TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Betriebssysteme und ***Sicherheit***

Stefan Köpsell, Thorsten Strufe

Modul 5: Mechanismen - Vertraulichkeit

Disclaimer: Inhalte von Dan Boneh, Mark Manulis, Günter Schäfer, Mitarbeitern des Lehrstuhls

Dresden, WS 16/17

The story so far…

Sicherheit als abstrakte Anforderung

Safety vs. Security

Datenschutz und Aspekte des Datenschutzes

Formale Ziele der IT Sicherheit

Bedrohungen – abstrakt und technisch

Bedrohungsanalysen

Angreifermodelle und Angriffstechniken

Risiko-Analysen

Was kann man tun: Sicherheitsmechanismen

Identität und Authentisierung

Schlüssel-Management

Integrität

Vertraulichkeit

Zugriffskontrolle

Module Outline

Einige Worte zu Krypto

Ein kleiner Abriß der Geschichte

- Transposition
- Substitution
- Krypto-analyse
- Cesar Cipher
- Vigenère Cipher
- Enigma
- Vernam Cipher – The One Time Pad

Confidential Communication

Spaces

$\mathcal{M}$ plaintext space (e.g. words over an alphabet)

$\mathcal{C}$ space of ciphertexts

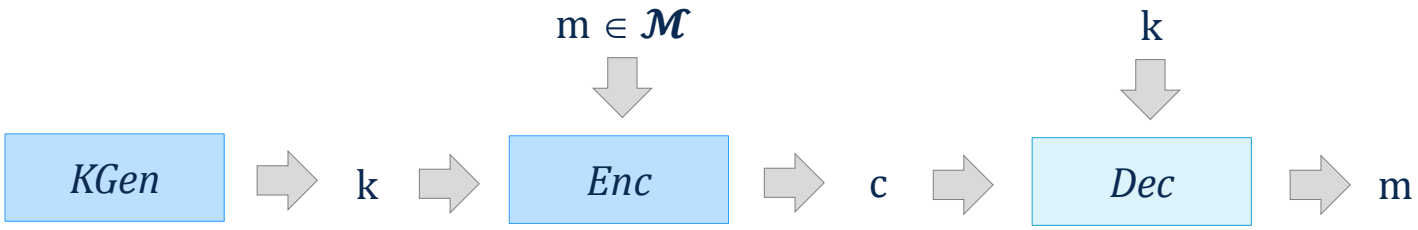
$\mathcal{K}$ space of keys

Algorithms of a private-key (symmetric) encryption scheme

KGen generates some (usually random) *key* k

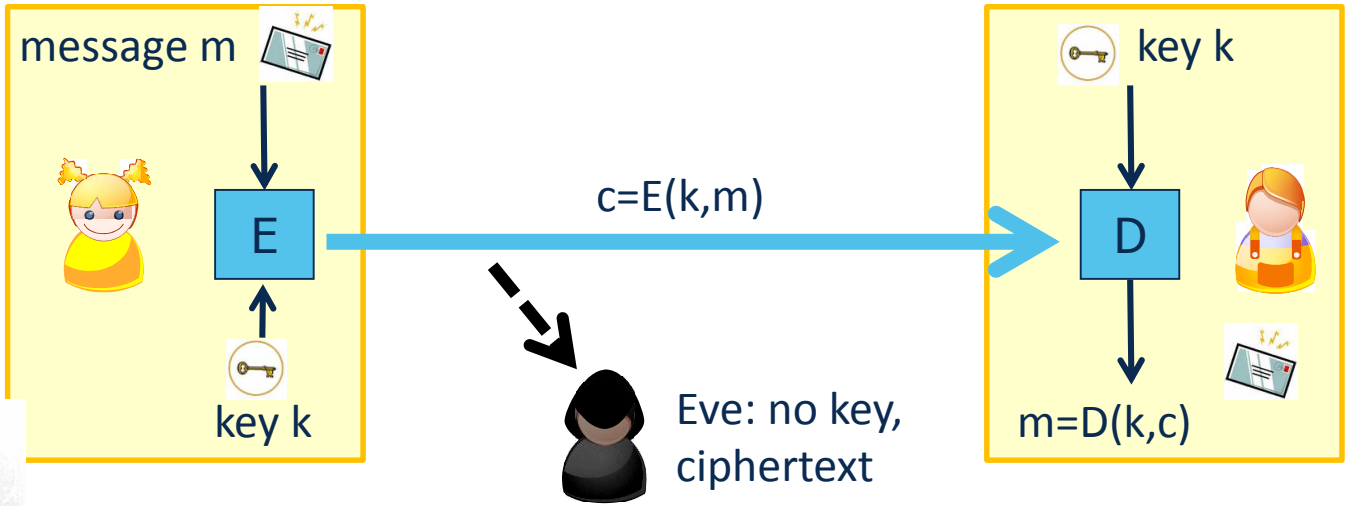
Enc encrypts a *plaintext* m using key k and outputs the ciphertext c

Dec decrypts a *ciphertext* c using key k and outputs the plaintext m



Correctness $\text{for all } k \in \mathcal{K}, m \in \mathcal{M} \ : \ \text{Dec}(k, \text{Enc}(k, m)) = m$

The communication model and Kerckhoff



"The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience."

i.o.w: KGen, E, and D will inevitably be discovered at some stage

→ All algorithms should be public

→ security must rely on secrecy of the key only

Terminology: Cryptology (Kryptologie)

Cryptology:

- Science concerned with communications in secure and usually secret form
- Derived from the Greek
  - *kryptós (*hidden) and
  - *lógos (*word)

- Cryptology encompasses:
  - ***Cryptography*** (*gráphein* = to write): principles and techniques by which information can be concealed in *ciphertext* and later revealed by legitimate users employing a secret key

  - ***Cryptanalysis*** (*analýein* = to loosen, to untie): recovering information from ciphers without knowledge of the key

Terminology: Cipher (Chiffren)

Cipher (Chiffren):

- Method of transforming a message (plaintext) to conceal its meaning (and to transform it back)

- Ciphers are one class of cryptographic algorithms (E,D)
- The transformation usually takes the message and a *(secret) key* as input

- Unfortunately: sometimes also used as synonym for the concealed *ciphertext (Chiffrat!)*

Achieving the security goals

Recall CIA:

- ***C***onfidentiality: only authorized access to information
- ***I***ntegrity: detection of message modification
- ***A***vailability: services are live and work correctly

Where crypto can (trivially) help:

- Confidentiality: Encryption transforms plaintext to conceal it
  - Symmetric crypto (single key)
  - Asymmetric crypto (key-pair)
- Integrity
  - Message authentication / signing of data with authenticated digest (cryptographic hash/signature)

Classifying Encryption Algorithms

Type of operation

- Substitution
- Transposition

Number of keys

- Symmetric: secret key
- Asymmetric: „public key“, pair of public and private key

Processing of plaintext

- Stream ciphers: operate on streams of bits
- Block ciphers: operate on b-bit blocks

Crypto Basics

Encrypt written communication:

$\mathcal{M}$: language over
$\mathcal{C}$: language over

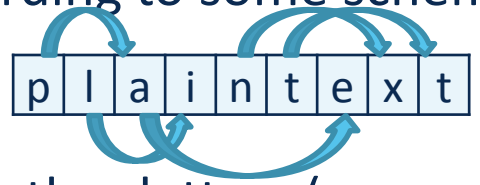
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

$\mathcal{K}$ is determined by a *bijective* mapping

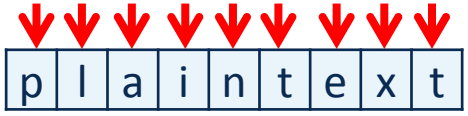
f : $\mathcal{M} \longrightarrow \mathcal{C}$ for Enc and f^{-1} : $\mathcal{C} \longrightarrow \mathcal{M}$ for Dec

Classification

Transposition permute letters according to some scheme



Substitution substitute letters by other letters (or symbols)



A simple Substitution Cipher

Key Generation

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

choose a *shift value* k $\in$ [0, 25]

Auch: ROT-x(m)
-> „Terroristen haben Double ROT-13 benutzt…“ – B. Schneier

Encryption

Let m = $m_0 \ldots m_n$ and
Let #m_i denote the position of m_i in the alphabet.

Enc(k, m) = $c_0 \ldots c_n$ where for each c_i : **$\#c_i = \#m_i + k \pmod{26}$**

Security

How would you break it?
What is the size of the key space?
-> how many random guesses would you need?

Slightly better Substitution Cipher

Use arbitrary permutation:

k:=

A → X
B → C
C → Y
…
Y → V
Z → B

More formally:

Key Generation

choose k as a permutation of the alphabet/an l-tuple of distinct symbols

Encryption

Let $m = m_0 \ldots m_n$.

$Enc(k, m) = c_0 \ldots c_n$ where each $c_i = f(k, m_i)$.

What is the size of the key space in this case?

How would you break it?

Breaking a Substitution Cipher

Brute force:

Assuming a shift cipher, try all 26 possible keys

Assuming permutation: 2^{88} (too large)

More intelligent:

1. Use frequency of letters in the expected language

   „e“:12.7%, „t“: 9.1%, „a“: 8.1%, …

How large is large? (Some context)

Reference Numbers Comparing Relative Magnitudes

Reference	***Magnitude***		
Seconds in a year		$\approx 3 * 10^7$	
Seconds since creation of solar system		$\approx 2 * 10^{17}$	$\approx 4.6 * 10^9$ y
Clock cycles per year (50 MHz computer)		$\approx 1.6 * 10^{15}$	
Instructions per year (i7 @ 3.0 GHz)	$\approx 2^{63}$	$\approx 7.15 * 10^{18}$	
Binary strings of length 64	2^{64}	$\approx 1.8 * 10^{19}$	
Binary strings of length 128	2^{128}	$\approx 3.4 * 10^{38}$	
Binary strings of length 256	2^{256}	$\approx 1.2 * 10^{77}$	
Number of 75-digit prime numbers		$\approx 5.2 * 10^{72}$	
Number of 80-digit prime numbers		$\approx 5.4 * 10^{77}$	
Electrons in the universe		$\approx 8.37 * 10^{77}$	

Attacking Cryptography: Brute Force Attack

Brute force attack, try all keys until intelligible plaintext found:

- Every crypto algorithm (save: OTP) can be attacked by brute force
- On average, half of all possible keys will have to be tried

Average Time Required for Exhaustive Key Search

Key Size [bit]	*Number of keys*	*Time required at 1 encryption / µs*	*Time required at 10^6 encryption / µs*
32	$2^{32} = 4.3 * 10^9$	2^{31} µs = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 * 10^{16}$	2^{55} µs = 1142 years	10.01 hours
128	$2^{128} = 3.4 * 10^{38}$	2^{127} µs = $5.4 * 10^{24}$ years	$5.4 * 10^{18}$ years
(Vigenère 26 chars	$26! = 4 * 10^{26}$	2^{88} µs = $6.4*10^{12}$ years	$6.4*10^6$ years)

i7 could get to around 10^4 encryptions/µs
GPU can perform around $7x10^5$ hashes

Time since human/chimpanzee lines diverged:
$5×10^6$ years,
Homo sapiens: $5×10^4$ years

Perfect Secrecy

Observation: Patterns are your enemy!

Concept:

- Long key (long/no periodicity)
- No recognizable pattern

Gilbert Vernam
(1890-1960)

Key Generation

choose $k = (k_1, \ldots, k_n)$ where each k_i is truly random permutation

Encryption

Let $m = m_0 \ldots m_n$.

$Enc(k, m) = c_0 \ldots c_n$ where $c_i = f(k_i, m_i)$

…actually XOR, not ADD mod 26 in this case…

The One-Time-Pad (Vernam cipher)

Truly random key, as long as the message:

m =	A	T	T	A	C	K	T	H	E	C	I	T	Y	A	T	T	W	E	L	V	E
k =	P	S	P	I	U	H	G	D	S	P	H	G	D	S	P	I	W	E	E	W	O

(+ mod 26)

c =	P	L	I	I	W	R	Z	K	W	R	P	Z	B	S	I	B	S	I	P	R	S

k =	Y	H	P	R	S	R	G	F	F	D	D	X	N	S	Q	I	S	P	W	N	F
m =	R	E	T	R	E	A	T	F	R	O	M	C	O	A	S	T	A	T	T	E	N

(+ mod 26)

… or any other message of the same length, for that matter

Now, what are the two problems with this method?

Security of ciphers

So is the one time pad secure?

What does „secure“ mean, in the first place?

Let‘s assume a CT-only attacker, what is a good requirement?

- *Attacker cannot recover the secret key*
  - *„E(k,m) = m“ !*
- *Attacker cannot recover the complete plaintext*
  - *„E(k,* $m_0 \mid m_1$*)* = $m_0 \mid k \oplus m_1$“!

Information Theoretic Security

Shannon (1949): „CT should not reveal ***any*** information about PT“

Def: A cipher (E,D) over ($\mathcal{K}$, $\mathcal{M}$, $\mathcal{C}$) has ***perfect secrecy*** if

$\forall\ m_0, m_1 \in \mathcal{M}$ (with $len(m_0) = len(m_1)$)
$\forall\ c \in \mathcal{C}$ and $k \xleftarrow{R} \mathcal{K}$:

$$Pr[E(k,m_0) = c] = Pr[E(k,m_1) = c]$$

So being an attacker, what do I learn?

No CT attack can tell if msg is m_0, m_1 (or any other message)

→ No CT only attacks

Security Definitions

Perfect Secrecy

- The ciphertext does not reveal **any** information about the PT
- Caveat: *Key must be random and as long as the message*

Provable Security

- Reduction of construction to some mathematical problem which is known to be hard (then so is breaking the construction)

Semantic Security

- An „efficient“ algorithm cannot find any information in the CT (the CT is polynomially indistinguishable from a CT with PS)

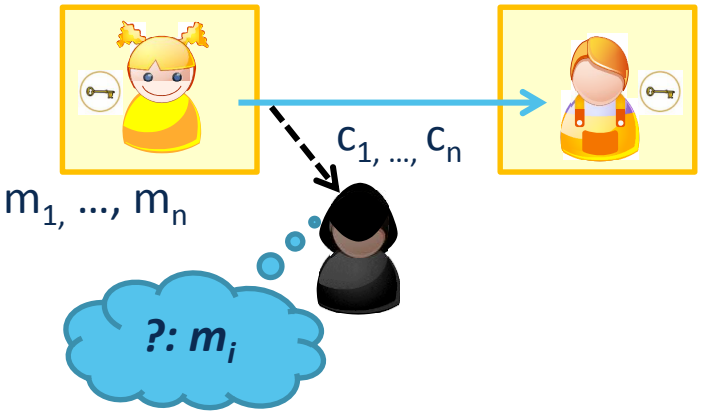
Attacker models revisited: Passive attacks

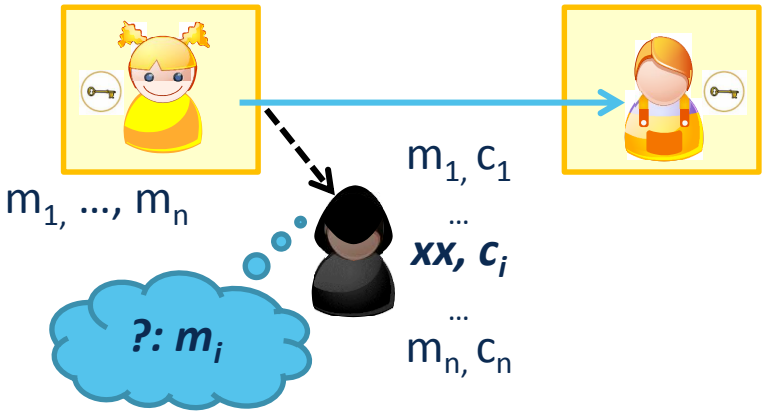
Ciphertext-only attack:

- despite concealed key
- using ciphertext only
- learn about plaintext (or key)

Known-plaintext attack:

- despite concealed key
- Knowing some plaintexts
- Learn about plaintext (or key)



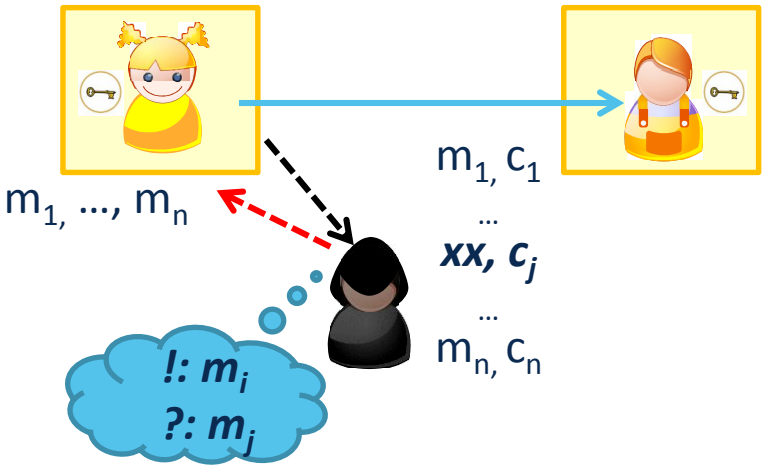


- *Represents weakest attacker!*

Attacker models revisited: Active attacks

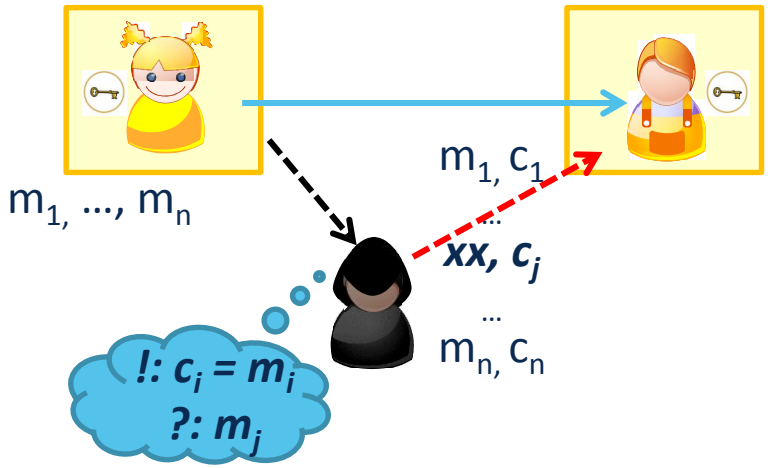
Chosen-plaintext attack:

- despite concealed key
- asking Alice to encrypt m_a
- learn about $\boldsymbol{m_j}$ (or key)



Chosen-ciphertext attack:

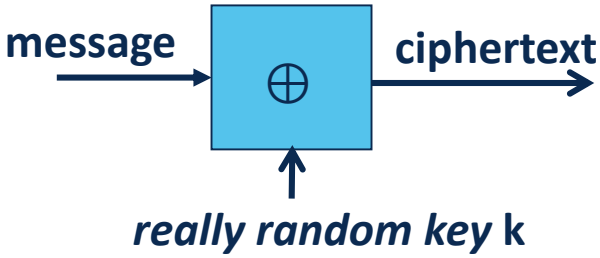
- despite concealed key
- asking Bob to decrypt c_a
- learn about $\boldsymbol{m_j}$ (or key)

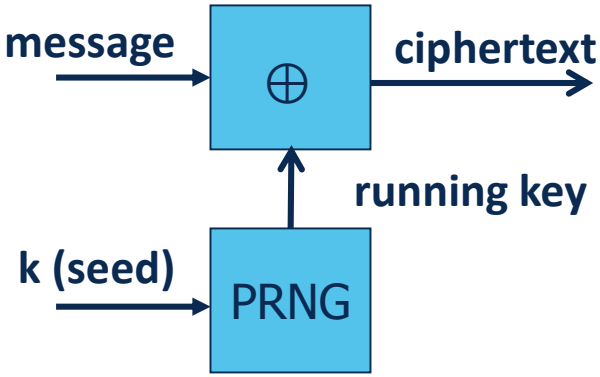


Strongest attacker!

Stream ciphers in general

OTP:





Idea: replace random by „pseudorandom“ key

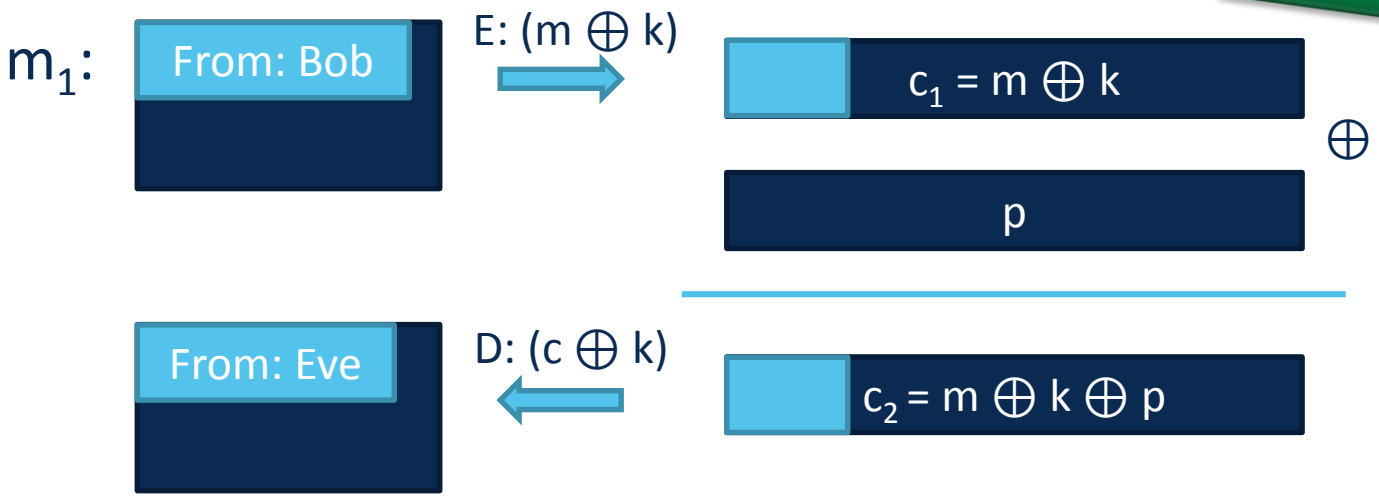
PRNG is a function G: $\{0,1\}^s \rightarrow \{0,1\}^n$ n >> s

Det. algorithm from seed space to key space (looking random)

Lack of integrity (Malleability)

Operation on the cipher has predicted effect on cleartext!
-> Computation on CT!

These stream ciphers are malleable:



In this simple example:

Bob = 42 6F 62; Eve = 45 76 65; Bob ⊕Eve = 07 19 07 (p:= 0 0 0 0 07 19 07)

Lesson: Modification is undetected and has predictable impact!

Teil 2: Block-Chiffren

Erinnerung an Funktionstheorie

Pseudo-Zufallsfunktionen (PRF) und Pseudo-Zufallspermutationen (PRP)

Grundidee der Block-Chiffren

Substitutions-Permutations Netzwerk

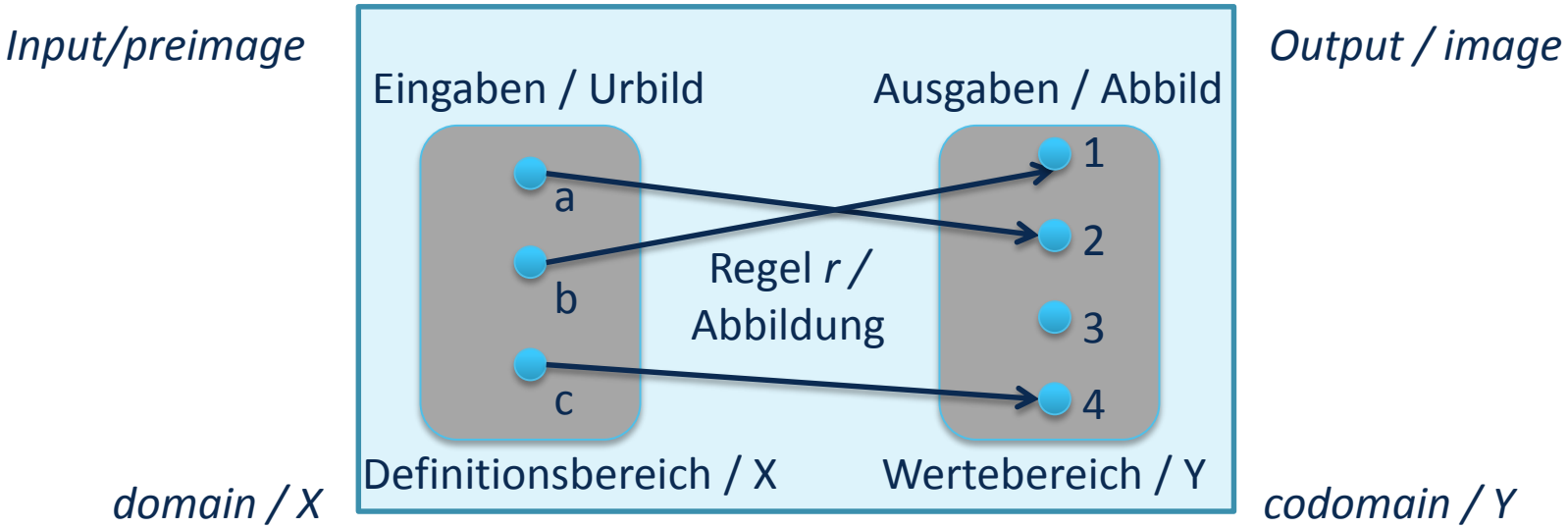
Feistel Netzwerke

Zwei Beispiele: DES und AES

Verschlüsselung langer Sequenzen (Operationsmodi)

ECB, DCM, CBC mit zufälligem IV, OFB, R-CTR

Kleine Erinnerung zu Funktionen



$f: X \longrightarrow Y$

$X = \{a,b,c\}$

$Y=\{1,2,3,4\}$

$y=f(x)$

$Im(f) = \{1,2,4\}$

Pseudo Random Functions and Permutations

Pseudo Random Functions (PRF):

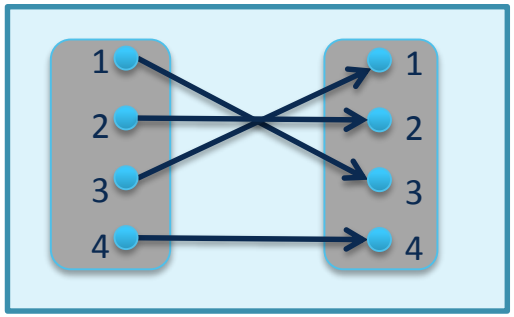
$$F: K \times X \longrightarrow Y$$

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

Permutations:

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

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

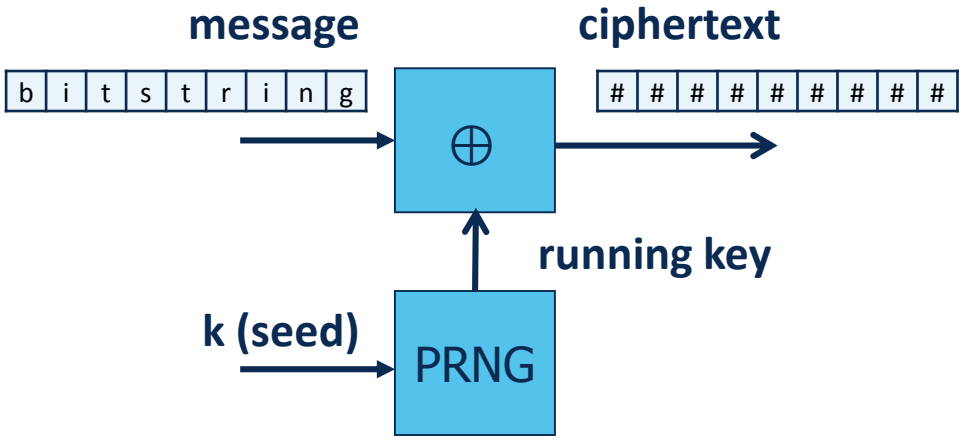


Pseudo Random Permutation (PRP):

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

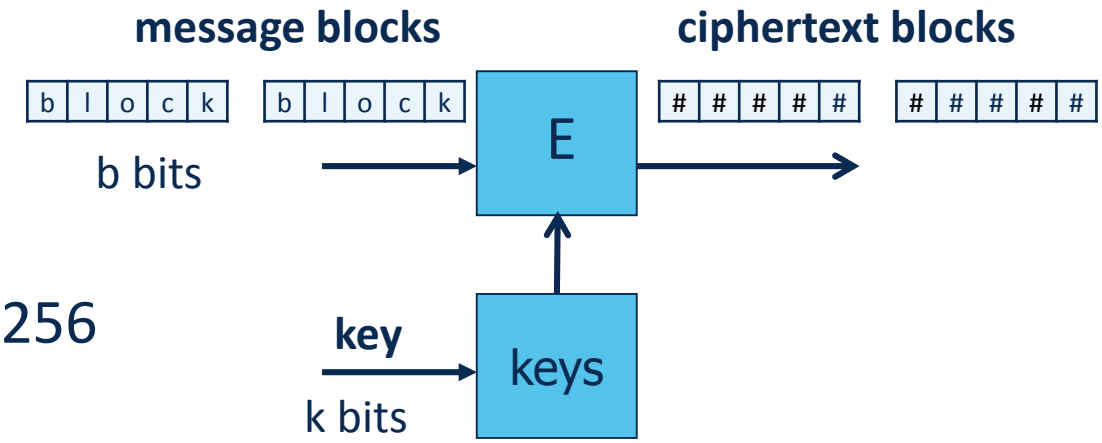
- has efficient deterministic algorithm to evaluate $E(k,x)$ ***and***
- efficient inversion algorithm $D = E^{-1}$

Stream Ciphers and Block Ciphers



Goal:

Build a secure PRP for b-bit blocks



Examples:

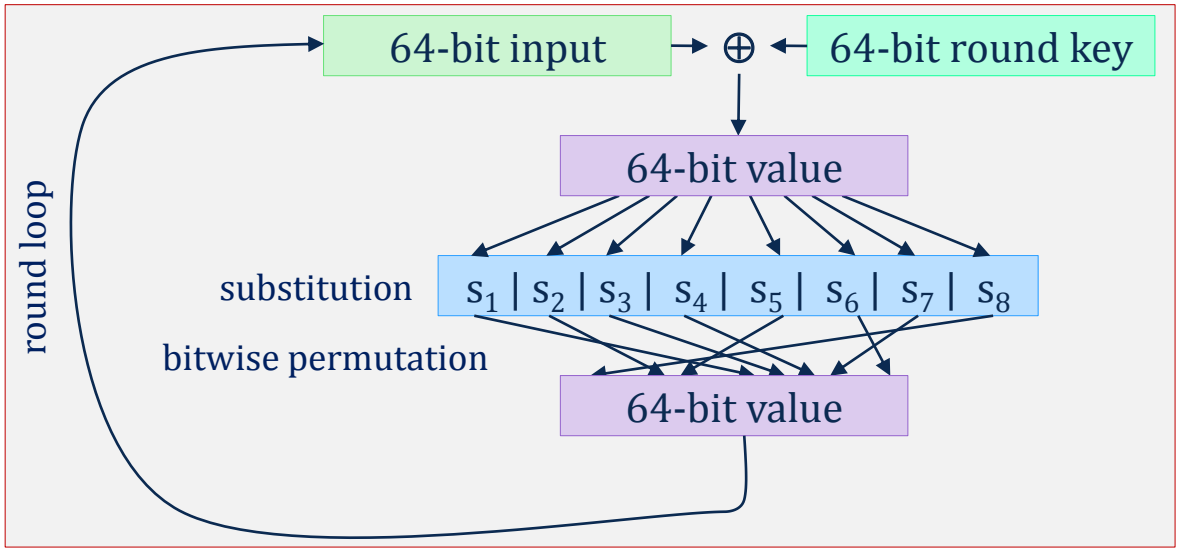
3DES: n = 64, k = 168

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

Interlude: Substitution – Permutation Nets

SPN implement the Confusion – Diffusion Paradigm:

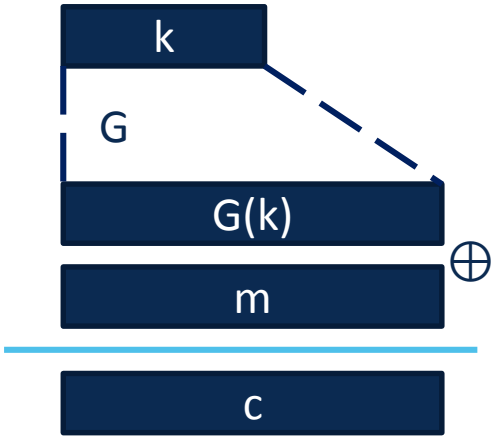
- Round keys k_i are derived from k, then usually $\oplus$-ed with intermediate round output
- round functions f_i are *fixed, invertible* substitution boxes (S-Box)



Rounds and Round Keys: Key Expansion

Recall from stream ciphers:

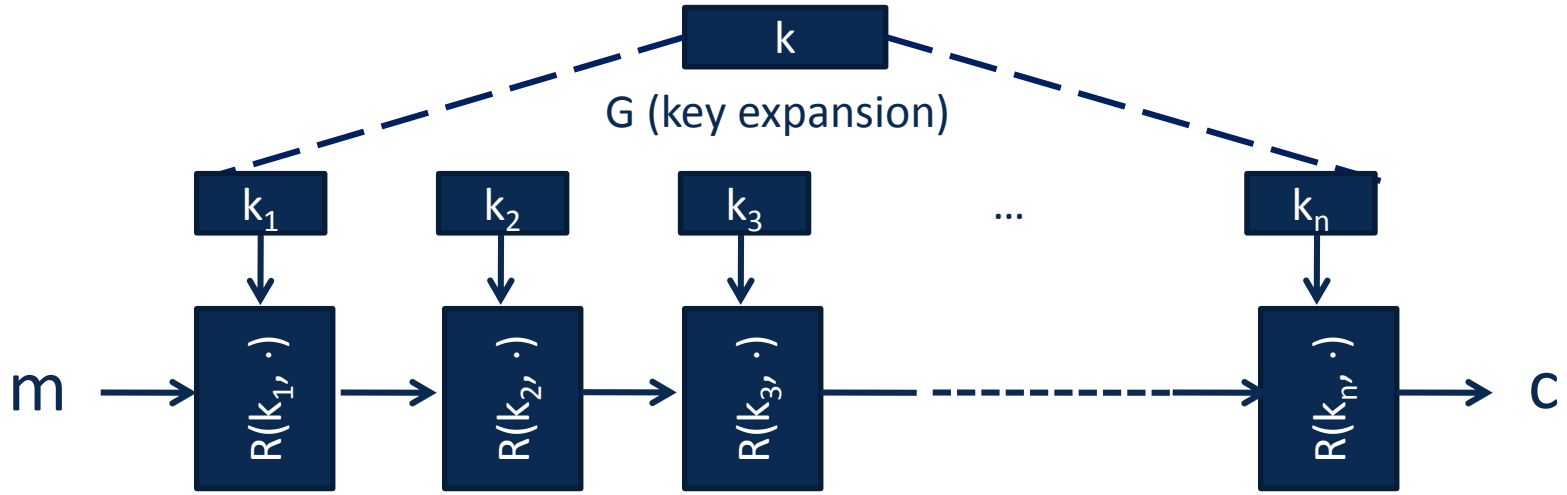
Short key expanded to encrypt bitstream



Idea:

Perform several keyed permutations in rounds

Expand key to round keys as parameters for random permutations



Feistel Networks / DES

Horst Feistel

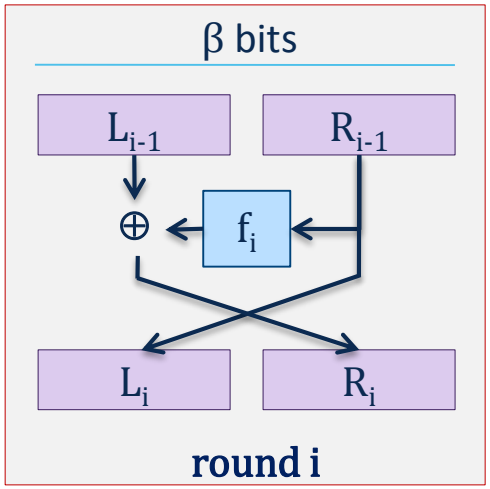
Goal:

Create a PRP from arbitrary (non-invertible) functions

Idea:

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

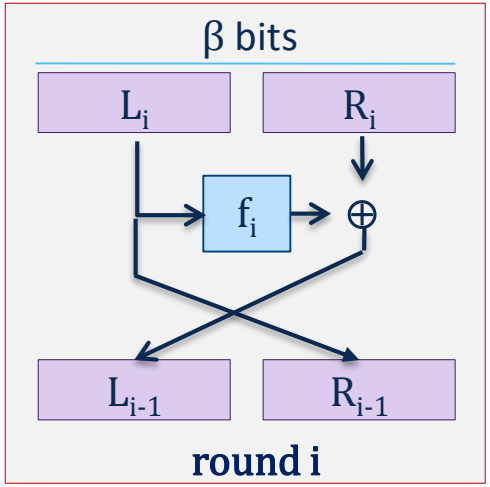
with round function f_i (possibly non-invertible),
keyed with round key k_i



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

$R_{i-1} = L_i$

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



Lucifer (DES):

56bit keys, 16 rounds standardized by NIST as DES

DES is broken (22h for total break in 1999)

Extension to 3DES: $E(k_1, D(k_2, E(k_3, m)))$

The Advanced Encryption Standard

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

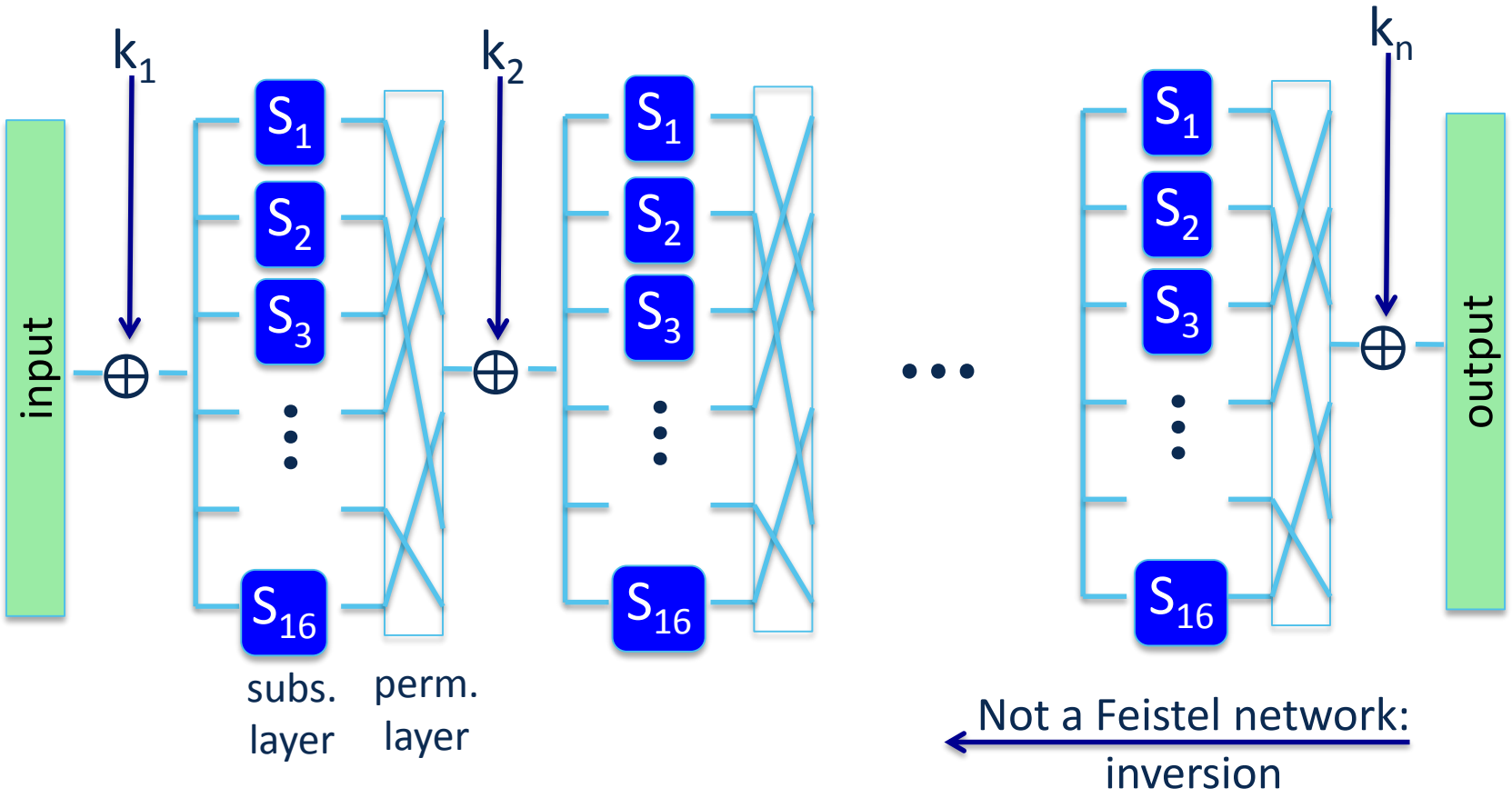
Key sizes: 128, 192, 256 bits Block size: 128 bits

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

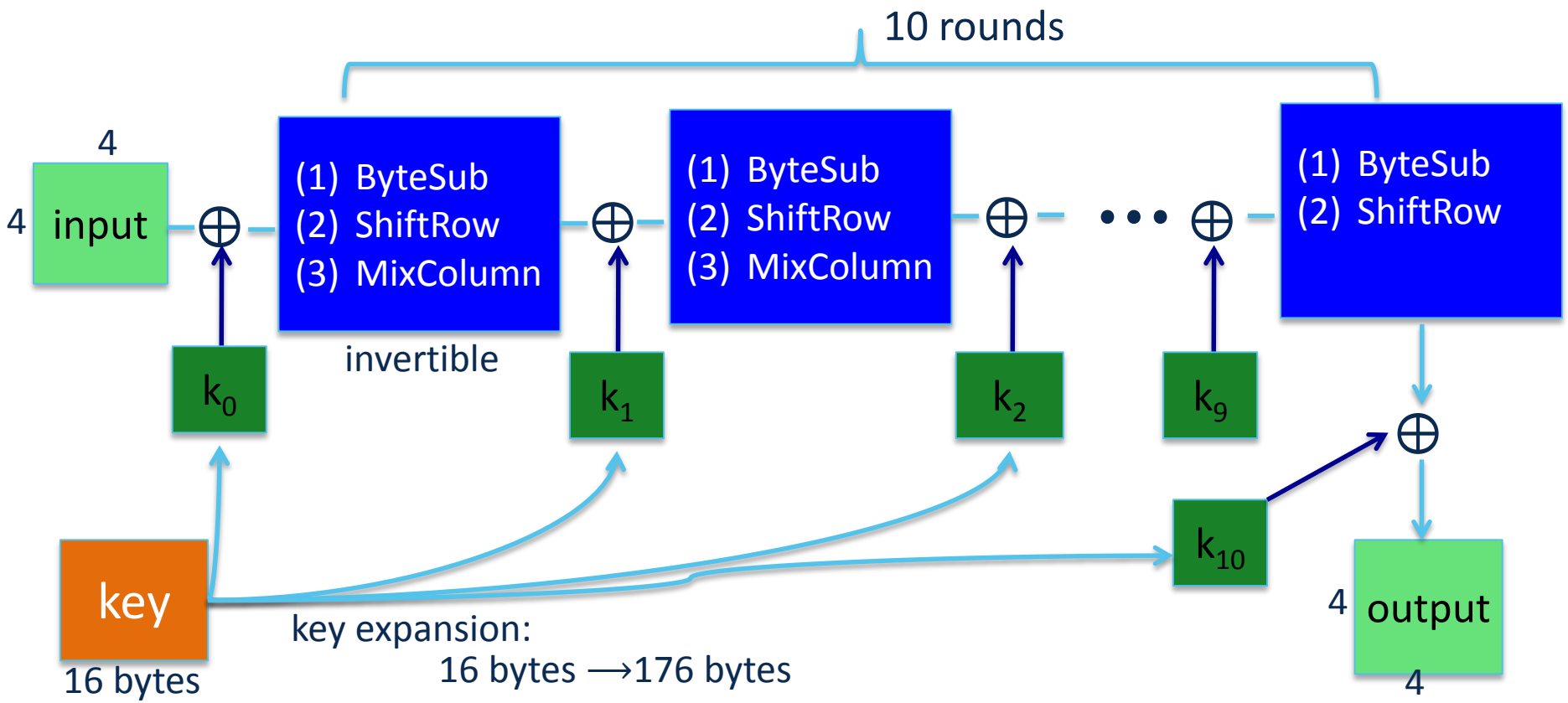
AES Substitution-Permutation Network

1997: NIST commissions „Advanced Encryption Standard“ (AES)

2000: NIST chooses Belgian proposal Rijndael as AES:



AES-128 scheme



Building Block Ciphers (Modes of Operation)

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

… with fixed input sizes (64 or 128 bits)

Your average message will usually be longer than 128 bits…

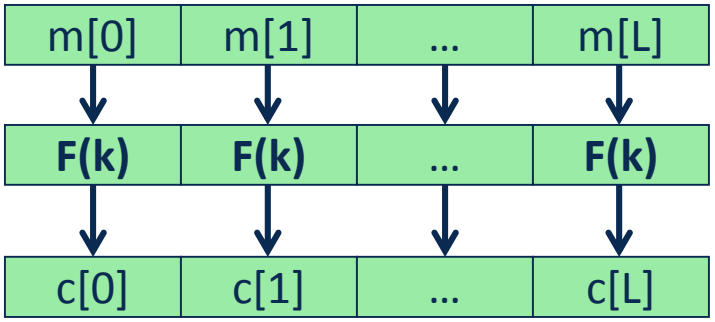
Goal:

Build „secure“ encryption from secure PRPs

Electronic Code Book Mode

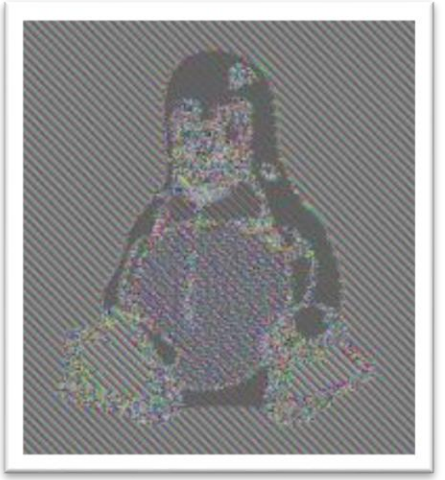
No semantic security!

Encrypt each block with the keyed PRP:



ECB encryption is *deterministic*

⇒ identical PT is encrypted to identical CT:



Is this "secure" (how)?

Consider: Eve can send two messages and needs to distinguish which one has been encrypted…

Making operation modes more secure

ECB of a deterministic PRP is broken:

- Two messages encrypted with the same key yield identical CT
- Even two identical PT blocks are encrypted to identical CT blocks

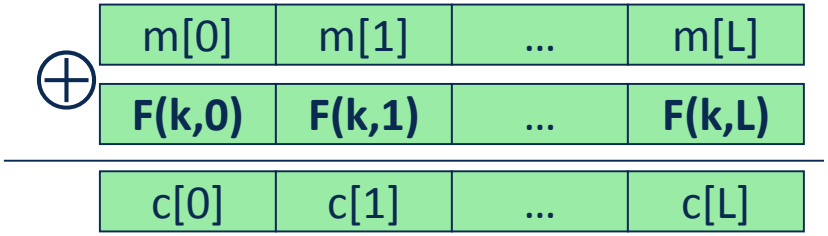
What can we do about this?

- One-time key (internal): encrypt each block differently
- Many-time key (external): introduce randomness

Counter Modes and Chaining

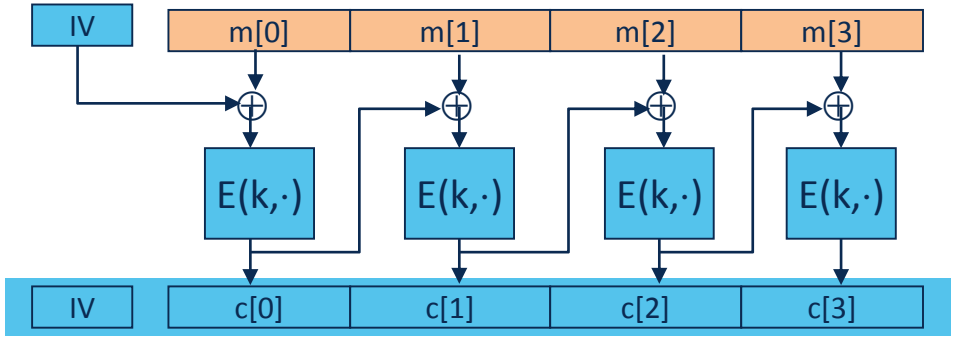
(1) Encrypt each block differently:

- Integrate some (changing) value into the encryption of blocks
  - Use Nonces: $c_i = E(k,n_i,m_i) = E(k, (n_i,m_i))$ or $E((k,n_i),m_i)$?
  - …and transmit Nonces n_i?
  - Introduce counters!



(2) Many-time key (introduce randomness):

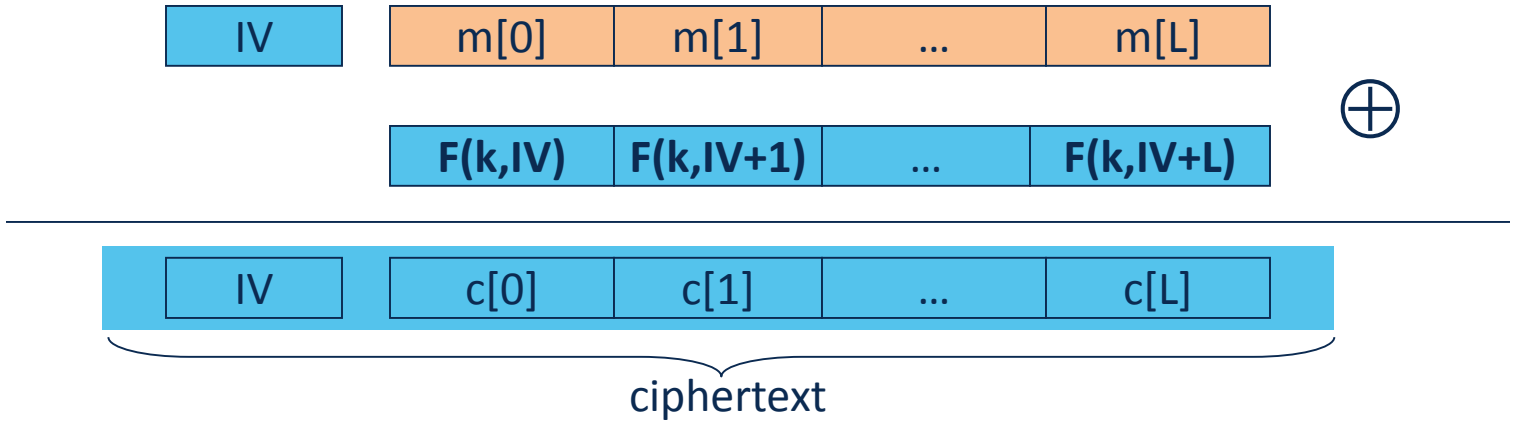
- Integrate some (changing) value into the encryption of blocks!
  - Independent randomness for each block (do we need this)?
  - Choose random initial IV
  - Chain/feedback



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, to avoid repetition

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

Zusammenfassung

Sie kennen Kryptologie, Kryptographie und Kryptoanalyse

Sie wissen was Strom- und Block-Chiffren sind

Sie verstehen die unterschiedlichen Angreifermodelle

Sie kennen perfekte und semantische Sicherheit und Sie wissen, wie diese nachgewiesen werden

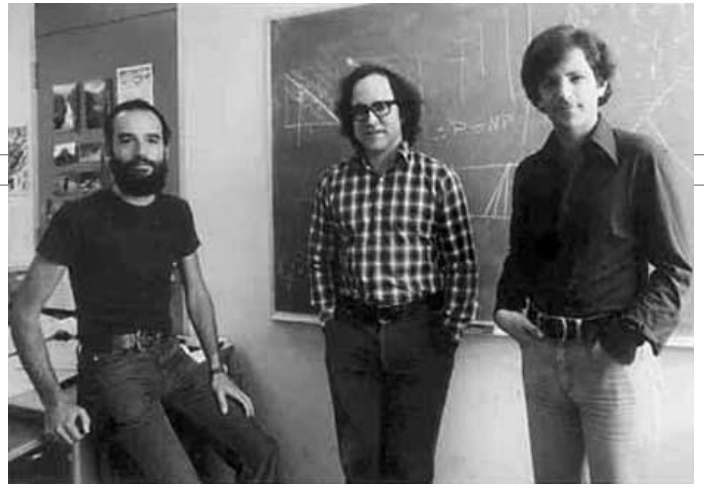
Sie kennen das One-Time-Pad

Sie können Feistel-Netzwerke und 3DES erklären

Sie kennen und verstehen AES

Sie kennen unterschiedliche Operationsmodi und kennen ihre Eigenschaften

Teil 3: Asymmetrie…



Merkle-Puzzles/Diffie-Hellman:

- Vereinbarung von (symm.) Schlüsseln

Ziel: Direkt asymmetrisch verschlüsseln
(nicht nur Schlüssel etablieren)

Einfachste Idee mit DH (quasi El-Gamal):

- *Öffentlicher Schlüssel von Bob: g^b privater Schlüssel: b*
- *Alice sendet: $E(g^{ab},m)$, g^a*

Aber historisch:

Ronald L. Rivest, Adi Shamir, Leonhard M. Adleman: *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.* Communications of the ACM, vol. 21, no. 2, 1978, 120-126.

RSA – Mathematische Grundlagen

Bei freier Wahl großer Primzahlen *p* und *q:*

- Die Berechnung von $n = p \cdot q$ leicht
- Faktorisierung von *n* zu *p* und *q* schwierig

Mit Wissen der Primfaktoren und erweitertem euklidischen Algorithmus ist einfach zu berechnen:

Für multiplikative zykl. Gruppe Z_n^* und $\boldsymbol{e}$ (teilerfremd zu *n*), gilt $\boldsymbol{e^{-1}}$:

$$ggT(e,n) = 1: d \cdot e + k \cdot n$$

$$k \cdot n \equiv 0 \bmod n$$

$$e \cdot e^{-1} \equiv 1 \bmod n$$

$$\Rightarrow \quad e^{-1} = d$$

RSA - Schlüsselgenerierung

Jeder Teilnehmer

- wählt zufällig und unabhängig 2 verschiedene Primzahlen p, q ungefähr gleicher Länge
- berechnet $n = p \cdot q$ und $\varphi(n) = n\text{-}p\text{-}q+1 = (p\text{-}1)(q\text{-}1)$
- wählt zufällige Zahlen e,d mit $2 < e < \varphi(n)$, $ggT(e, \varphi(n)) = 1$
- Und $e \cdot d = 1 \bmod (\varphi(n))$ (mit erweitertem euklidischen Algorithmus)

Öffentlicher Schlüssel: (n, e)

Geheimer Schlüssel: (p, q, d)

Ver- und Entschlüsselung

Verschlüsselung:

Gegeben (N,e), RSA (m):

$= m^e \mod N$

Entschlüsselung:

Gegeben (p,q,d), RSA^{-1} (c):

„$c^{-1/e} \mod N$“ $= c^d \mod N$

„$m^{e^{-1/e}} \mod N$“ $= m$

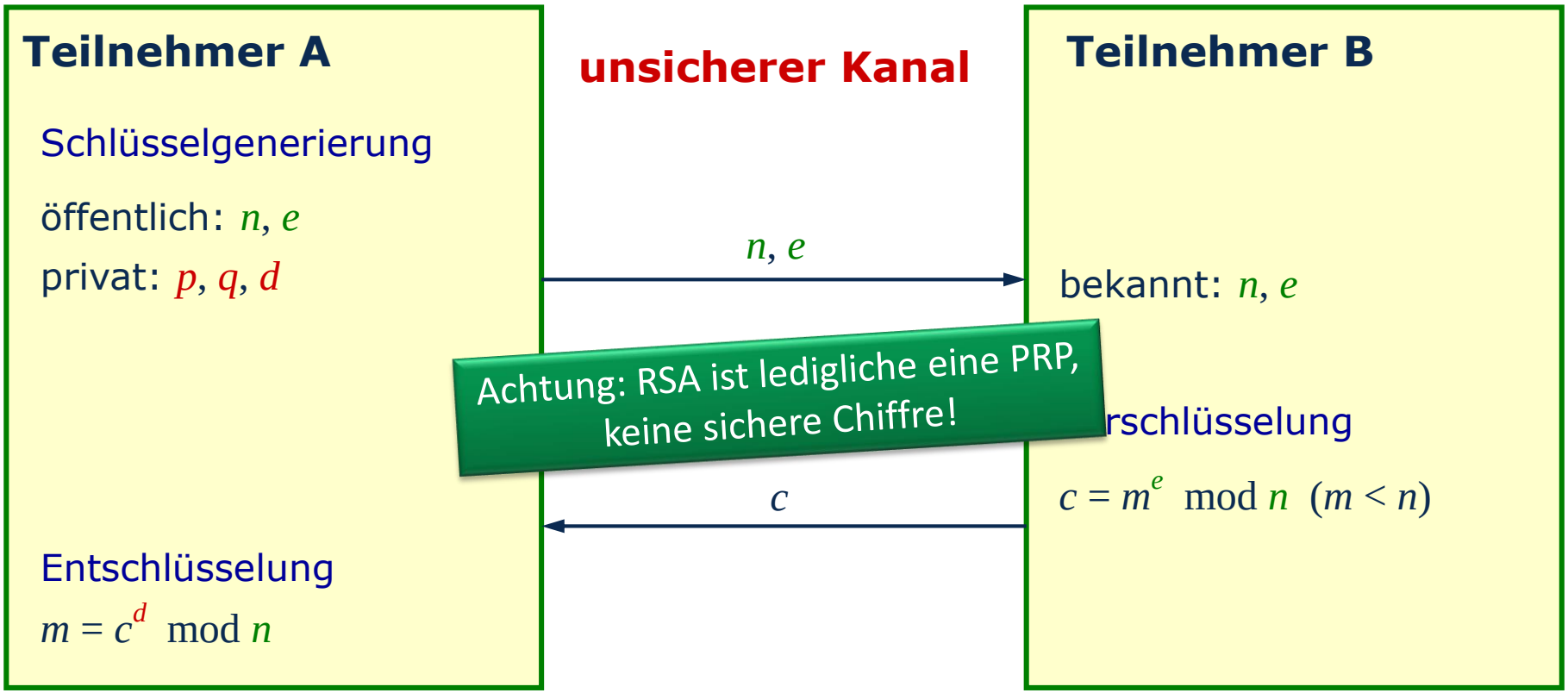
Bonus, Signieren einer Nachricht:

Gegeben sk (p,q,d):

$tag = RSA^{-1}(pk, h(m)) = RSA(d,h(m))$

Beispiel RSA

RSA als Verschlüsselungssystem (unsicher, kommen Sie zu SaC!)



Hybride Verschlüsselung

secret key (session key)

$k_{A,B}$

Public key

$k_{e,B}$

Private key

$k_{d,B}$

E

c_1

D

$k_{A,B} = D(k_{d,B}, c_1)$

Message

m

E

c_2

D

Message

$m = D(k_{A,B}, c_2)$

c_1, c_2

$c_1 = E(k_{e,B}, k_{A,B}), c_2 = E(k_{A,B}, m)$

Zusammenfassung

Sie haben Sicherheit, Safety, Datenschutz kennengelernt

Sie kennen die Ziele von Sicherheit und Wege zur Minimierung operationeller Risiken

Sie kennen den Ablauf einer Sicherheitsanalyse und können eine Bedrohungs-, Schwachstellen- und Risikoanalyse durchführen

Sie kennen die Sicherheitsziele CIA und wissen, dass Sie die Annahmen klar festlegen müssen (wie definieren Sie ein Angreifermodell?)

Sie kennen einfache Angriffsmethoden und deren Auswirkung

Sie wissen, wozu Authentisierung gut ist und wie sie funktioniert

Sie kennen Schlüssel und typische Austauschverfahren

Sie wissen was Kryptologie, Kryptographie und Kryptanalyse ist

Sie kennen einige historische Chiffren, das One-Time Pad und Strom-Chiffren

Sie wissen wie Block-Chiffren funktionieren, was Operationsmodi sind und worauf hier zu achten ist!

Sie kennen asymmetrische Verfahren und wissen, wie diese einzusetzen sind.

References

Dan Boneh, „Introduction to Cryptography“, course materials, Stanford University, 2016

Stefan Katzenbeisser, „Einführung in Trusted Systems“, teaching materials, TU Darmstadt, 2015

Mark Manulis, „Introduction to Cryptography“, teaching materials, TU Darmstadt, 2011

Günter Schäfer, „Netzwerksicherheit“, course materials, TU Ilmenau, 2013

Elke Franz, „Datensicherheit“, teaching materials, TU Dresden, 2014