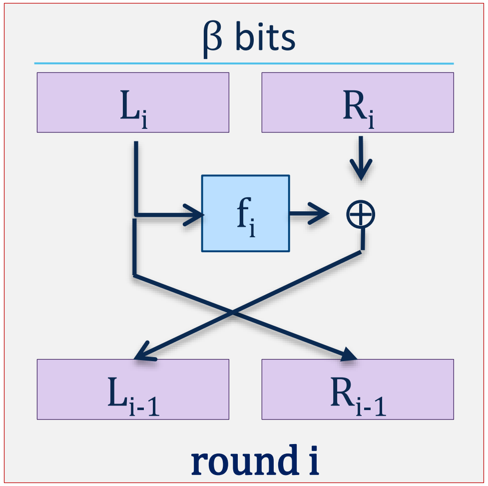
with round function f_i (possibly non-invertible),

keyed with round key k_i

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

$R_{i-1} = L_i$

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



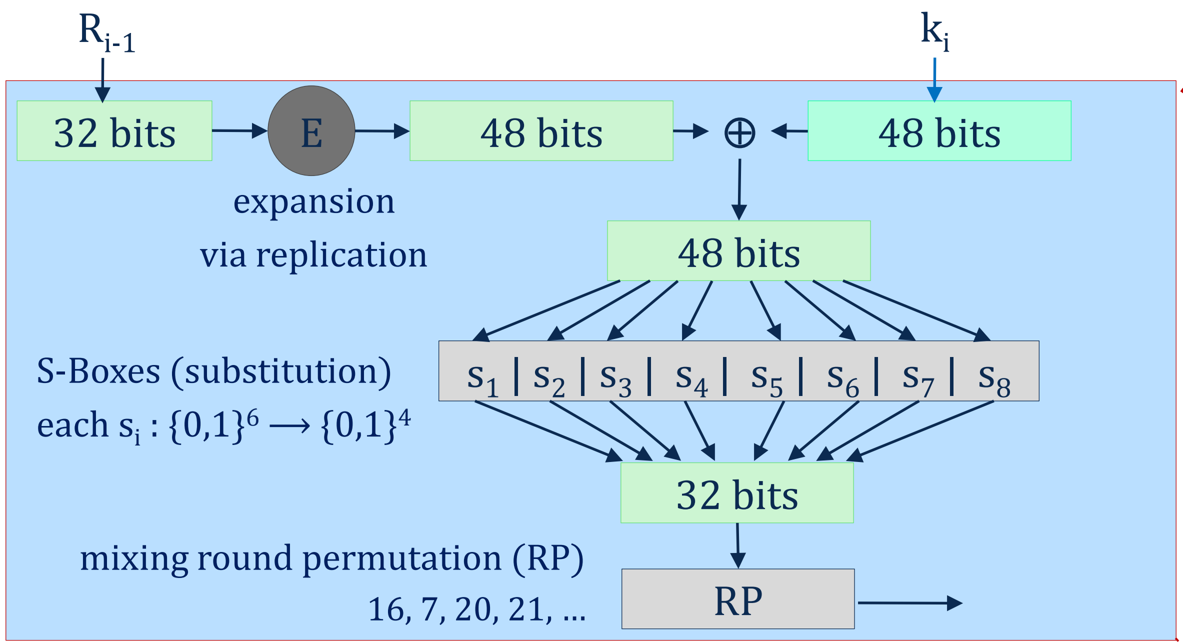
Luby-Rackoff '85: a 3 round Feistel-Network

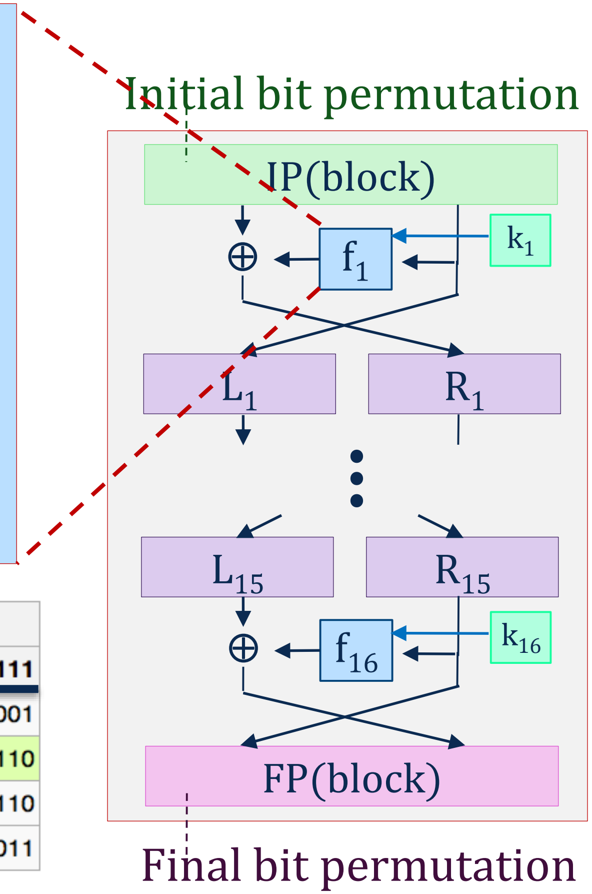
$F\colon\ K^3 \times \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$, built using PRF, is a PRP

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 $\left(m_i, c_i = E(k, m_i)\right)$ 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 **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)))? …

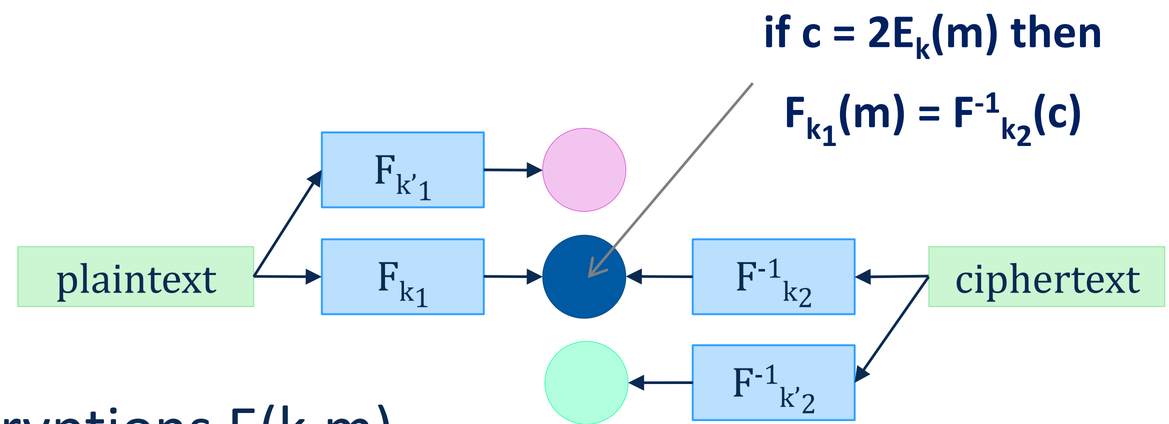
What if: $k_1 = k_2 = k_3$?

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 2nd column.

$k^0 = 00\ldots00$	$E(k^0, M)$
$k^1 = 00\ldots01$	$E(k^1, M)$
$k^2 = 00\ldots10$	$E(k^2, M)$
⋮	⋮
$k^N = 11\ldots11$	$E(k^N, M)$

2^{56} entries

$k^0 = 00\ldots00$	$E(k^0, M)$
$k^1 = 00\ldots01$	$E(k^1, M)$
$k^2 = 00\ldots10$	$E(k^2, M)$
⋮	⋮
$k^N = 11\ldots11$	$E(k^N, M)$

Complexity of Meet-in-the-Middle

$$m \rightarrow E(\mathbf{k_2},\cdot) \rightarrow E(\mathbf{k_1},\cdot) \rightarrow c$$

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

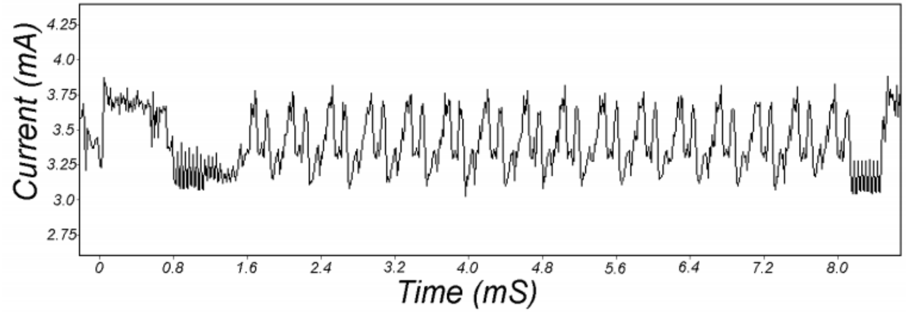
Same attack on 3DES: Time = 2^{118} , space $\approx 2^{56}$

$$m \rightarrow E(\mathbf{k_3},\cdot) \rightarrow E(\mathbf{k_2},\cdot) \rightarrow E(\mathbf{k_1},\cdot) \rightarrow c$$

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)