Introduction to Cryptography

(Lecture 1: Admin, introduction, historical ciphers)

Georg Fuchsbauer

TUW 2025W

Administrativia

Admin

- Lectures + exercises

Lectures Thursday, 12:00 – 14:00 FAV1

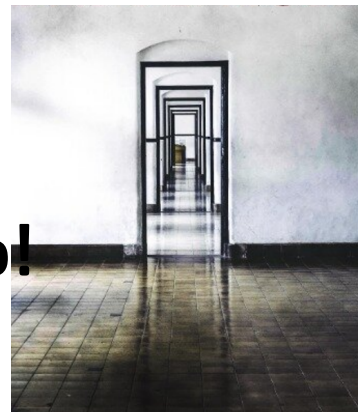
- presence not mandatory, but encouraged
- recorded, slides and recordings on TUWEL

Exercises

- TAs: Fabian Regen, Marek Sefranek, Andreas Weninger, Stefano Trevisani
- 4 groups:
  - Thursday, 10:00 – 12:00, FAV3
  - Thursday, 16:00 – 18:00, FAV3
  - Friday, 9:00 – 11:00, EI 1
  - Friday, 11:00 – 13:00, EI 1

Admin

Exercises



- 1st sheet: on TUWEL, to be done by 9 Oct

1st exercise mandatory to register for a group!

- Group registration opens next week (TUWEL)
- from 16 Oct: in-presence exercise sessions
- In total: 9 exercise sheets (see TISS for exercise dates)

Exercise mode

- Presence is mandatory (if you cannot attend $\rightarrow$ email TA)
- Tick solved (sub)problems
- Upload pdf (1 subproblem per sheet) on TUWEL
- Use TUWEL forum for questions
- In session: a student who ticked will be asked to present

Admin

Exercise grading

- Presentations:
  - Students explain their solutions
  - If a student cannot explain (or obviously did not write) their solution, the points for the sheet will be withdrawn.
  - If this happens a 2nd time, the student fails the course.
- To pass, you need to tick at least 50% of all exercises.
- You must not use AI tools to solve exercises. (Contact us if you need an exception.)

Admin

Exams

- If at least 50% of first 5 sheets solved

→ **Midterm exam** ("closed book")

3 Dec, 14:00 – 16:00 (Audi. Max.)

- If at least 50% of all exercises solved

→ **Final exam** ("closed book")

30 Jan, 10:00 – 12:00 (Audi. Max.)

Retake exam ("closed book")

- You can retake **one** of the two exams (which will "overwrite" the result of the retaken exam)

27 Feb, 12:00 – 14:00 (Audi. Max.)

Admin

Grading

- For a positive grade,
  you require at least 50% **on average** on two exams

- Total points:

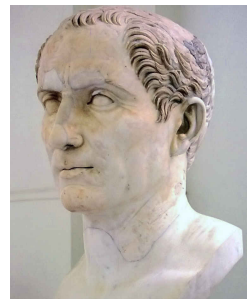
  50% *exercise percentage*
  $+ \ 50\%$ *exam percentage*

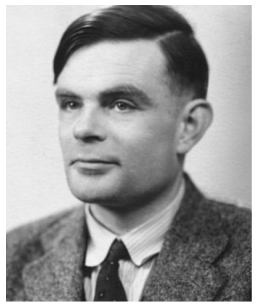
Total points	Grade
Some requirement not fulfilled	5
$\geq 50.0\%$	4
$\geq 62.5\%$	3
$\geq 75.0\%$	2
$\geq 87.5\%$	1

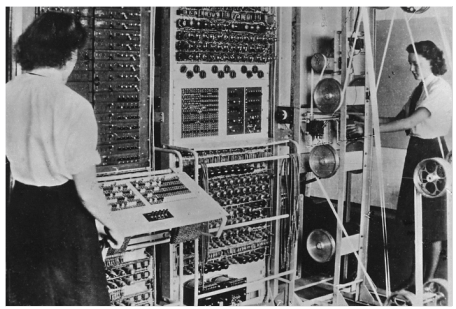
Cryptography

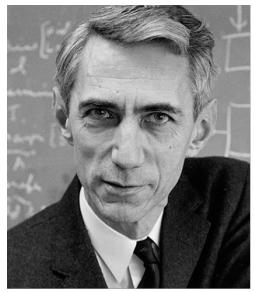
History

- Encryption of messages since ancient times
- Until 1970s: military, governments
- 1975: DES



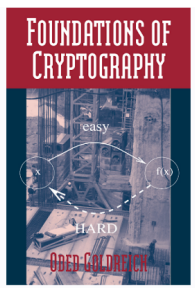






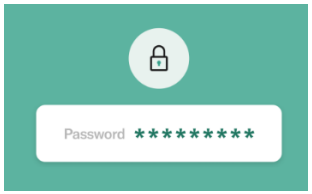
- 20th century: rigorous approach

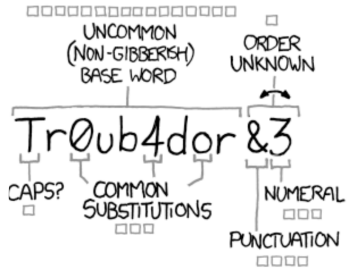
Cryptography is now a science



Modern cryptography

– *authentication*, integrity





Private-key encryption Secret communication between parties that share a secret ("key")

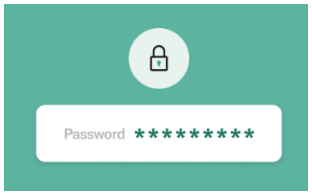
$\longrightarrow$ – *public-key* cryptography

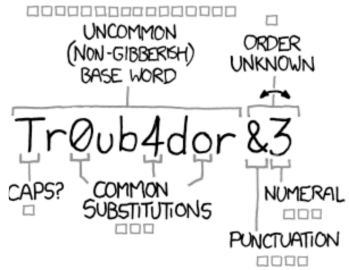
– e-cash, "cryptocurrencies"

– IBE, MPC, FHE, ZKP, .

Modern cryptography

— *authentication*, integrity





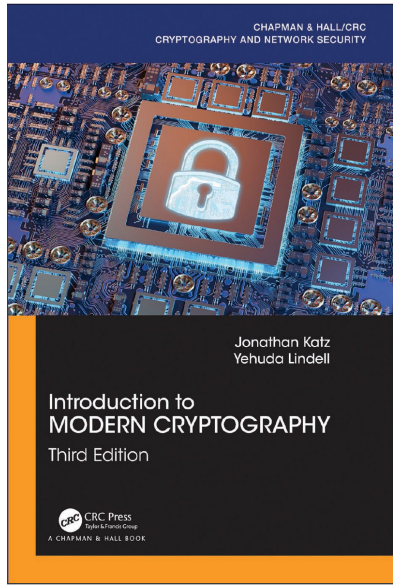
P

e

> Cryptography... deals with mechanisms for ensuring integrity, techniques for exchanging secret keys, protocols for authenticating users, electronic auctions and elections, digital cash, and more. Without attempting to provide a complete characterization, we would say that modern cryptography involves *the study of mathematical techniques for securing digital information, systems, and distributed computations against adversarial attacks.*
>
> [§1.1]

— IBE, MPC, FHE, ZKP, . 

Overview



Reference book:

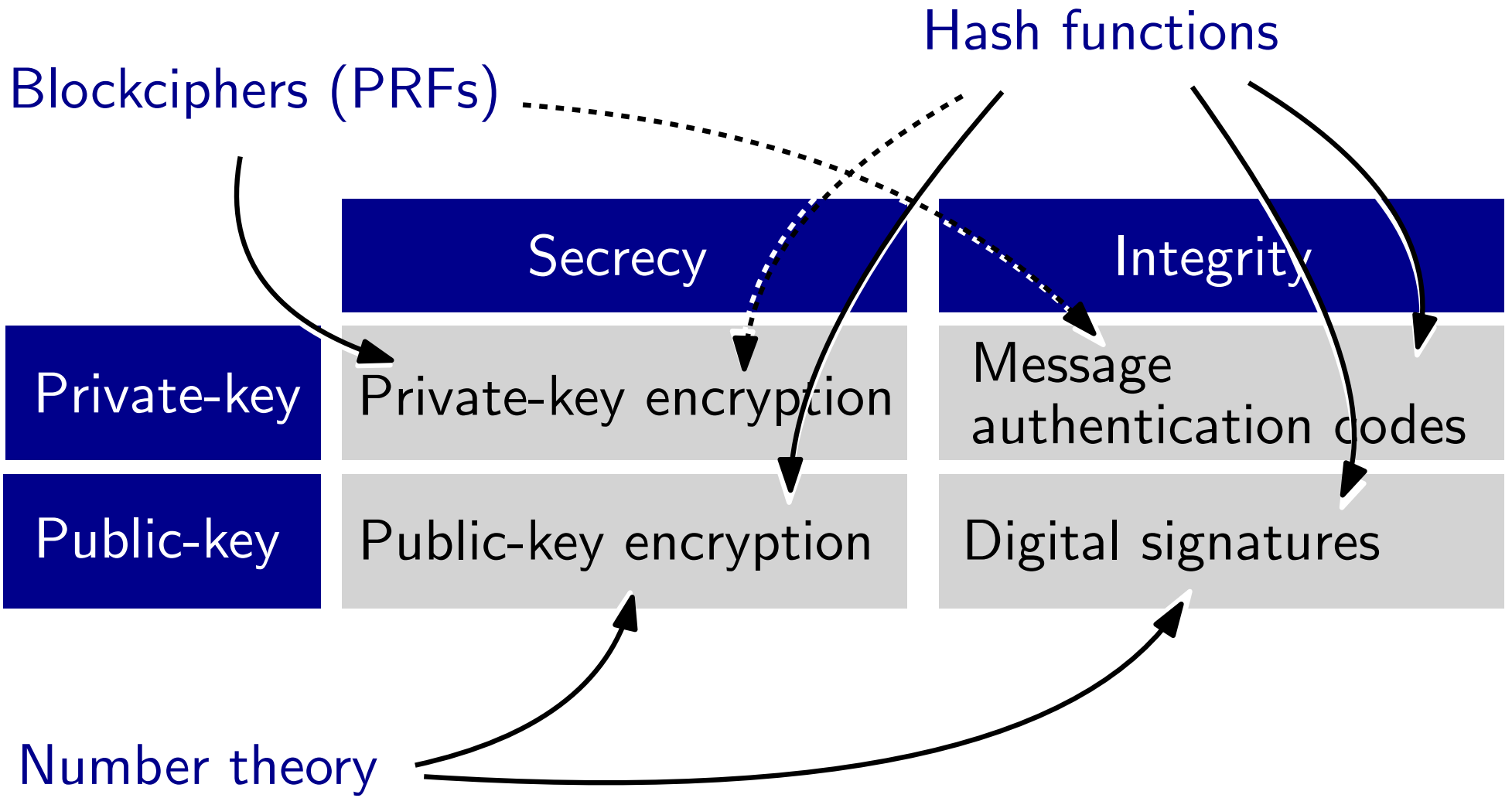
Katz, Lindell: *Introduction to Modern Cryptography* (3rd edition)

This lecture series:

- History, one-time pad
- Blockciphers (PRFs): DES, AES
- Computational security, proofs by reduction
- Private-key encryption, MACs
- Hash functions
- Number theory, public-key encryption
- Digital signatures

§

Topics

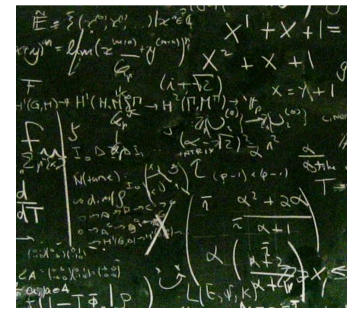

Hash functions
Blockciphers (PRFs)
Secrecy
Integrity
Private-key
Private-key encryption
Message authentication codes
Public-key
Public-key encryption
Digital signatures
Number theory

Goals

Understand cryptography

as used in the real world

(schemes that are used)



taking a rigorous approach

("provable security" paradigm)

Use cryptography

- which primitives are there
- what precise security guarantees do they provide

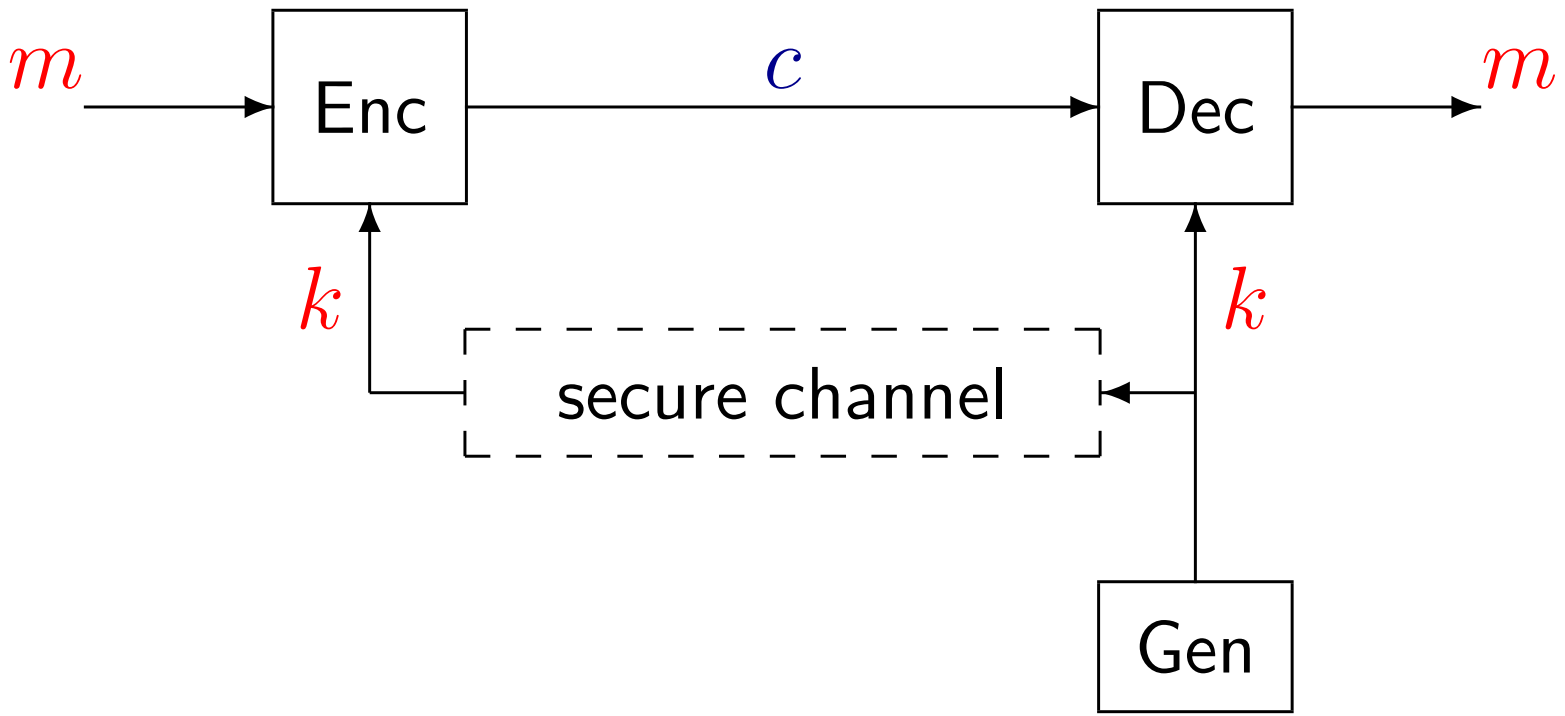
Private-key Encryption

Private-key encryption

("Symmetric encryption")

Alice

Bob



Private-key encryption

Definition (§1.2). A **private-key encryption scheme** is defined by a *message space* $\mathcal{M}$ and three algorithms:

Gen (key-generation) is probabilistic and outputs key k

Enc (encryption) takes key k and message (plaintext) m and outputs ciphertext c

$$c \leftarrow \mathsf{Enc}_k(m)$$

Dec (decryption) takes key k and ciphertext c and outputs m

$$m := \mathsf{Dec}_k(c)$$

Correctness:

For all $k \leftarrow \mathsf{Gen}$ and all $m \in \mathcal{M}$: $\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m$

Kerckhoffs' Principle

Kerckhoffs' Principle: (1883)

Enc and Dec are public and the secrecy of m,
given c, depends entirely on the secrecy of k

[The system] should not require secrecy, and
it should not be a problem if it falls into enemy hands

Arguments:

- Easier to keep keys secret
  (algorithms will not stay secret – reverse engineering. . .)
- Easier to replace key than scheme
- Schemes can be standardized
  - compatibility
  - public scrutiny

Historical Ciphers

Ancient ciphers

- Scytale (Spartans)
  key = "circumference"

```
    |   |   |   |   |   |   |
    | a | m | h | u | r |   |
|---| t | b | a | d | l |---|
|   | y | h | e | l | p |
|   |   |   |   |   |   |
```

$k = 3$

$\rightarrow$ `atymbhhaeudlrlp`

Ancient ciphers

- Scytale (Spartans)
  key = "circumference"

```
    |   |   |   |   |   |
    | a | m | h | u |   |
|---| r | t | b | a |---|
|   | d | l | y | h |
|   | e | l | p |   |
|   |   |   |   |   |
```

$k = 4$

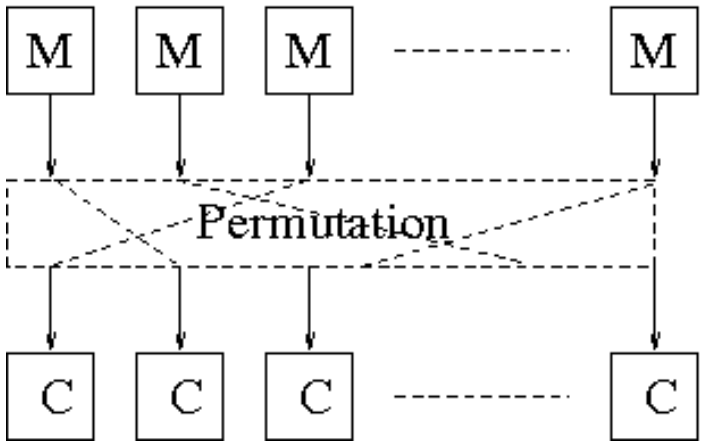
$\rightarrow$ `ardemtllhbypuah`

(only certain permutations possible)

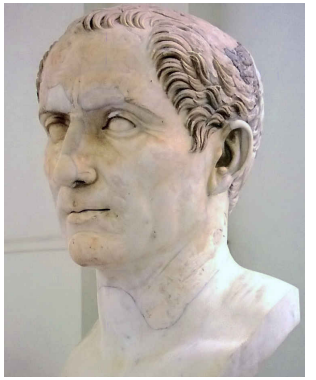
Ancient ciphers

- Scytale (Spartans)
  key = "circumference"

- **Permutation**:
  The key defines a "rearranging" of the message



Ancient ciphers



- Monoalphabetic **substitution** (e.g. Caesar cipher, ROT13)

  $\Sigma = \{A, \ldots, Z\}$

  Key: bijection $\pi : \Sigma \to \Sigma$

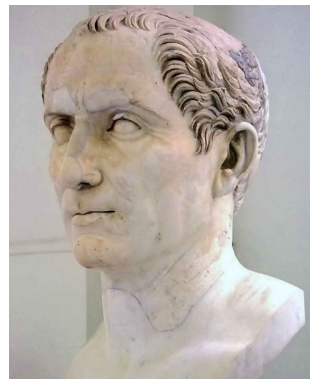
  $\mathsf{Enc}(\pi, m_1 m_2 \ldots m_n)$
  $= \pi(m_1)\pi(m_2)\ldots\pi(m_n)$

  $\mathsf{Dec}(\pi, c_1 c_2 \ldots c_n)$
  $= \pi^{-1}(c_1)\pi^{-1}(c_2)\ldots\pi^{-1}(c_n)$

π:

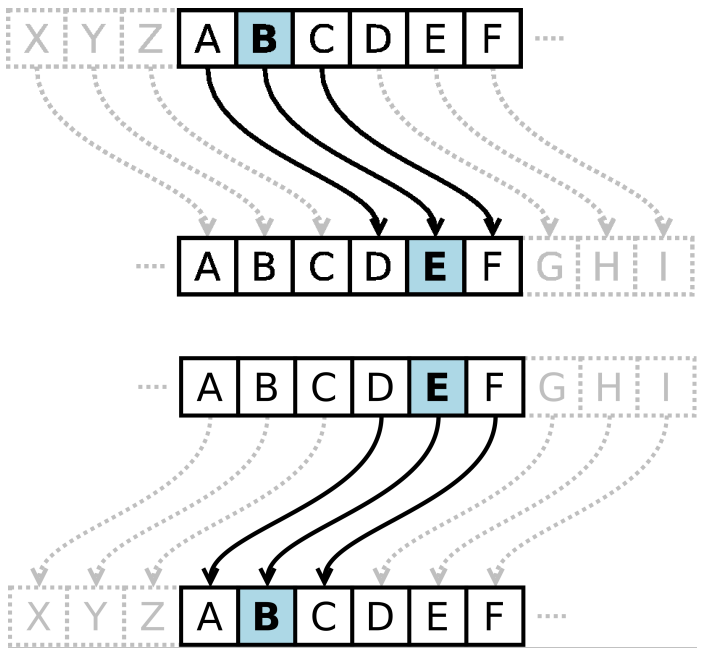
A	$\to$	K
B	$\to$	X
C	$\to$	M
D	$\to$	Q
	$\vdots$	

Ancient ciphers



- **Shift ciphers**
  ($\subseteq$ substitution ciphers)

  Caesar cipher



A ~ 0
B ~ 1
$\vdots$
Z ~ 25

key: $k \in \{0, \ldots, 25\}$

Caesar: $k = 3$

$$\mathsf{Enc}(k, m) := (m + k) \bmod 26$$
$$\mathsf{Dec}(k, c) := (c - k) \bmod 26$$

Cryptanalysis

Breaking shift ciphers

- how many possible keys?
- only 26 (!)
- given c, try all keys k,

  which $m = \mathrm{Dec}(k, c)$ makes sense?

"Brute force" attack (or *exhaustive search*)

$\Rightarrow$ *key space must be large enough*

Examples:

- Key length for Mifare Crypto-1: 48 bits
- 2^{48} Mifare Crypto-1 (breakable on phone)
- 2^{56} DES (. . . laptop)
- Key size of the AES: 2^{128} or 2^{256}

Historical ciphers

Breaking monoalphabetic **substitution** (not just shifts!)

- every letter replaced by other (arbitrary) letter
- how many keys? $26!$ $(> 2^{88})$ (# of permutations)

Cryptanalysis: **statistical analysis** ($\Leftarrow$ frequency of letters)

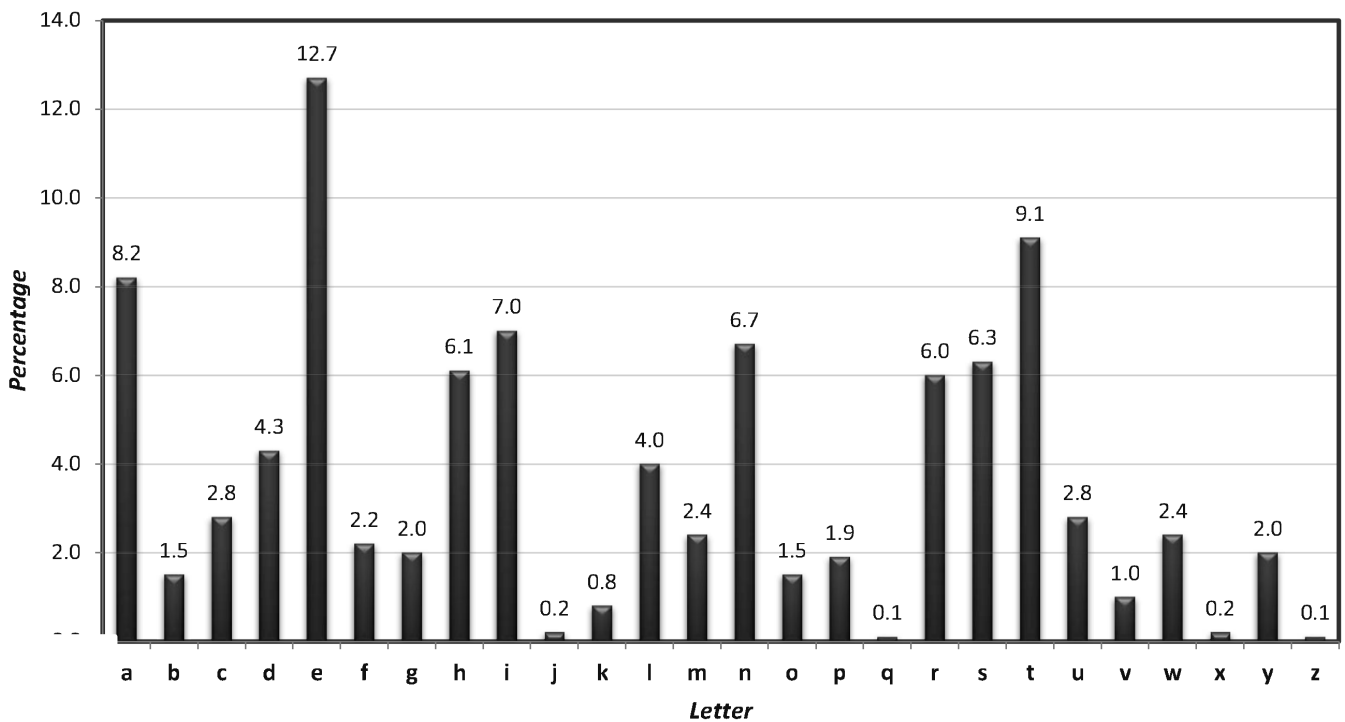


Fig. 1.3

Historical ciphers

- **Polyalphabetic** substitution
  - key: $k = (\pi_1, \ldots, \pi_\ell)$
  - $\mathsf{Enc}(k, m_1 m_2 \ldots m_n)$
    $= \pi_1(m_1)\pi_2(m_2)\ldots\pi_\ell(m_\ell)\pi_1(m_{\ell+1})\ldots$

π_1:

A	$\rightarrow$	K
B	$\rightarrow$	L
C	$\rightarrow$	M
D	$\rightarrow$	N
	$\vdots$	

π_2:

A	$\rightarrow$	I
B	$\rightarrow$	K
C	$\rightarrow$	P
D	$\rightarrow$	W
	$\vdots$	

π_3:

A	$\rightarrow$	J
B	$\rightarrow$	L
C	$\rightarrow$	E
D	$\rightarrow$	Y
	$\vdots$	

Historical ciphers

- **Polyalphabetic** substitution
  - key: $k = (\pi_1, \ldots, \pi_\ell)$
  - $\mathrm{Enc}(k, m_1 m_2 \ldots m_n)$
    $= \pi_1(m_1)\pi_2(m_2)\ldots\pi_\ell(m_\ell)\pi_1(m_{\ell+1})\ldots$

Vigenère Cipher (1553): (⊆ polyalphab. substitution)

M	E	S	S	A	G	E	C	L	A	I	R
C	L	E	C	L	E	C	L	E	C	L	E
O	P	W	U	L	K	G	N	P	C	T	V

12	04	18	18	00	06	04	02	11	00	08	17
02	11	04	02	11	04	02	11	04	02	11	04
14	15	22	20	11	10	06	13	15	02	19	21

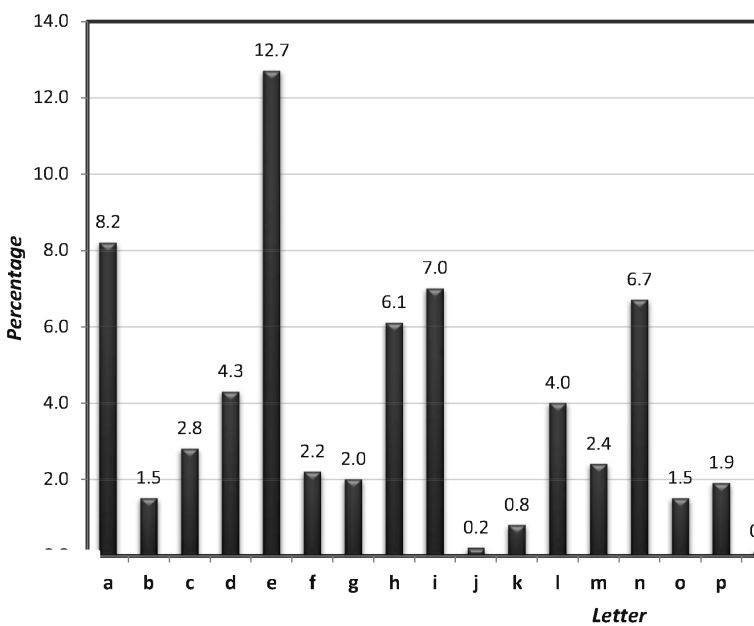
A ↔ 0
⋮
Z ↔ 25

Historical ciphers

- **Polyalphabetic** substitution
  - key: $k = (\pi_1, \ldots, \pi_\ell)$
  - $\mathrm{Enc}(k, m_1 m_2 \ldots m_n)$
    $= \pi_1(m_1)\pi_2(m_2)\ldots\pi_\ell(m_\ell)\pi_1(m_{\ell+1})\ldots$

Vigenère Cipher (1553): ($\subseteq$ polyalphab. substitution)

M	E	S	S	A	G	E	C	L	A	I	R
C	L	E	C	L	E	C	L	E	C	L	E
O	P	W	U	L	K	G	N	P	C	T	V
12	04	18	18	00	06	04	02	11	00	08	17
02	11	04	02	11	04	02	11	04	02	11	04
14	15	22	20	11	10	06	13	15	02	19	21

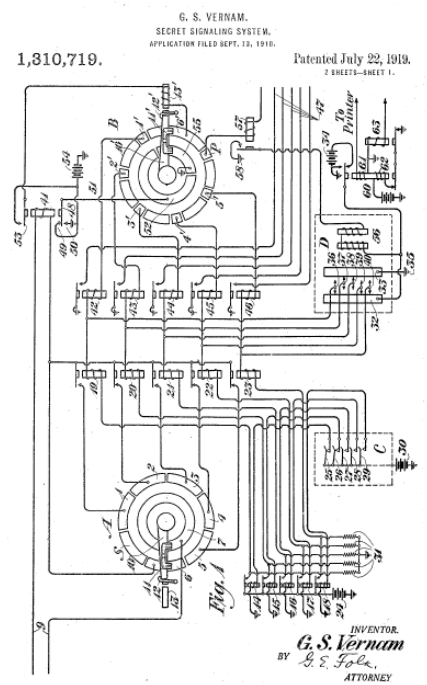


Cryptanalysis: statistical attacks

Historical ciphers

Vernam Cipher (cf. one-time pad)

- key: random sequence of *same* length as message



P	L	A	I	N	T	E	X	T
A	B	A	D	G	W	T	A	B
P	M	A	L	T	P	X	X	U

P	L	A	I	N	T	E	X	T
O	C	C	J	L	W	P	H	Q
E	N	C	R	Y	P	T	E	D

D	I	F	F	E	R	E	N	T
B	F	Y	M	U	Y	P	R	K
E	N	C	R	Y	P	T	E	D

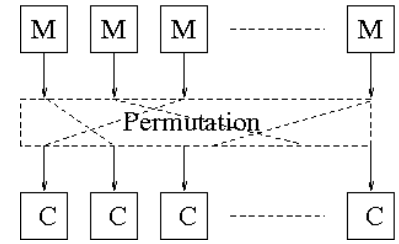
$$c_i := (m_i + k_i) \bmod 26$$

Historical ciphers

Overview:

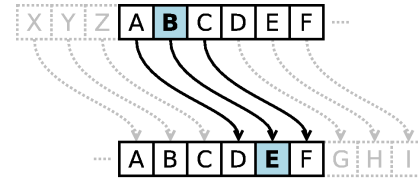
- **Permutation** (of message blocks / letters)

  E.g. scytale
  (few possible permutations)



- **Substition** (of message blocks / letters by others)
  - Monoalphabetic

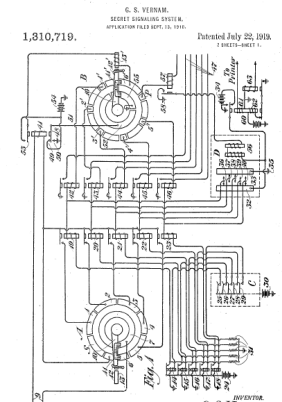
    Shift (Caesar) or random bijection



  - Polyalphabetic (substitute depends on location)

    Vigenère (key "repeats") or Vernam (long key)

M	E	S	S	A	G	E	C	L	A	I	R
C	L	E	C	L	E	C	L	E	C	L	E
O	P	W	U	L	K	G	N	P	C	T	V



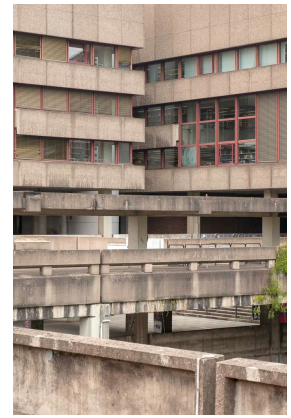
Introduction to Cryptography

(Lecture 2: Modern cryptography, the one-time pad)

Georg Fuchsbauer

TUW 2025W

Modern cryptography



- Vernam Cipher (1917)
- Information theory, perfect secrecy (Shannon 1949)
- Encryption standard: DES (Feistel 1977)
- Public-key cryptography (Diffie, Hellman 1976)
- RSA (Rivest, Shamir, Adleman 1978)
- Semantic security (Goldwasser, Micali 1984)
- Digital signatures (Goldwasser, Micali, Rivest 1986)
- Multiparty computation (Yao, Goldreich, Micali, Wigderson1986)
- Zero-knowledge proofs (Goldwasser, Micali, Rackoff 1989)
- Standards: AES (2000) SHA-2 (2001) SHA-3 (2015)
- Bitcoin (Nakamoto 2008), E-cash (Chaum, 1982)
- Lattice-based crypto (Ajtai, Dwork 1996)
- Fully homomorphic encryption (Gentry 2009)
- Obfuscation (Garg, Gentry, Halevi, Raykova, Sahai, Waters '13)
- • •

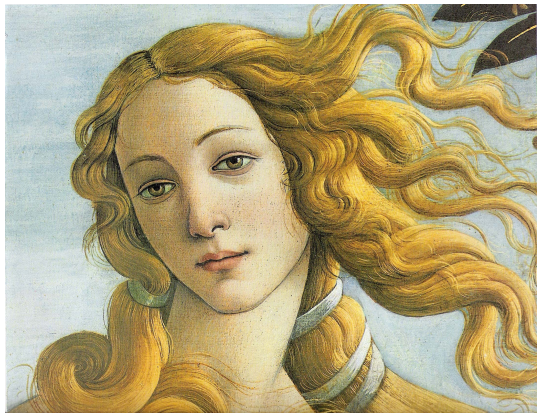
This course

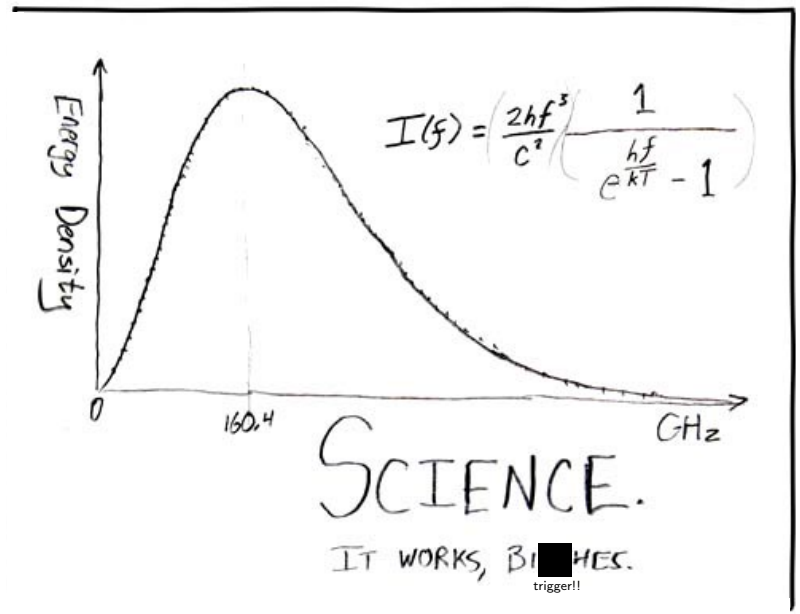
Advanced Crypto 192.115

“Modern” Cryptography

§1.4

Modern Cryptography





$\sim$ 1980: cryptography: art $\rightarrow$ science

Modern Cryptography

Provable Security (a.k.a. reductionist security):

- **Definitions:** What is *security goal*, what is *threat model* (e.g. find message, seeing several encryptions)
- **Assumptions:** Computational assumptions (e.g. factoring large integers is hard)
- **Security proof:** Mathematical proof that construction achieves *definition* under *assumptions*.

Modern Cryptography

Provable Security (a.k.a. reductionist security):

- **Definitions:** What is *security goal*, what is *threat model* (e.g. find message, seeing several encryptions)

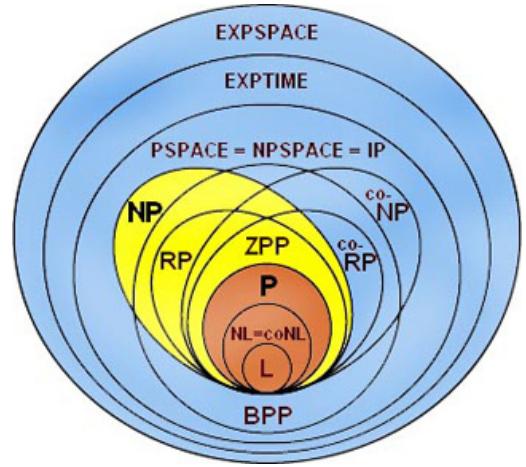
– understand what we want to achieve!

– evaluate and compare schemes

– use schemes as component (modularity!)

Modern Cryptography

Provable Security (a.k.a. reductionist security):



- **Assumptions:**
  (e.g. factoring is hard)

– on the *impossibility of solving a certain problem* <u>*efficiently*</u>

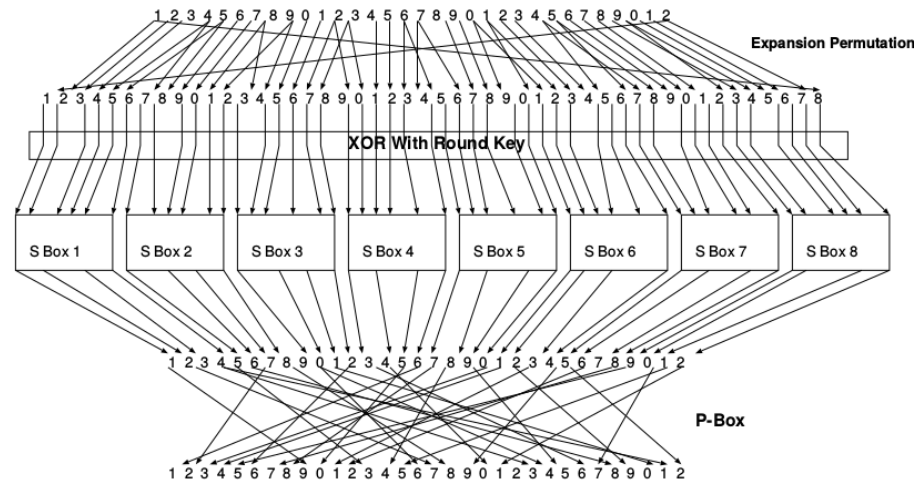
– precisely defined $\Rightarrow$ study, validation

– comparison between schemes

$P \stackrel{?}{=} NP$: are there problems where finding a solution is harder than verifying it?

Modern Cryptography

Provable Security (a.k.a. reductionist security):



- **Assumptions:**
  (e.g. factoring is hard)

  - Security of any (interesting) cryptographic scheme *implies* $P \neq NP$.
  - $\Rightarrow$ need "hardness assumptions" for proving security

- **Public-key crypto:** "mathematical" (factoring, discrete logarithm, ...)
- **Symmetric crypto:** typically *ad-hoc*

Modern Cryptography

Provable Security (a.k.a. reductionist security):

- **Security proof:** Mathematical proof that construction achieves *definition* under *assumptions*.

Provably secure scheme can still "fail" if

- assumption was wrong, or (e.g., quantum computers efficiently factor)
- definition did not reflect real requirements (e.g., scheme secure when key only used once, but saw two ciphertext)

Advantages:

- well-defined problems to analyze
- base security of *new* scheme on **well-studied** assumptions!

Security definitions

Security definitions

Kerckhoffs' Principle: The adversary knows the scheme

Auguste Kerckhoffs: *La cryptographie militaire* (1883)

- Adversary's **goals:**
  - Find the key?
  - Recover the plaintext
  - *Guess* a single letter of the plaintext
  - Obtain *any* information about the plaintext

$\mathsf{Enc}(k, m) := m$

$\mathsf{Enc}(k, m_1 \| m_2) := \mathsf{AES}(k, m_1) \| m_2$

- Adversary's **power:**
  - Sees ciphertexts (one/many)
  - Has seen plaintext/ciphertext pairs
  - Has chosen the plaintexts

    ...and can ask for *decryption*

Perfectly secret encryption

§2

One-time pad

- Red phone between Kremlin and White House
- Perfectly secure (if used correctly)
- Not practical

P	L	A	I	N	T	E	X	T
A	B	A	D	G	W	T	A	B
P	M	A	L	T	P	X	X	U

0	0	0	1	1	1	1	0	1
1	0	0	1	0	1	1	0	0
1	0	0	0	1	0	0	0	1

One-time pad

- Red phone between Kremlin and White House
- Perfectly secure (if used correctly)
- Not practical

Uses binary arithmetic ("XOR")

$\oplus$	0	1
0	0	1
1	1	0

$(x + y) \bmod 2$

- Key: $k \leftarrow \{0,1\}^\ell$ (bit strings of length ℓ)
- Message: $m \in \{0,1\}^\ell =: \mathcal{M}$
- Encryption: $c := k \oplus m = k_1 \oplus m_1 \| \ldots \| k_\ell \oplus m_\ell$
- Decryption: $m := k \oplus c$

One-time pad

Message space: $\mathcal{M} := \{0,1\}^\ell$

$\mathsf{Gen}(\ell)$:

- $k \leftarrow \{0,1\}^\ell$ // choose uniformly random ℓ-bit string
- return k

$\mathsf{Enc}(k, m)$: // $k, m \in \{0,1\}^\ell$

- for $i = 1 \ldots \ell$ do
  $c_i := k_i \oplus m_i$
- return $(c_1, \ldots, c_\ell)$

$\mathsf{Dec}(k, c)$: // $k, c \in \{0,1\}^\ell$

- for $i = 1 \ldots \ell$ do
  $m_i := k_i \oplus c_i$
- return $(m_1, \ldots, m_\ell)$

$$\begin{aligned}\mathsf{Dec}(k, \mathsf{Enc}(k, m)) &= \\ &= k \oplus (k \oplus m) \\ &= (\underbrace{k \oplus k}_{=0,\ldots,0}) \oplus m \\ &= m\end{aligned}$$

Security? Definition!

Probability theory

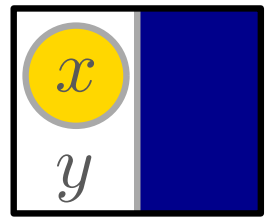
§A.3

Random variable X

$\Pr[X=x]$ . . . probability that X takes value x

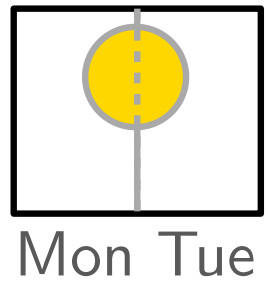
Conditional probability:

$$\Pr[X=x \mid Y=y] := \frac{\Pr[X=x \wedge Y=y]}{\Pr[Y=y]}$$



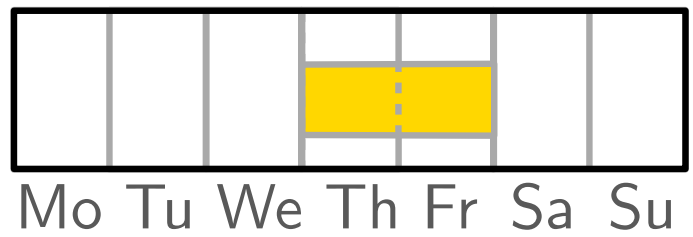
Bayes' Theorem:

$$\Pr[X=x \mid Y=y] = \Pr[Y=y \mid X=x] \cdot \frac{\Pr[X=x]}{\Pr[Y=y]}$$



X and Y are **independent** if

$$\forall\, x, y : \Pr[X=x \mid Y=y] = \Pr[X=x]$$



Total probability: $\sum_{y_i} \Pr[Y=y_i] = 1$

$$\Rightarrow\ \Pr[X=x] = \sum_{y_i} \Pr[X=x \mid Y=y_i] \cdot \Pr[Y=y_i]$$

Perfect secrecy

Regardless of any information an attacker already has, a ciphertext should leak no additional information about the underlying plaintext

- Let M be a random variable representing **known information** about message

Example. $\Pr[M = \text{“}\texttt{buy}\text{”}] = 0.7$

$\Pr[M = \text{“}\texttt{sell}\text{”}] = 0.3$

- Let K be random variable denoting the key value, i.e. $\Pr[K = k] = \Pr[k \leftarrow \text{Gen}]$

  We assume that K and M are independent

Perfect secrecy

Example 2.1 Shift cipher ($k \in \{0, \ldots, 25\}, \Pr[K = k] = 1/26$)

Assume $\Pr[M = \mathtt{B}] = 0.7$ and $\Pr[M = \mathtt{D}] = 0.3$

$\Pr[C = \mathtt{F}] = ?$

$$= \Pr[M = \mathtt{B} \wedge K = 4] + \Pr[M = \mathtt{D} \wedge K = 2]$$

$$\overset{M, K \text{ indep.}}{=} 0.7 \cdot 1/26 + 0.3 \cdot 1/26$$

$$= (0.7 + 0.3) \cdot 1/26 = 1/26$$

- Let C be random variable denoting the ciphertext resulting:
  - choose m from distribution M
  - $k \leftarrow \mathsf{Gen}$
  - $c \leftarrow \mathsf{Enc}_k(m)$

Perfect secrecy

Example 2.1. Shift cipher ($k \in \{0, \ldots, 25\}, \Pr[K = k] = 1/26$)

Assume $\Pr[M = \mathtt{B}] = 0.1$ and $\Pr[M = \mathtt{D}] = 0.9$

$\Pr[C = \mathtt{G}] = ?$

$$= \Pr[M = \mathtt{B} \wedge K = 5] + \Pr[M = \mathtt{D} \wedge K = 3]$$

$$\overset{M, K \text{ indep.}}{=} 0.1 \cdot 1/26 + 0.9 \cdot 1/26$$

$$= (0.1 + 0.9) \cdot 1/26 = 1/26$$

- Let C be random variable denoting the ciphertext resulting:
  - choose m from distribution M
  - $k \leftarrow \mathsf{Gen}$
  - $c \leftarrow \mathsf{Enc}_k(m)$

Perfect secrecy

Regardless of any information an attacker already has, a ciphertext should leak no additional information about the underlying plaintext

- Let M be a random variable representing **known information** about message
- Let C be random variable denoting the resulting ciphertext

Def. 2.3. An encryption scheme is **perfectly secret** if for every probability distribution over the message space $\mathcal{M}$, every $m \in \mathcal{M}$ and every ciphertext c with $\Pr[C = c] > 0$:

$$\Pr\left[M = m \,\middle|\, C = c\right] = \Pr\left[M = m\right]$$

(Adversary's power: only sees one ciphertext)

Security of the one-time pad

Theorem 2.10. The one-time pad is perfectly secret.

$\mathsf{Gen}(\ell)$: return $k \leftarrow \{0,1\}^{\ell}$

$\mathsf{Enc}_k(m) := k \oplus m$

$\mathsf{Dec}_k(c) := k \oplus c$

Security of the one-time pad

Theorem 2.10. The one-time pad is perfectly secret.

$$(\forall\, M, m, c : \Pr\left[M = m \,\middle|\, C = c\right] = \Pr\left[M = m\right])$$

Proof:

Keys random $\Rightarrow \Pr[K = k] = 1/2^\ell$ for all k $\qquad (*)$

Let M, m and c be arbitrarily fixed. We have

$$\Pr[C = c \mid M = m] = \Pr[K \oplus M = c \mid M = m] = \Pr[K \oplus m = c \mid M = m]$$

$$= \Pr[K = m \oplus c \mid M = m] \overset{K, M \text{ indep.}}{=} \Pr[K = m \oplus c] \overset{(*)}{=} 1/2^\ell \quad (**)$$

$$\Pr[C = c] \overset{\text{tot. prob.}}{=} \sum_{m'} \Pr[C = c \mid M = m'] \cdot \Pr[M = m']$$

$$\overset{(**)}{=} 1/2^\ell \cdot \sum_{m'} \Pr[M = m'] = 1/2^\ell \quad (***)$$

$$\Pr[M = m \mid C = c] \overset{\text{Bayes}}{=} \frac{\Pr[C = c \mid M = m] \cdot \Pr[M = m]}{\Pr[C = c]} \overset{(**) \text{ and } (***)}{=} \Pr[M = m]$$

□

Security of the one-time pad

- What if key is used twice?

$$c = k \oplus m$$
$$c' = k \oplus m' \quad \Rightarrow \quad c \oplus c' = \underbrace{k \oplus k}_{=0,\ldots,0} \oplus m \oplus m' = m \oplus m'$$

Bad?

m : "B" = 100 0010 or "S" = 101 0011 ("buy" or "sell")
m': "Y" = 101 0110 or "N" = 100 1110 ("yes" or "no")

$c \oplus c' = 001\,0100 \Rightarrow \quad m = \mathsf{B} \quad m' = \mathsf{Y}$

$c \oplus c' = 000\,0101 \Rightarrow \quad m = ? \quad m' = ?$

Security of the one-time pad

- What if key is used twice?

$$c = k \oplus m$$
$$c' = k \oplus m' \quad \Rightarrow \quad c \oplus c' = \underbrace{k \oplus k}_{=0,\ldots,0} \oplus m \oplus m' = m \oplus m'$$

Drawbacks of the one-time pad:

- key as **long** as message
- key can only be used **once**

$\Rightarrow$ requires long true random values

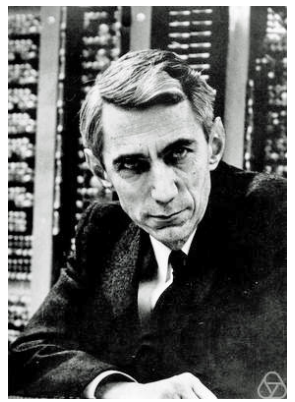
$\Rightarrow$ key distribution

$\Rightarrow$ key destruction

Limitations of perfect secrecy

Theorem 2.11. For any perfectly secret encryption: $|\mathcal{K}| \geq |\mathcal{M}|$

Theorem 2.12 (Shannon). An encryption scheme with $|\mathcal{M}| = |\mathcal{K}| = |\mathcal{C}|$ is perfectly secret *if and only if*



– every key is equiprobable and
– for all m and c there is a unique k
such that $\text{Enc}_k(m) = c$

- key as **long** as message
- key can only be used **once**

for any perfectly secret encryption scheme

secure against adversaries with **infinite power!**

. . . and if we want **practical** schemes
secure against **bounded** adversaries?

Introduction to Cryptography

(Lecture 3: Block ciphers)

Elena Andreeva

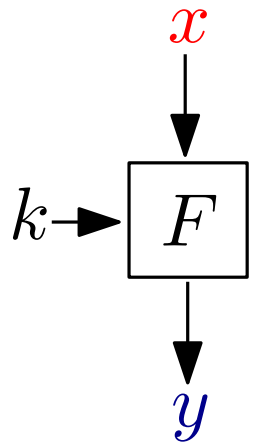
TUW 2025W

Block ciphers

§7.2

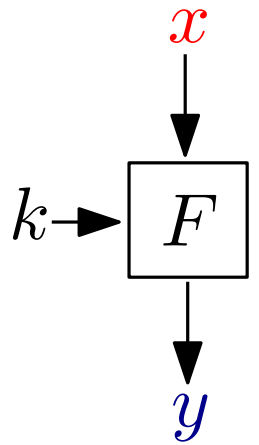
Block ciphers

- Main **building block** of private-key encryption schemes, MACs, ...

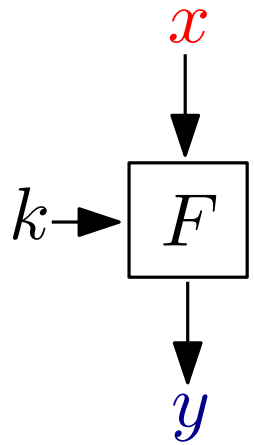


Block ciphers

- Main **building block** of private-key encryption schemes, MACs, ...
- Encrypts blocks of fixed size



Block ciphers



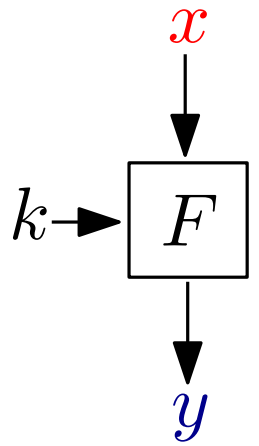
- Main **building block** of private-key encryption schemes, MACs, ...
- Encrypts blocks of fixed size

  (substitution cipher: block = 1 letter)

π:

A	$\rightarrow$	K
B	$\rightarrow$	X
C	$\rightarrow$	M
D	$\rightarrow$	Q
	$\vdots$	

Block ciphers

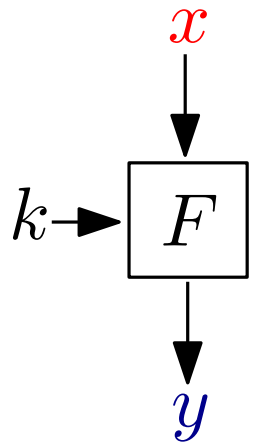


- Main **building block** of private-key encryption schemes, MACs, ...
- Encrypts blocks of fixed size

Setting. Adversary knows algorithm (Kerckhoffs)

– has **seen** (chosen) plaintext/ciphertext pairs

Block ciphers



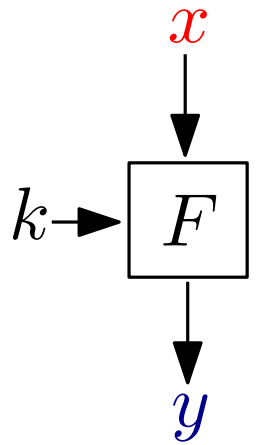
- Main **building block** of private-key encryption schemes, MACs, ...
- Encrypts blocks of fixed size

Setting. Adversary knows algorithm (Kerckhoffs)

– has **seen** (chosen) plaintext/ciphertext pairs

Generic attacks:

Block ciphers

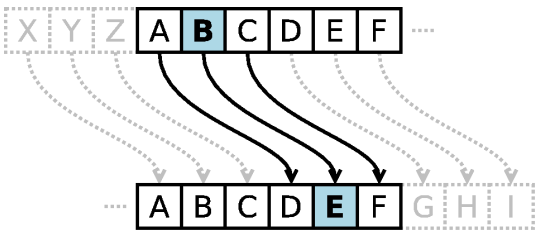


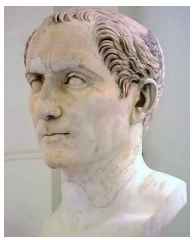
- Main **building block** of private-key encryption schemes, MACs, ...
- Encrypts blocks of fixed size

Setting. Adversary knows algorithm (Kerckhoffs)

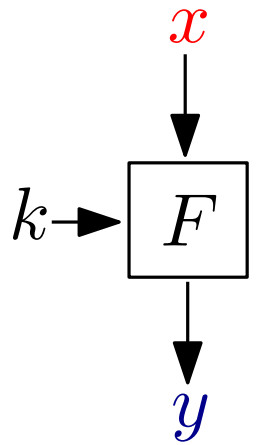
– has **seen** (chosen) plaintext/ciphertext pairs

Generic attacks:





Block ciphers



- Main **building block** of private-key encryption schemes, MACs, ...
- Encrypts blocks of fixed size

Setting. Adversary knows algorithm (Kerckhoffs)

– has **seen** (chosen) plaintext/ciphertext pairs

Generic attacks:

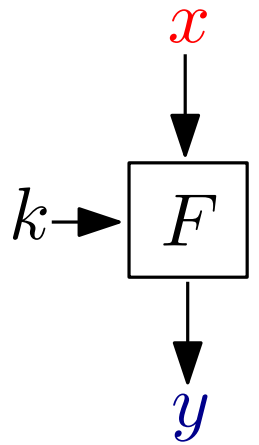
- **Exhaustive search** on the keys

  $\Rightarrow$ Number of possible keys must be large ($\geq 2^{128}$)

  (1 quintillion (10^{18}) keys /s $\Rightarrow$ takes age of universe)

Block ciphers

- Main **building block** of private-key encryption schemes, MACs, ...
- Encrypts blocks of fixed size



Setting. Adversary knows algorithm (Kerckhoffs)

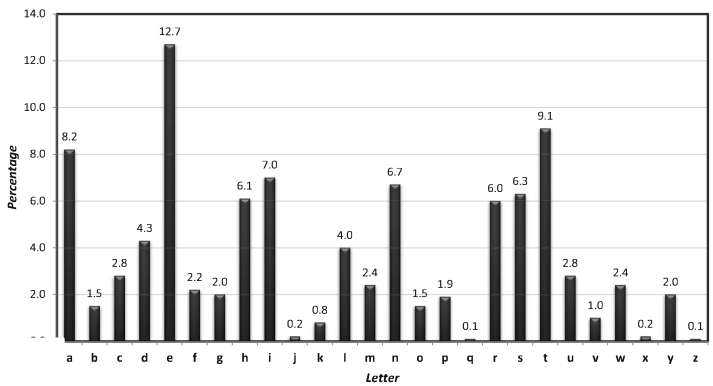
– has **seen** (chosen) plaintext/ciphertext pairs

Generic attacks:

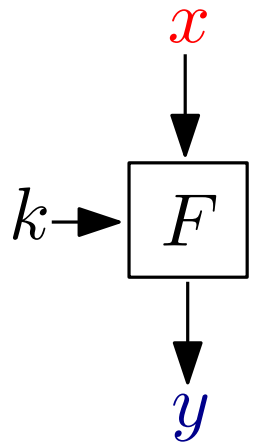
- **Exhaustive search** on the keys

  $\Rightarrow$ Number of possible keys must be large ($\geq 2^{128}$)

  (1 quintillion (10^{18}) keys /s



Block ciphers



- Main **building block** of private-key encryption schemes, MACs, ...
- Encrypts blocks of fixed size

Setting. Adversary knows algorithm (Kerckhoffs)

– has **seen** (chosen) plaintext/ciphertext pairs

Generic attacks:

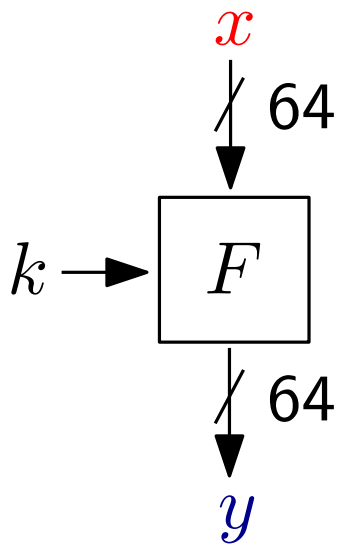
- **Exhaustive search** on the keys

  $\Rightarrow$ Number of possible keys must be large ($\geq 2^{128}$)

  (1 quintillion (10^{18}) keys /s $\Rightarrow$ takes age of universe)

- **Dictionary attacks** $\Rightarrow$ Block size ℓ must be large (≥ 64bits)

Security goal

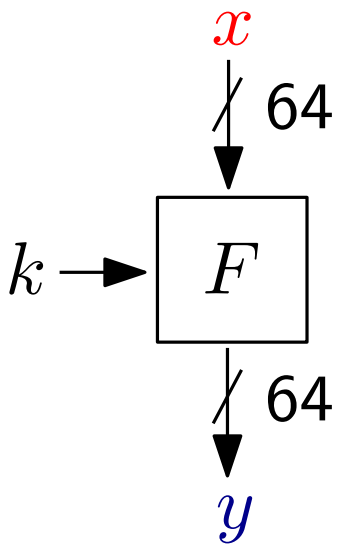


Ideal cipher

- key defines **random permutation**

$$F_k : \{0,1\}^\ell \to \{0,1\}^\ell$$

Security goal



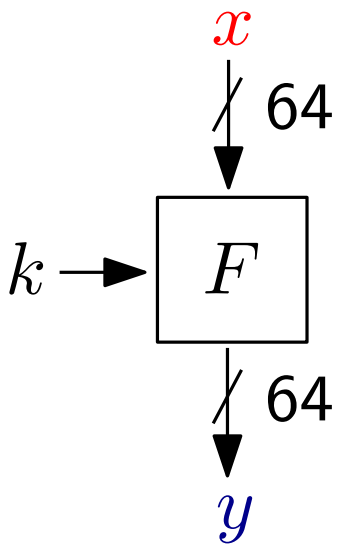
Ideal cipher

- key defines **random permutation**

$$F_k : \{0,1\}^\ell \to \{0,1\}^\ell$$

inputs	outputs
0...000	1...010
0...001	0...011
0...010	0...110
⋮	⋮
1...110	1...100
1...111	0...001

Security goal



Ideal cipher

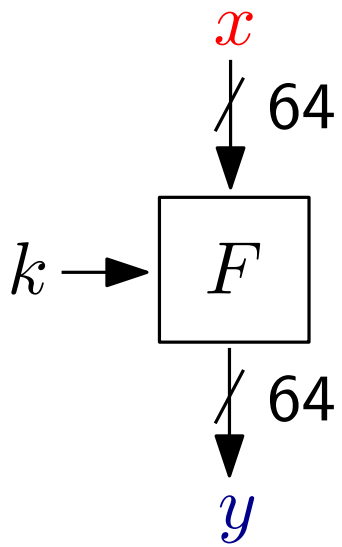
- key defines **random permutation**

$$F_k : \{0,1\}^\ell \to \{0,1\}^\ell$$

Compare to **one-time pad**:

inputs		outputs
0...000		1...010
0...001		0...011
0...010	→	0...110
⋮		⋮
1...110		1...100
1...111		0...001

Security goal

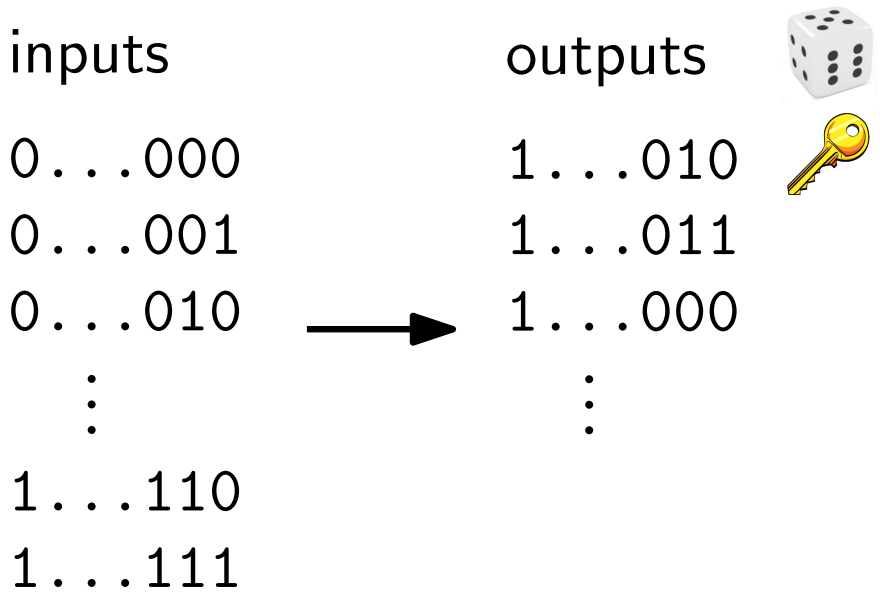


Ideal cipher

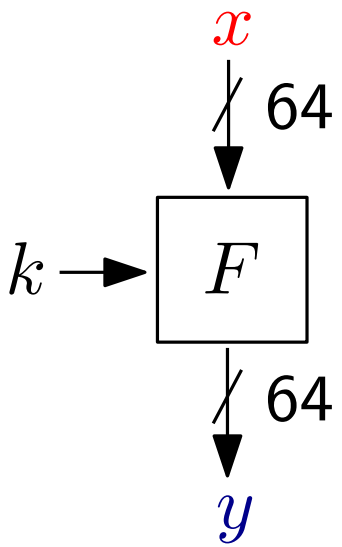
- key defines **random permutation**

$$F_k : \{0,1\}^\ell \to \{0,1\}^\ell$$

Compare to **one-time pad**:



Security goal



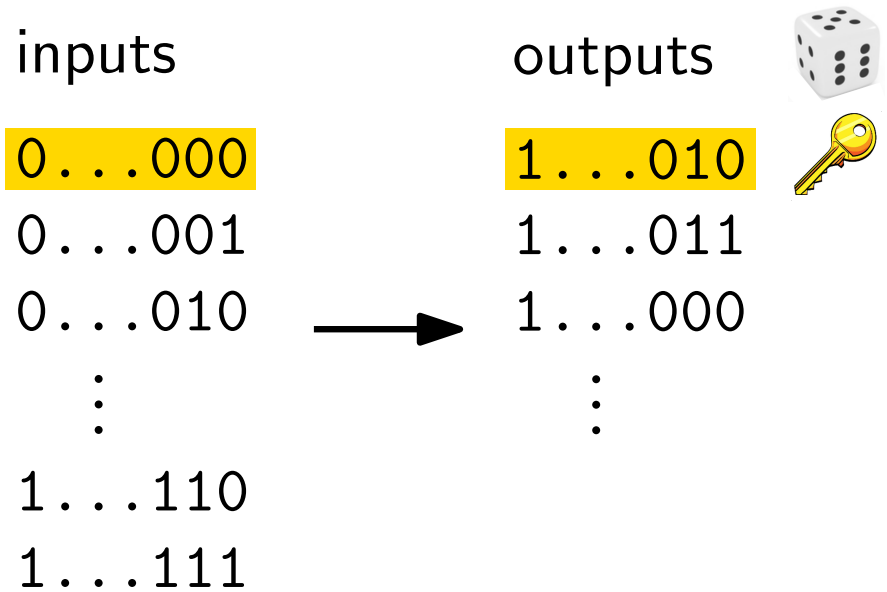
Ideal cipher

- key defines **random permutation**

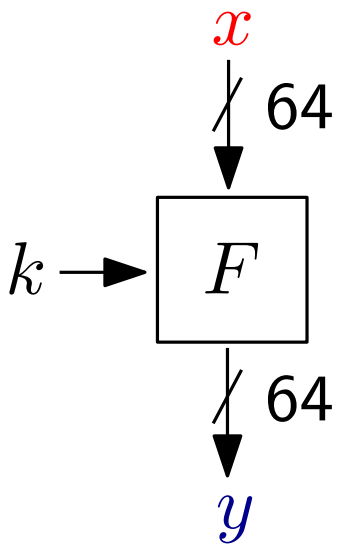
$$F_k : \{0,1\}^\ell \to \{0,1\}^\ell$$

Compare to **one-time pad**:

- one x/y pair determines **all** pairs!



Security goal



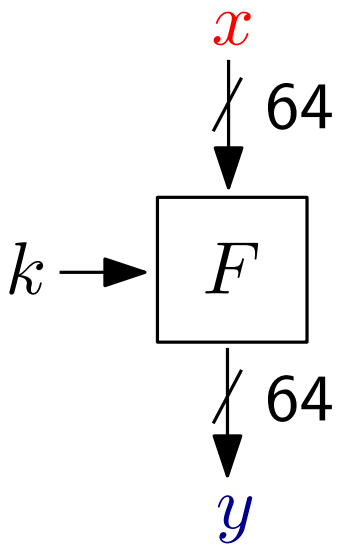
Ideal cipher

- key defines **random permutation**

$$F_k : \{0,1\}^\ell \to \{0,1\}^\ell$$

inputs	outputs
0...000	1...010
0...001	0...011
0...010	0...110
⋮	⋮
1...110	1...100
1...111	0...001

Security goal

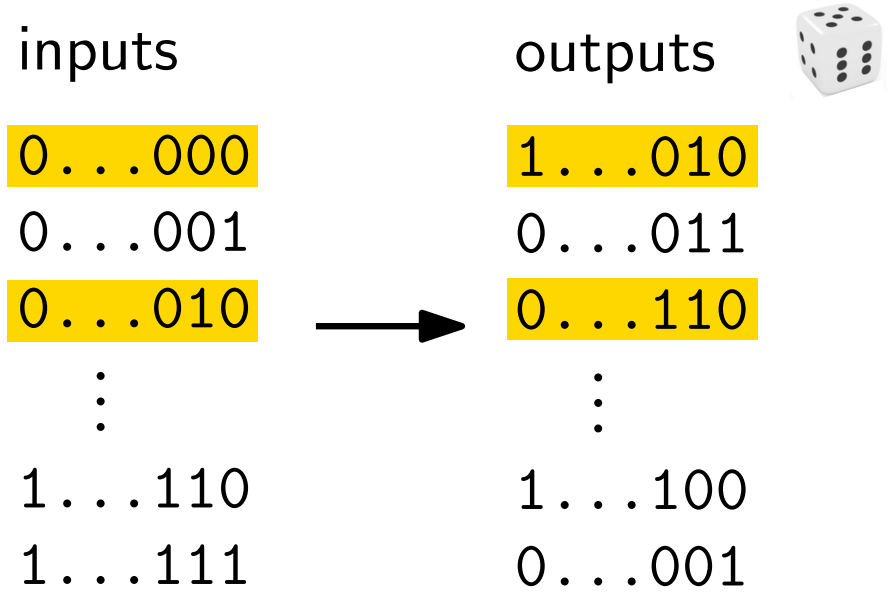


Ideal cipher

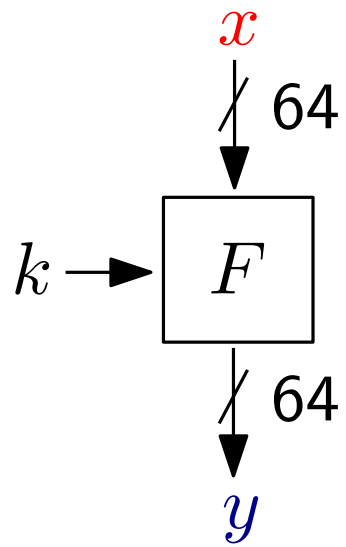
- key defines **random permutation**

$$F_k : \{0,1\}^\ell \to \{0,1\}^\ell$$

- after seeing x/y pairs, what do we learn?



Security goal

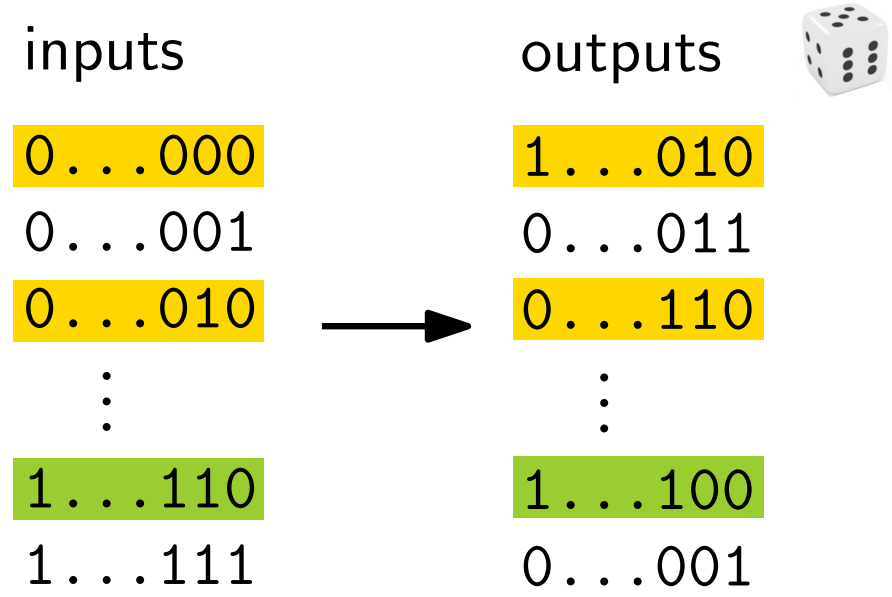


Ideal cipher

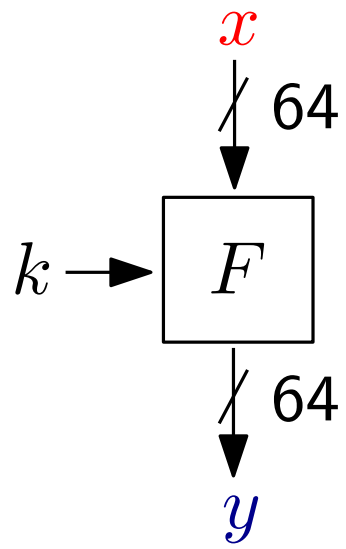
- key defines **random permutation**

$$F_k : \{0,1\}^\ell \to \{0,1\}^\ell$$

- after seeing x/y pairs, what do we learn?



Security goal

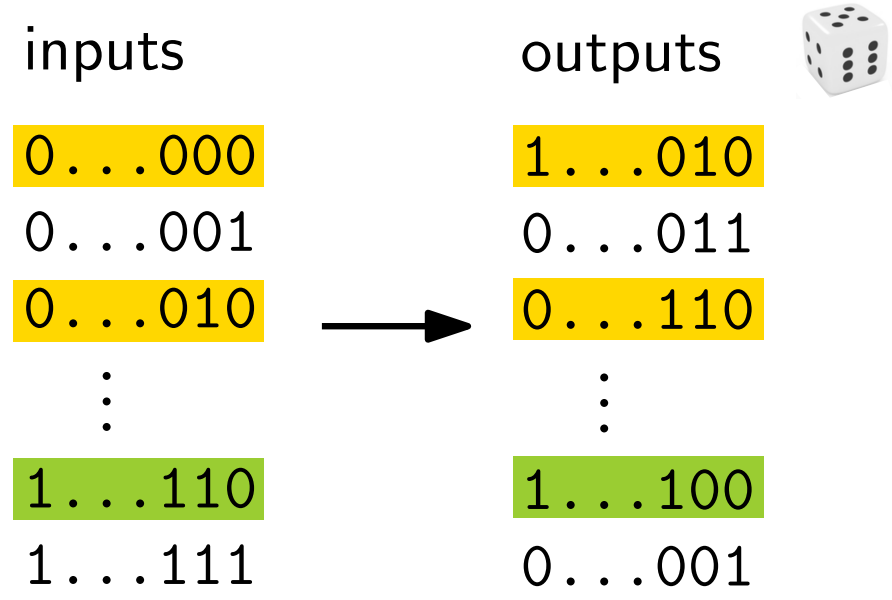


Ideal cipher

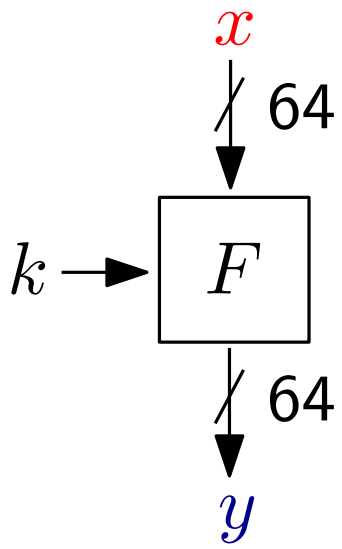
- key defines **random permutation**

$$F_k : \{0,1\}^\ell \to \{0,1\}^\ell$$

- after seeing x/y pairs, what do we learn?
- security is "ideal", but...



Security goal

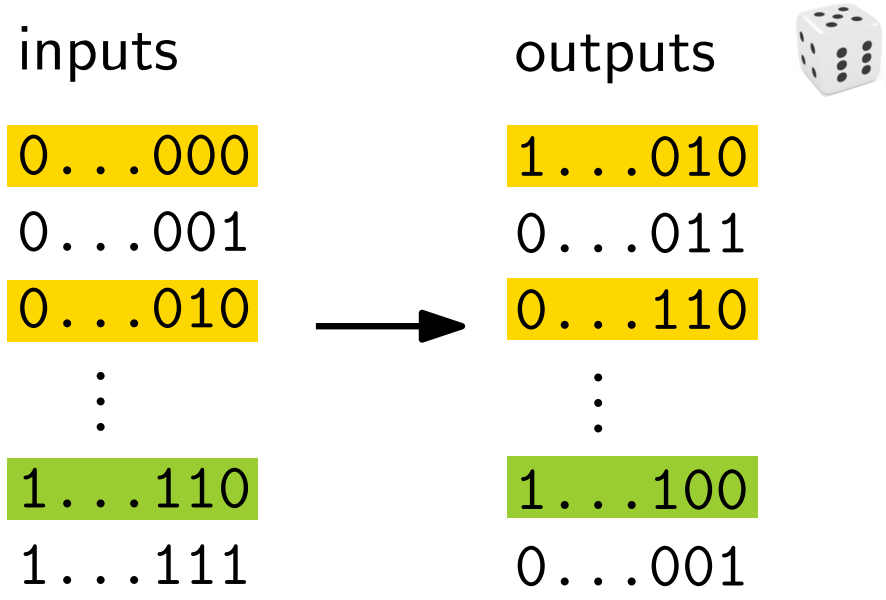


Ideal cipher

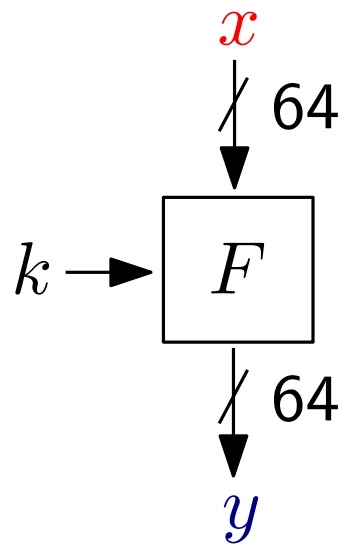
- key defines **random permutation**

$$F_k : \{0,1\}^\ell \to \{0,1\}^\ell$$

- after seeing x/y pairs, what do we learn?
- security is "ideal", but...
- Number of permutations:



Security goal

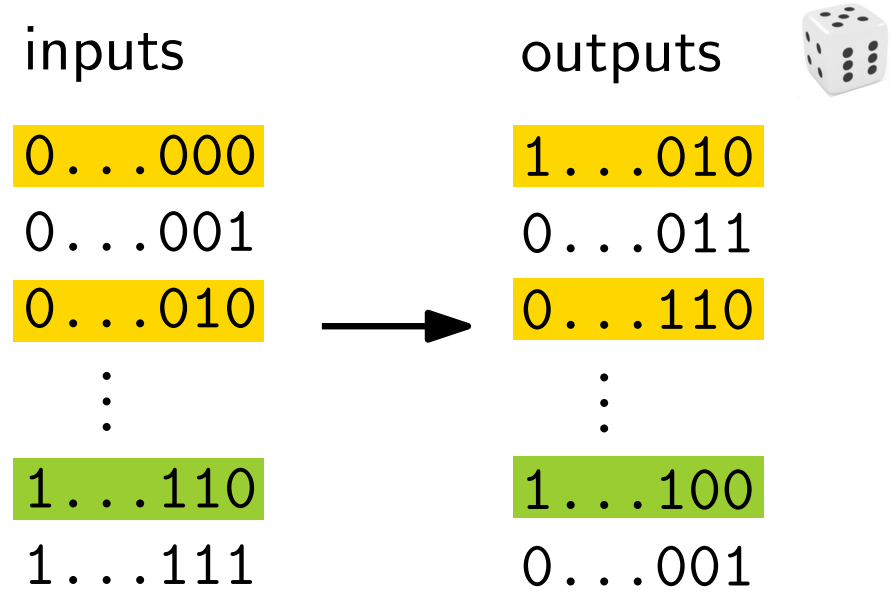


Ideal cipher

- key defines **random permutation**

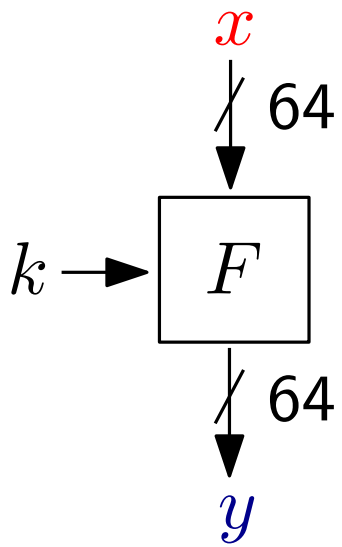
$$F_k : \{0,1\}^\ell \to \{0,1\}^\ell$$

- after seeing x/y pairs, what do we learn?
- security is "ideal", but...
- Number of permutations: $2^\ell!$
  $\ell = 64$: $|k| \approx 2^{70}$ $(^*)$



$(^*) \approx$ 150 million TB

Security goal



Ideal cipher

- key defines **random permutation**

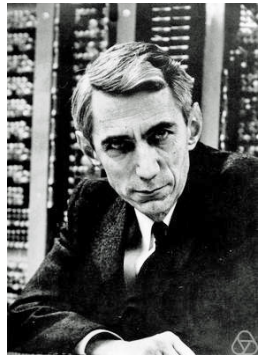
$$F_k : \{0,1\}^\ell \to \{0,1\}^\ell$$

Goal:

Construct an **efficiently computable** (keyed) permutation F with **short** keys that **"behaves"** like the ideal cipher

(*) $\approx$ 150 million TB

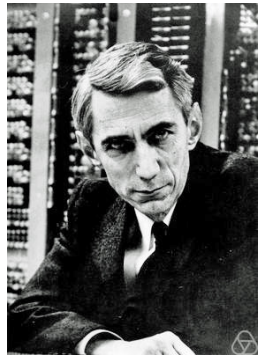
Methods



Goals (Shannon):

- Confusion: relation between x and y "obscured"
- Diffusion: every bit of y depends on many bits of x.

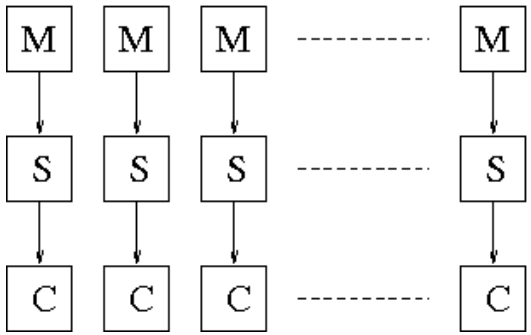
Methods

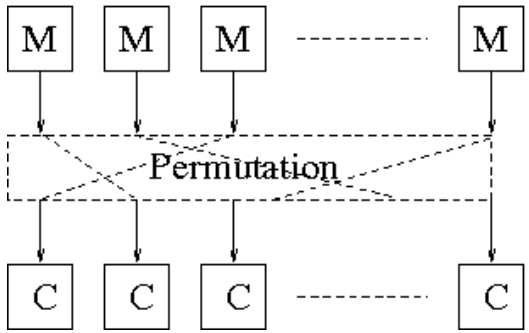


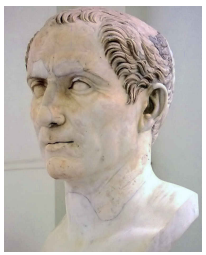
Goals (Shannon):

- Confusion: relation between x and y "obscured"
- Diffusion: every bit of y depends on many bits of x.

Principles:



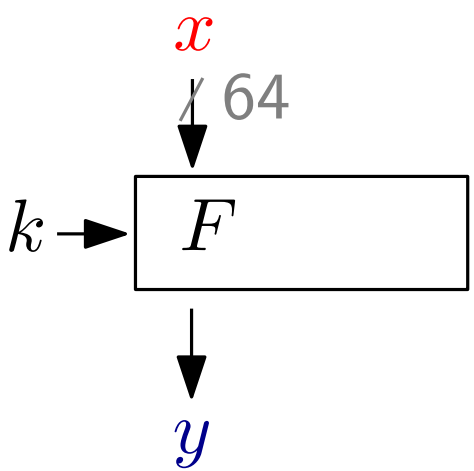




Substitution
$\Rightarrow$ confusion

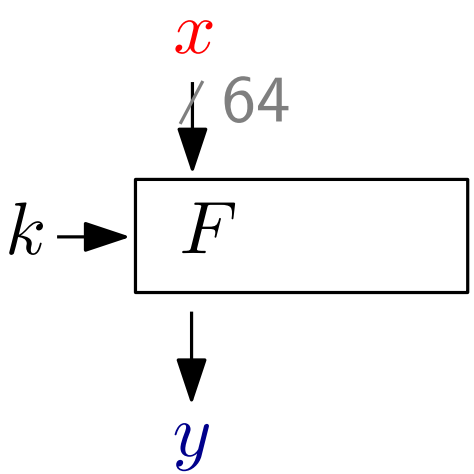
Permutation
$\Rightarrow$ diffusion

Substitution-permutation networks



- Split the input into (8-bit) blocks

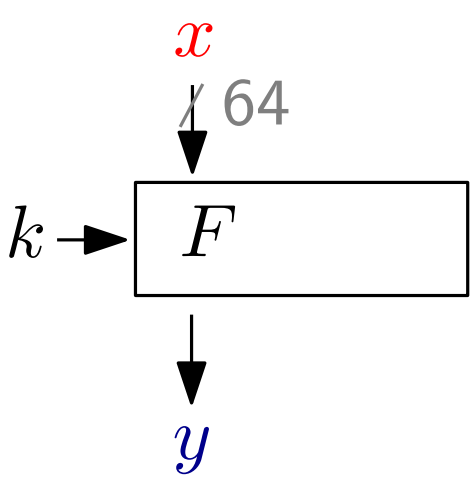
Substitution-permutation networks



x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8

- Split the input into (8-bit) blocks

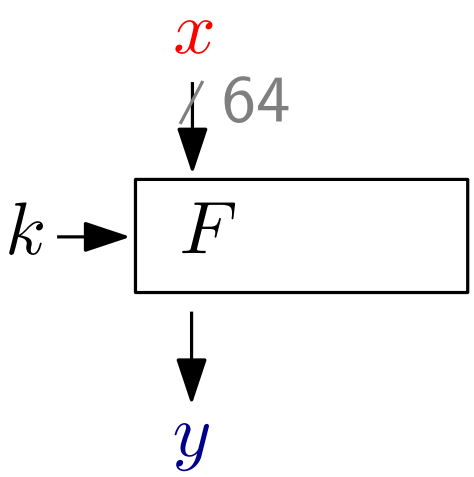
Substitution-permutation networks

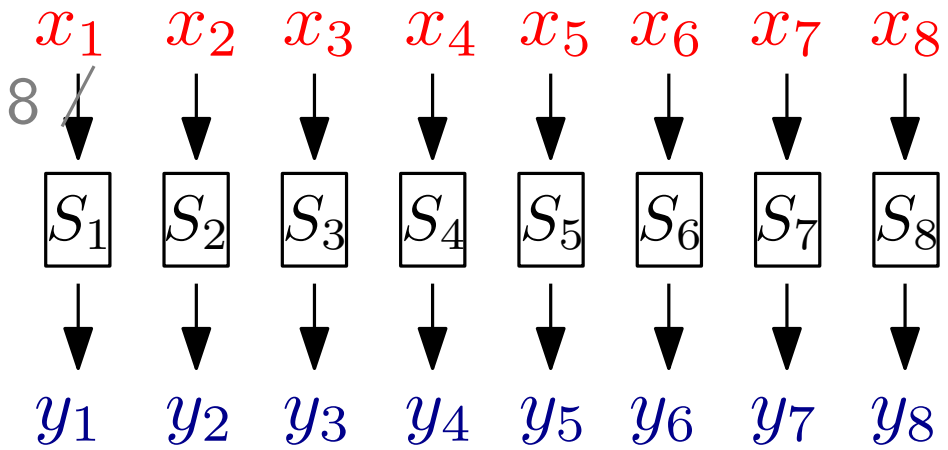


x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8

- Split the input into (8-bit) blocks
- Apply (random) substitution to small blocks

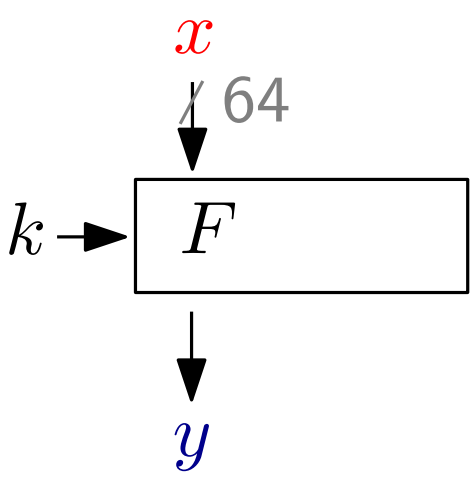
Substitution-permutation networks

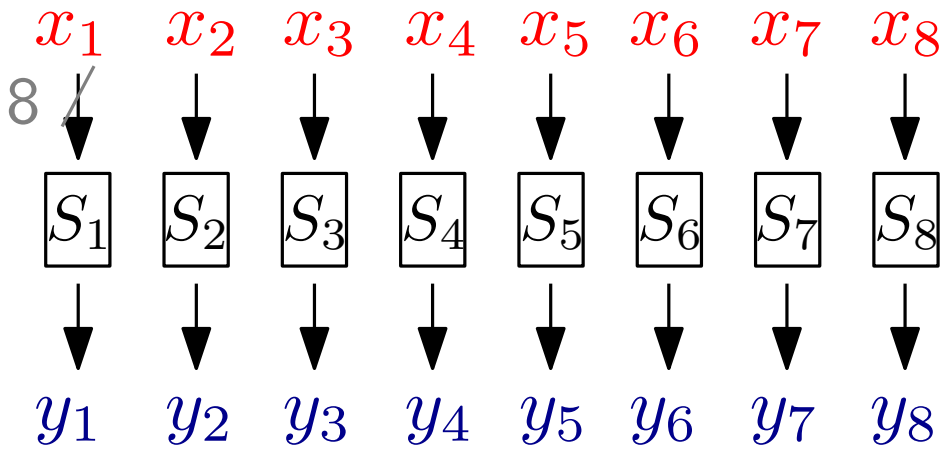




- Split the input into (8-bit) blocks
- Apply (random) substitution to small blocks

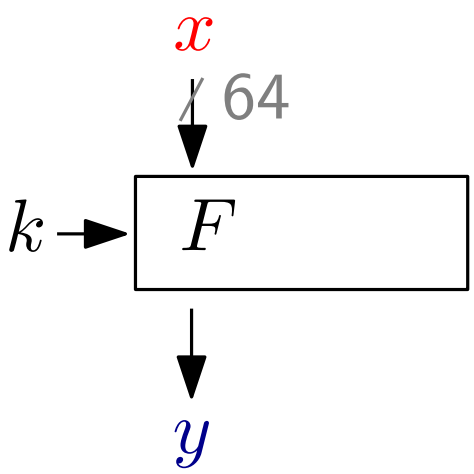
Substitution-permutation networks

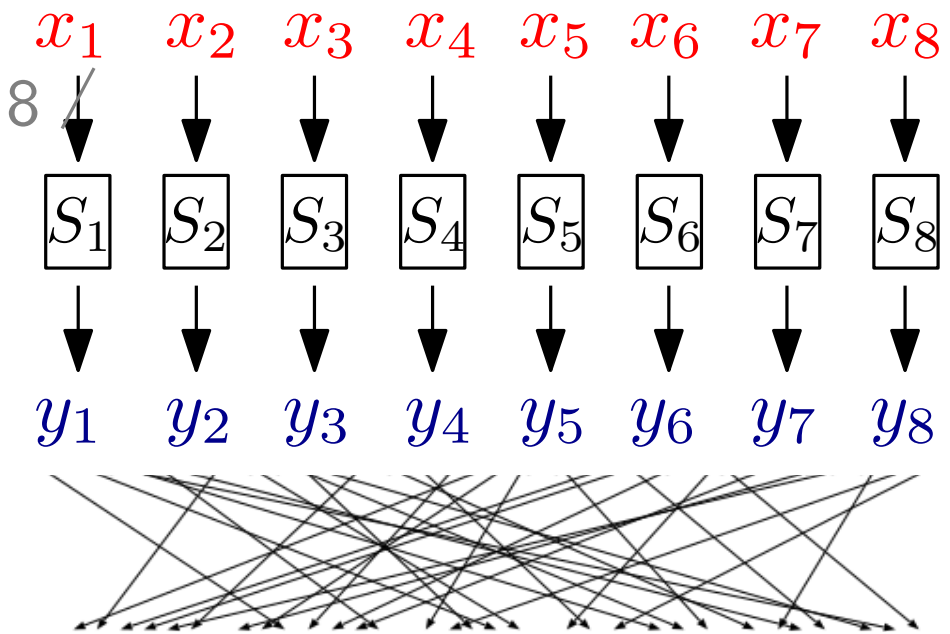




- Split the input into (8-bit) blocks
- Apply (random) substitution to small blocks
- Mix the ouptut bits

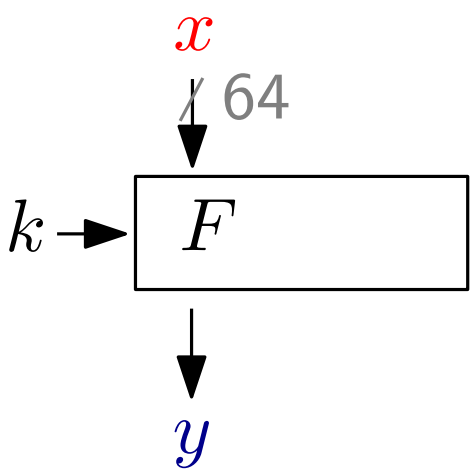
Substitution-permutation networks

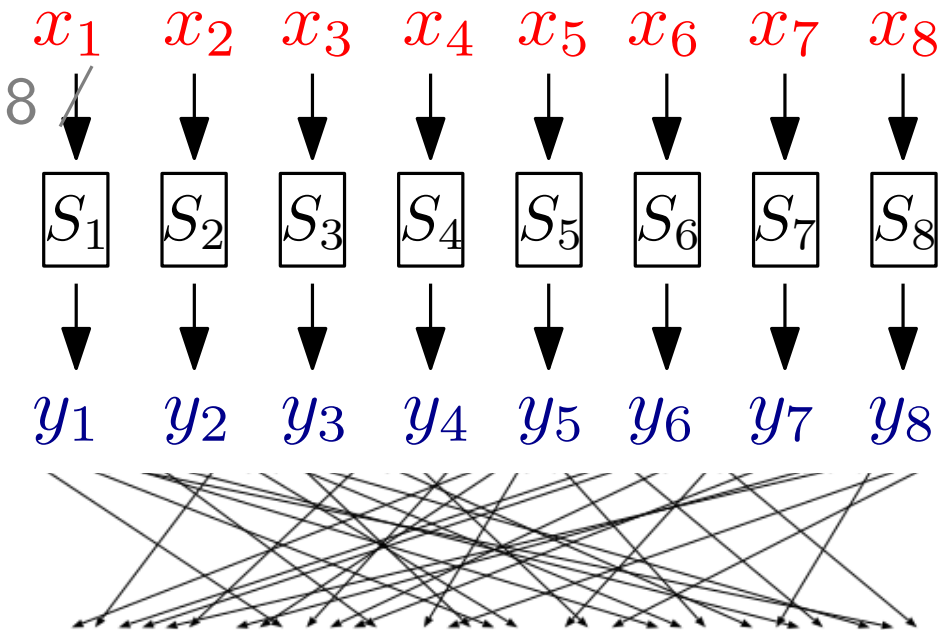




- Split the input into (8-bit) blocks
- Apply (random) substitution to small blocks
- Mix the ouptut bits

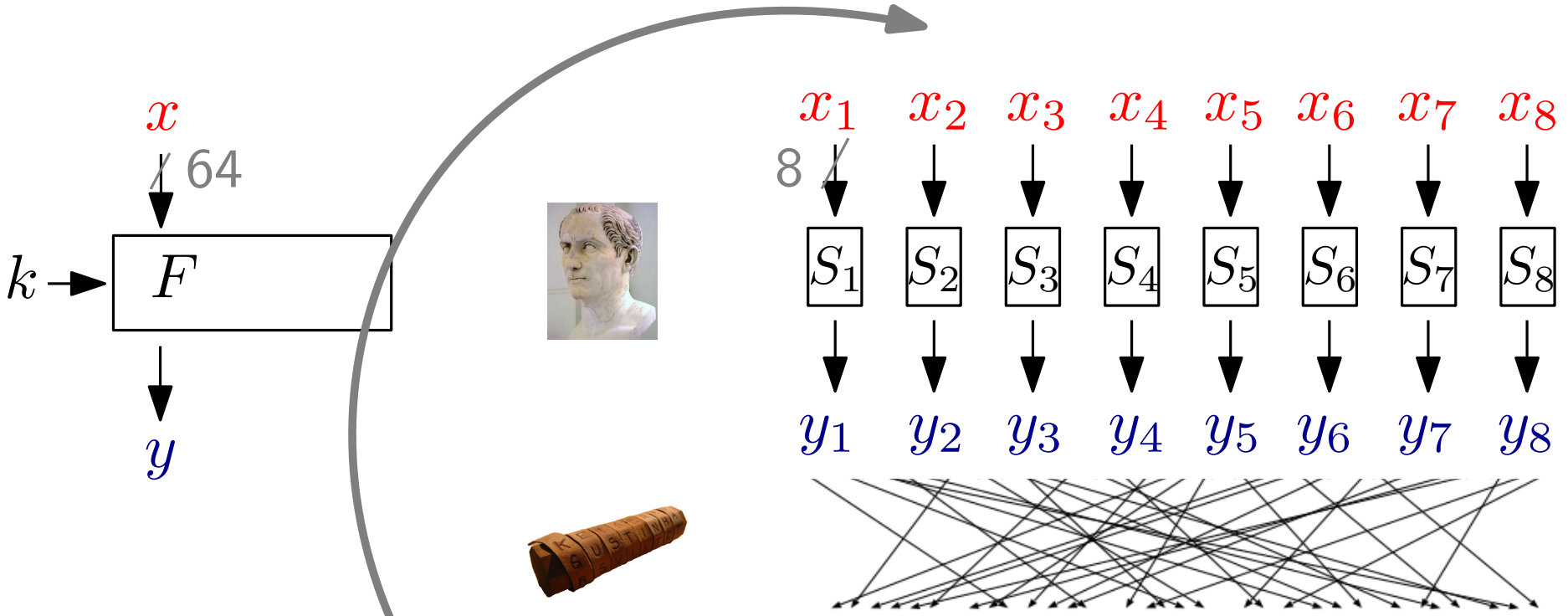
Substitution-permutation networks





- Split the input into (8-bit) blocks
- Apply (random) substitution to small blocks
- Mix the ouptut bits

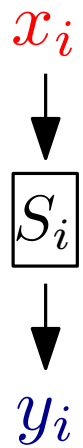
Substitution-permutation networks



- Split the input into (8-bit) blocks
- Apply (random) substitution to small blocks
- Mix the ouptut bits
- Start again . . . (using different S_i's)

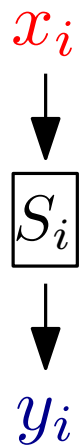
Characteristics

- Instead of $y_i := S_i(x_i)$, use **fixed** function S
- E.g., in AES: $S : \{0,1\}^8 \to \{0,1\}^8$



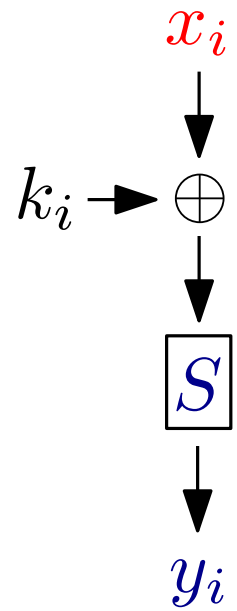
Characteristics

- Instead of $y_i := S_i(x_i)$, use **fixed** function S
- E.g., in AES: $S : \{0,1\}^8 \to \{0,1\}^8$
- Set $y_i := S(x_i \oplus k_i)$



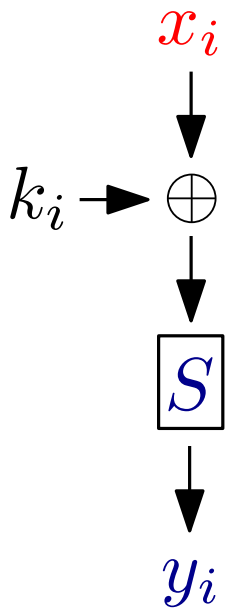
Characteristics

- Instead of $y_i := S_i(x_i)$, use **fixed** function S
- E.g., in AES: $S : \{0,1\}^8 \to \{0,1\}^8$
- Set $y_i := S(x_i \oplus k_i)$



Characteristics

- Instead of $y_i := S_i(x_i)$, use **fixed** function S
- E.g., in AES: $S : \{0,1\}^8 \to \{0,1\}^8$
- Set $y_i := S(x_i \oplus k_i)$

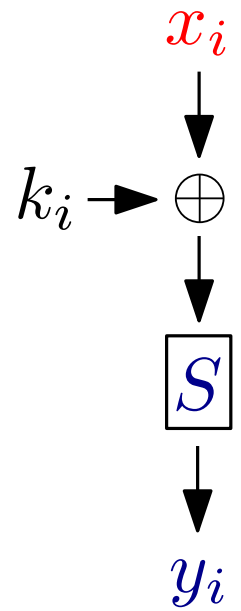


S-Box *(Substitution-box)*

- central component of block ciphers
- adds confusion (non-linearity)

Characteristics

- Instead of $y_i := S_i(x_i)$, use **fixed** function S
- E.g., in AES: $S : \{0,1\}^8 \to \{0,1\}^8$
- Set $y_i := S(x_i \oplus k_i)$



S-Box *(Substitution-box)*

- central component of block ciphers
- adds confusion (non-linearity)

Iterative encryption:

- proceed in rounds
- round i: apply **invertible** function $f_{k_i} : \{0,1\}^\ell \to \{0,1\}^\ell$ where k_i is **subkey** derived from k

DES

DES

(Data Encryption Standard), 1977

- Motivated by commercial applications
- 1972: call by NBS ($\rightarrow$ NIST)
- 1974: candiates, among them:

  Lucifer by IBM (Feistel)

DES

(Data Encryption Standard), 1977

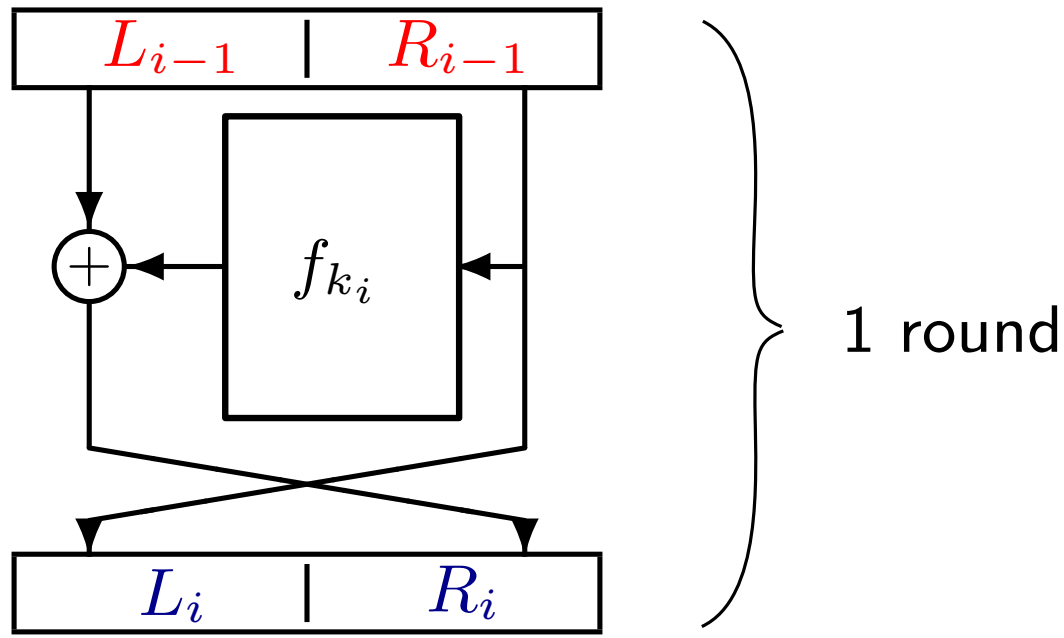
- Motivated by commercial applications
- 1972: call by NBS ($\rightarrow$ NIST)
- 1974: candiates, among them:

  Lucifer by IBM (Feistel)

- NSA "helped" (key length 128 $\rightarrow$ 56)

  $\rightarrow$ DES
- 1977: standard published (but criteria unknown)
- Standard prolonged until 1999

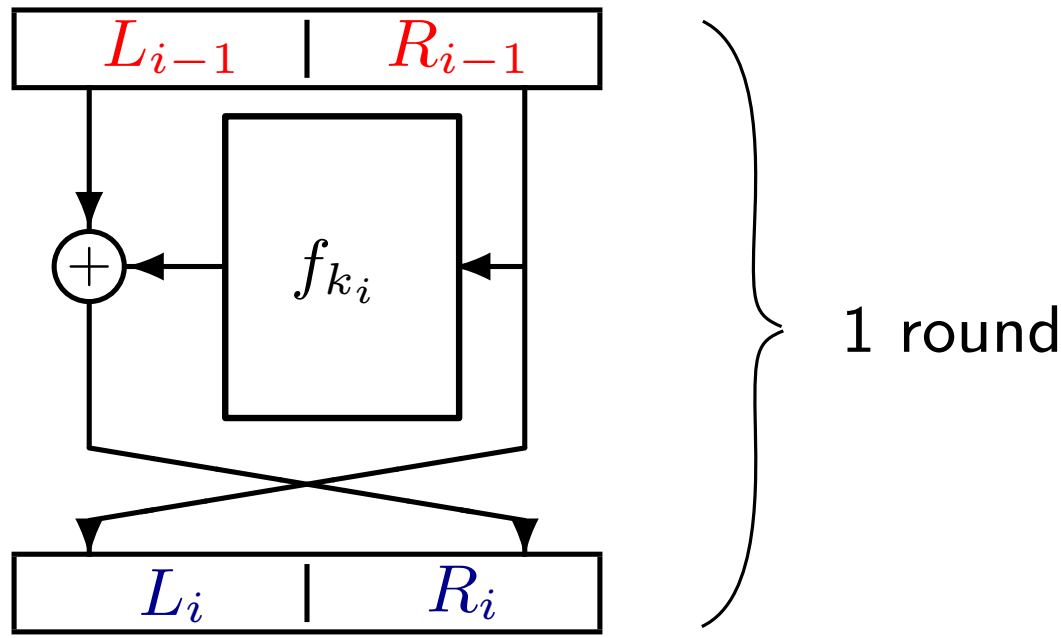
Feistel networks



Feistel: $L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus f_{k_i}(R_{i-1})$

(Must f be invertible?)

Feistel networks

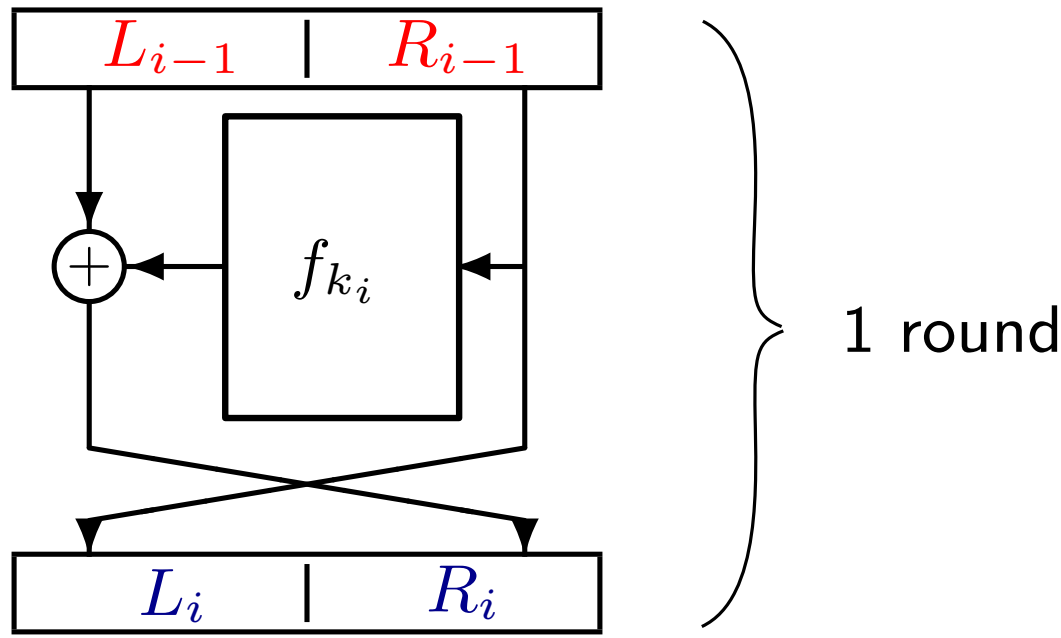


Feistel: $L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus f_{k_i}(R_{i-1})$

(Must f be invertible?)

Decryption for Feistel?

Feistel networks

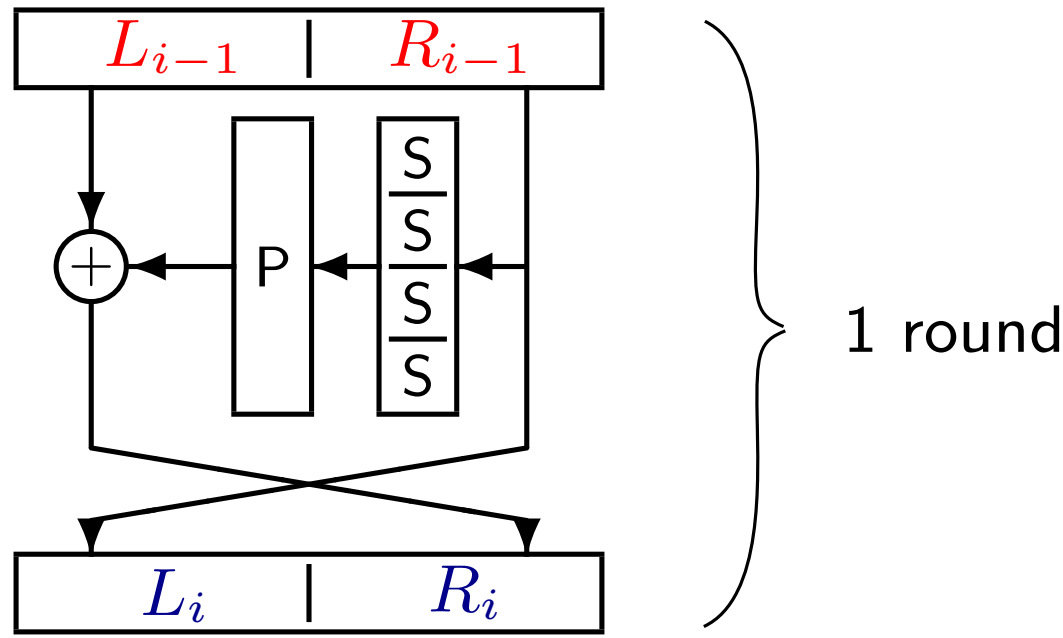


Feistel: $L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus f_{k_i}(R_{i-1})$

(Must f be invertible?)

Decryption for Feistel? $L_{i-1} = R_i \oplus f_{k_i}(L_i), \quad R_{i-1} = L_i$

Feistel networks



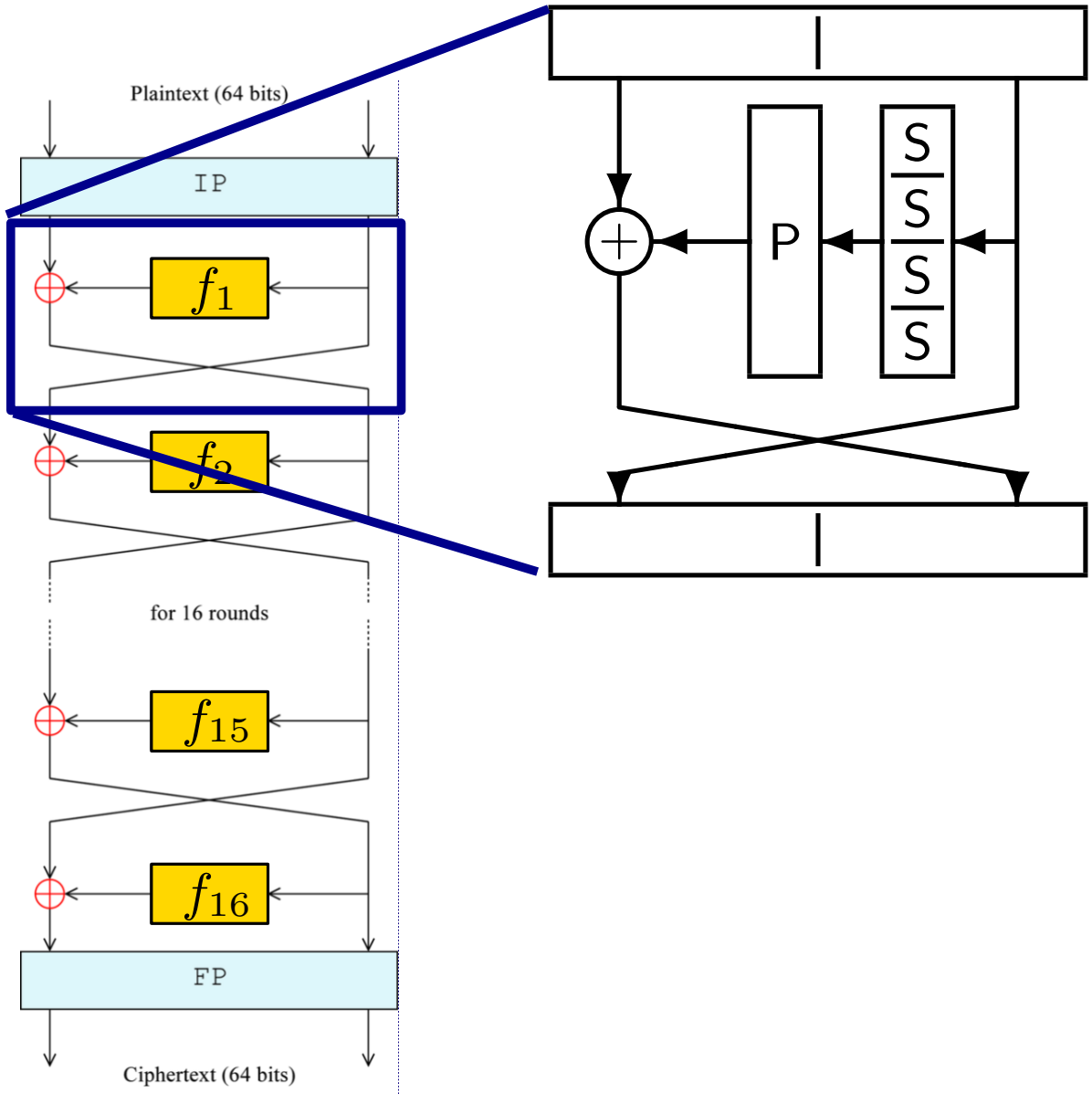
Feistel: $L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus f_{k_i}(R_{i-1})$

(Must f be invertible?)

Decryption for Feistel? $L_{i-1} = R_i \oplus f_{k_i}(L_i), \quad R_{i-1} = L_i$

DES

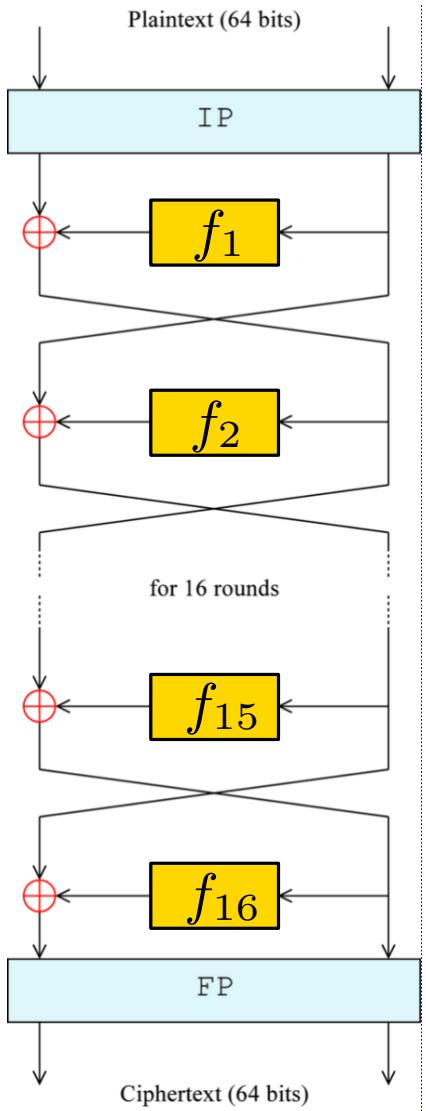
(Data Encryption Standard), 1977



DES

Block size: 64 bits
Key size: 56 bits

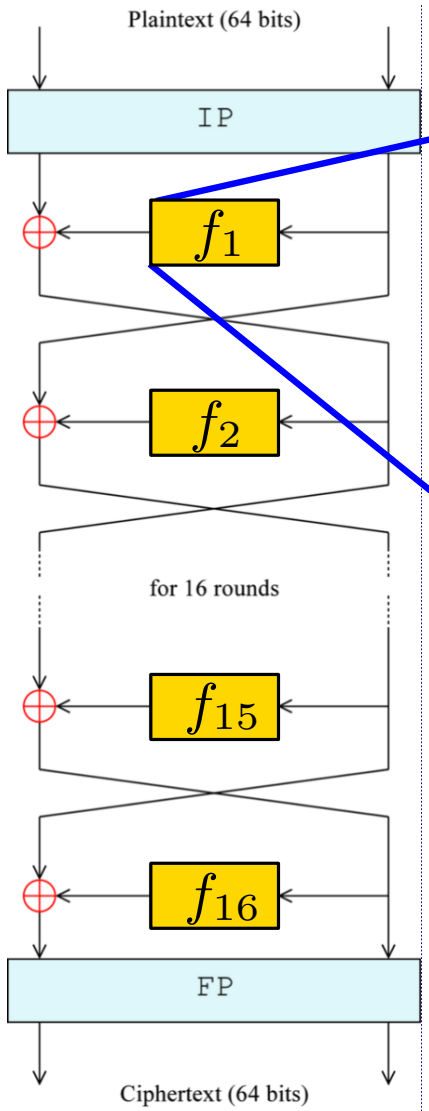
(Data Encryption Standard), 1977

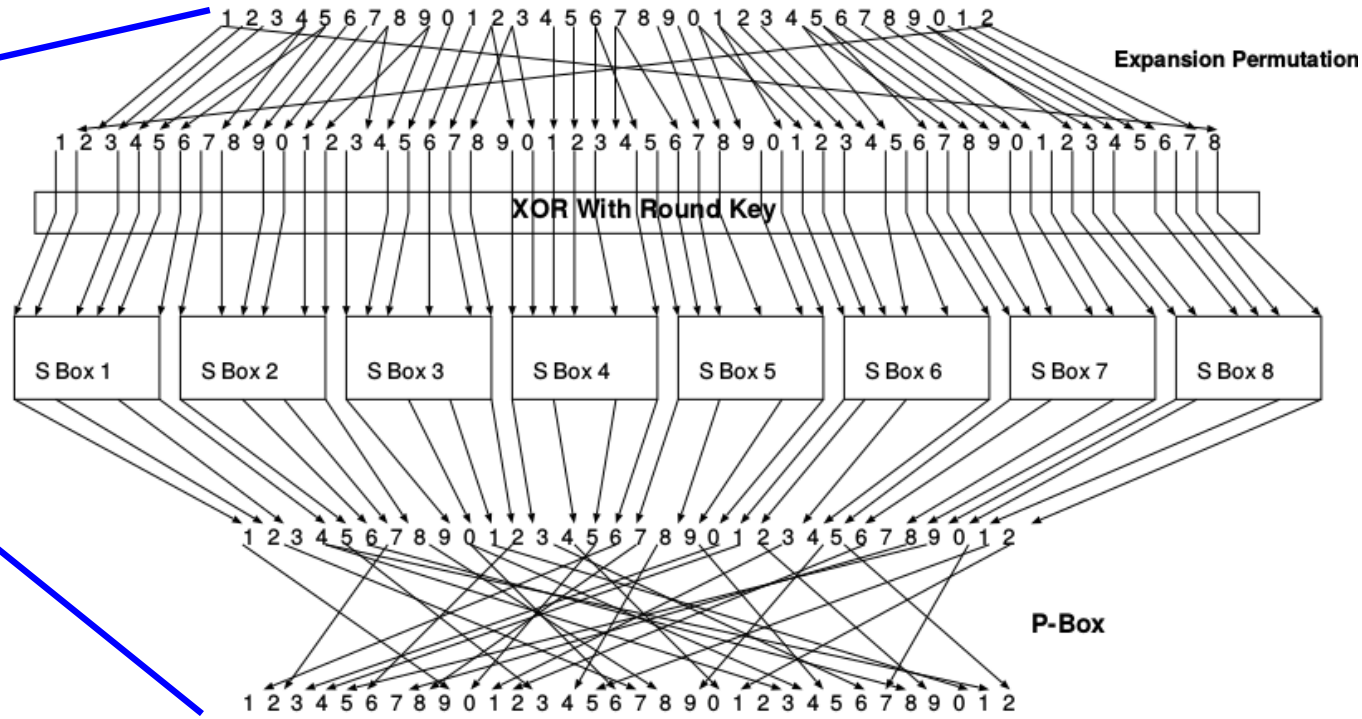


DES

Block size: 64 bits
Key size: 56 bits

(Data Encryption Standard), 1977





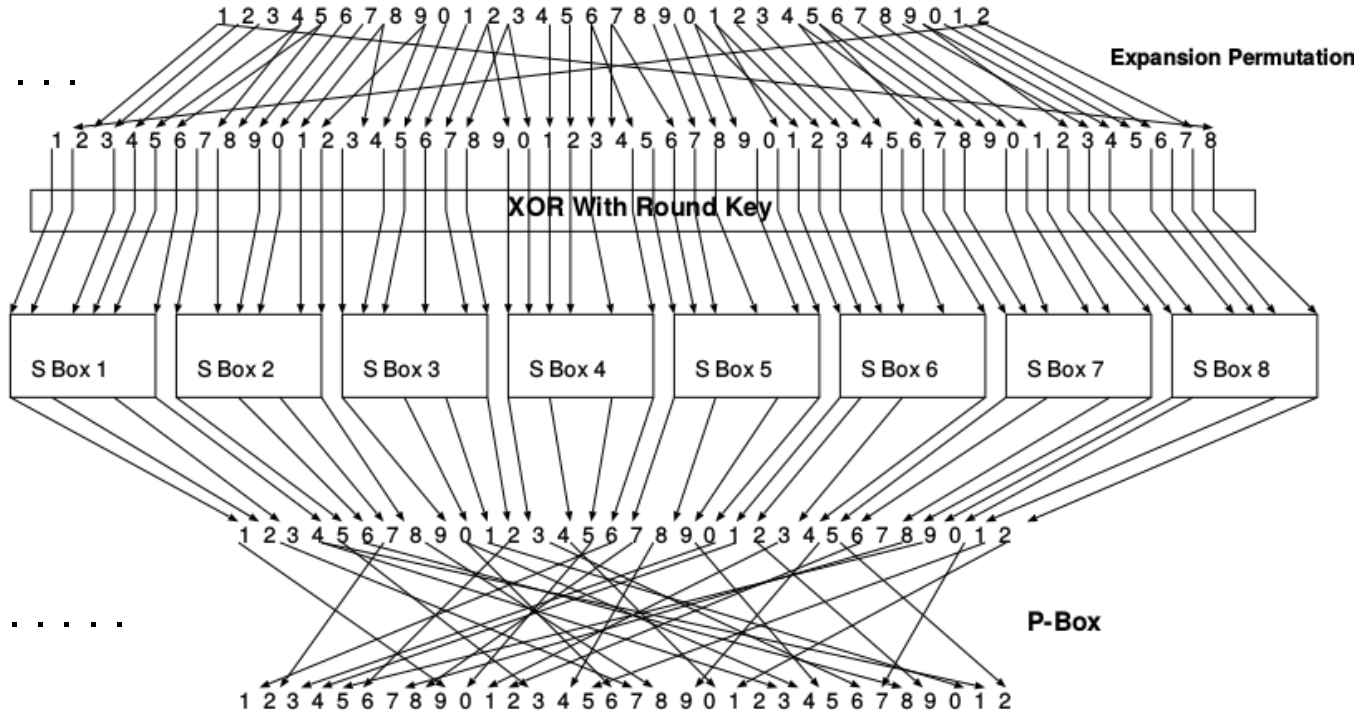
DES

Block size: 64 bits
Key size: 56 bits

(Data Encryption Standard), 1977

Round function:

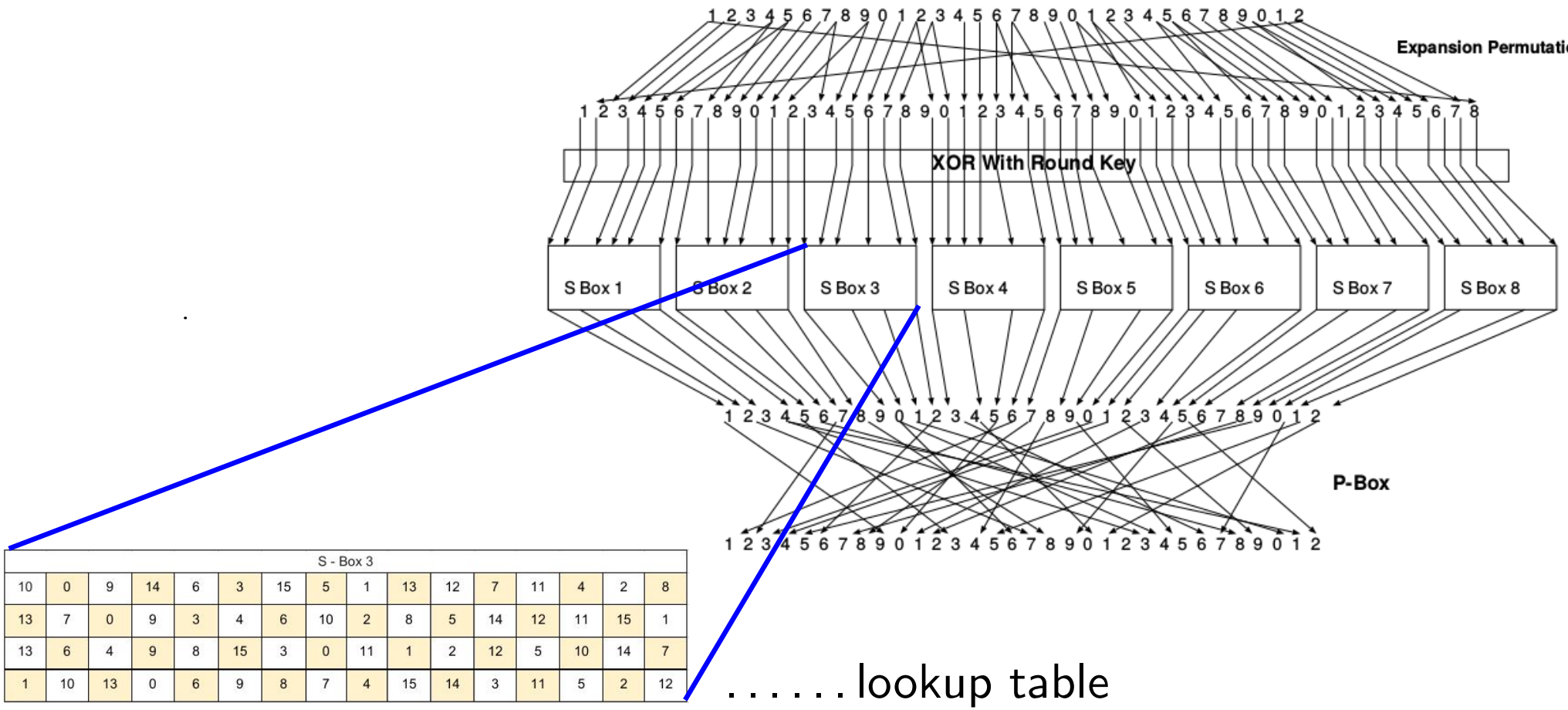
- Expansion $32 \rightarrow 48$ bits . . .
- XOR with subkeys . . .
- 8 S-boxes in parallel. . .
- Permutation . . . . . .



DES

Block size: 64 bits
Key size: 56 bits

(Data Encryption Standard), 1977



S - Box 3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

. . . . . lookup table

Attacks

§7.2.6

Differential cryptanalysis[a] (Biham and Shamir, 1990)

- impractical, but shows (theoretical) weakness

  DES: requires 2^{47} ciphertext for *chosen* plaintexts

[a] 192.124 Symmetric Cryptography 2026S

Attacks

§7.2.6

Differential cryptanalysis[a] (Biham and Shamir, 1990)

- impractical, but shows (theoretical) weakness

  DES: requires 2^{47} ciphertext for *chosen* plaintexts

[a] 192.124 Symmetric Cryptography 2026S

Attacks

§7.2.6

Differential cryptanalysis[a] (Biham and Shamir, 1990)

- impractical, but shows (theoretical) weakness

  DES: requires 2^{47} ciphertext for *chosen* plaintexts

  (known by IBM/NSA in 70's!)

Linear cryptanalysis[a] (Matsui, 1993)

- known-plaintext attack

  DES: requires 2^{43} plaintext/ciphertext pairs

[a] 192.124 Symmetric Cryptography 2026S

Attacks

§7.2.6

Differential cryptanalysis[a] (Biham and Shamir, 1990)

- impractical, but shows (theoretical) weakness

  DES: requires 2^{47} ciphertext for *chosen* plaintexts

  (known by IBM/NSA in 70's!)

Linear cryptanalysis[a] (Matsui, 1993)

- known-plaintext attack

  DES: requires 2^{43} plaintext/ciphertext pairs

Best attack:

exhaustive search

---

[a] 192.124 Symmetric Cryptography 2026S

Attacks

§7.2.6

Differential cryptanalysis[a] (Biham and Shamir, 1990)

- impractical, but shows (theoretical) weakness

  DES: requires 2^{47} ciphertext for *chosen* plaintexts

  (known by IBM/NSA in 70's!)

Linear cryptanalysis[a] (Matsui, 1993)

- known-plaintext attack

  DES: requires 2^{43} plaintext/ciphertext pairs

Best attack:

exhaustive search

- 1997: 69 days (1000's of computers)
- 1999: *Deep Crack* (ASICs): 22 hours
- now: minutes (with preprocessing)

[a] 192.124 Symmetric Cryptography 2026S

DES

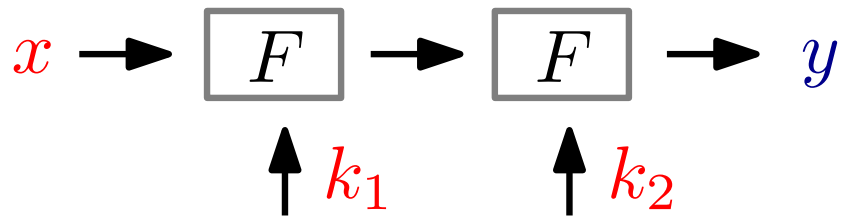
- Design withstood attacks, but *keys too short*
- Brute-force: given x and y find k such that $F_k(x) = y$

DES

- Design withstood attacks, but *keys too short*
- Brute-force: given x and y find k such that $F_k(x) = y$
- **Longer keys!**

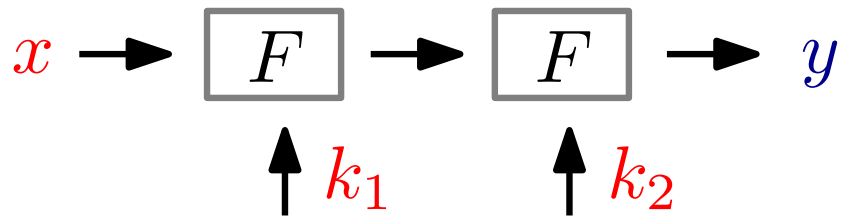
DES

- Design withstood attacks, but *keys too short*
- Brute-force: given x and y find k such that $F_k(x) = y$
- **Longer keys!**
  - Encrypt *twice*?
    Key length?



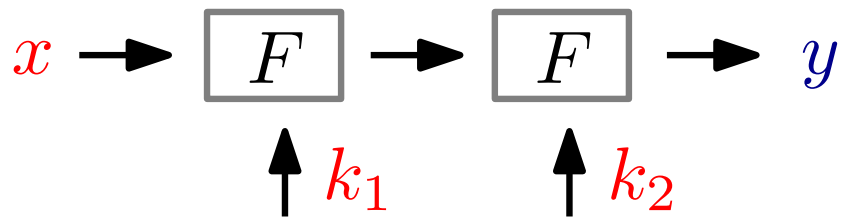
DES

- Design withstood attacks, but *keys too short*
- Brute-force: given x and y find k such that $F_k(x) = y$
- **Longer keys!**
  - Encrypt *twice*?
    Key length? 112 bits



DES

- Design withstood attacks, but *keys too short*
- Brute-force: given x and y find k such that $F_k(x) = y$
- **Longer keys!**
  - Encrypt *twice*?
    Key length? 112 bits

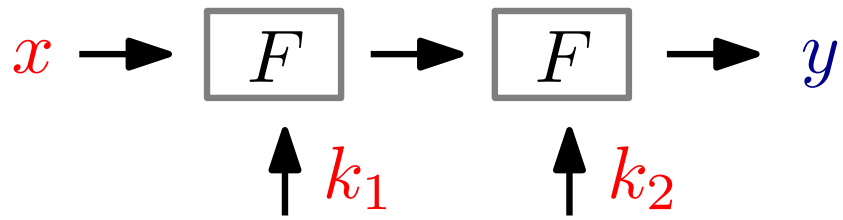


Meet-in-the-middle attack:

Given x and y with $y = F_{k_2}\left(F_{k_1}(x)\right) \Rightarrow F_{k_1}(x) = F_{k_2}^{-1}(y)$

DES

- Design withstood attacks, but *keys too short*
- Brute-force: given x and y find k such that $F_k(x) = y$
- **Longer keys!**
  - Encrypt *twice*?
    Key length? 112 bits



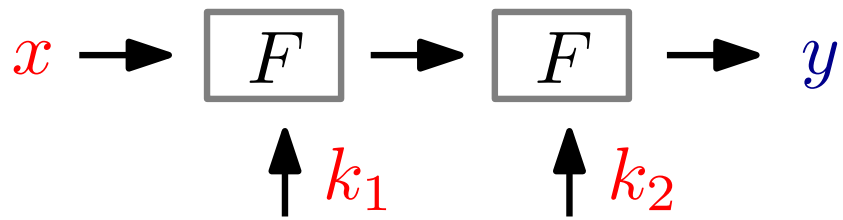
Meet-in-the-middle attack:

Given x and y with $y = F_{k_2}\left(F_{k_1}(x)\right) \Rightarrow F_{k_1}(x) = F_{k_2}^{-1}(y)$

- list $F_{k_1}(x)$ for all k_1
- compute $F_{k_2}^{-1}(y)$ for all k_2
- if match $\rightarrow (k_1, k_2)$

DES

- Design withstood attacks, but *keys too short*
- Brute-force: given x and y find k such that $F_k(x) = y$
- **Longer keys!**
  - Encrypt *twice*?
    Key length? 112 bits



Meet-in-the-middle attack:

Given x and y with $y = F_{k_2}\left(F_{k_1}(x)\right) \Rightarrow F_{k_1}(x) = F_{k_2}^{-1}(y)$

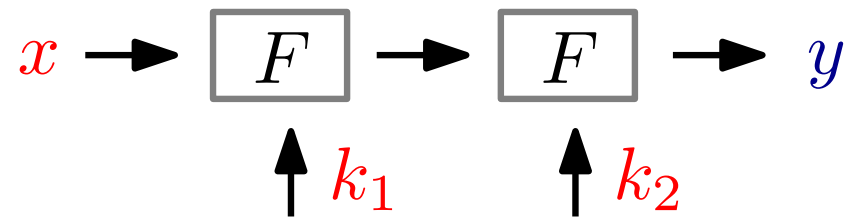
- list $F_{k_1}(x)$ for all k_1
- compute $F_{k_2}^{-1}(y)$ for all k_2
- if match $\rightarrow (k_1, k_2)$

Complexity:

Space: 2^{56}

DES

- Design withstood attacks, but *keys too short*
- Brute-force: given x and y find k such that $F_k(x) = y$
- **Longer keys!**
  - Encrypt *twice*?
    Key length? 112 bits



Meet-in-the-middle attack:

Given x and y with $y = F_{k_2}\left(F_{k_1}(x)\right) \Rightarrow F_{k_1}(x) = F_{k_2}^{-1}(y)$

- list $F_{k_1}(x)$ for all k_1
- compute $F_{k_2}^{-1}(y)$ for all k_2
- if match $\rightarrow (k_1, k_2)$

Complexity:

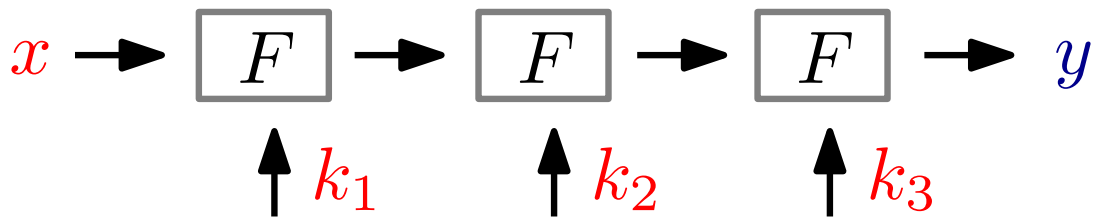
Space: 2^{56}

Time: 2^{57} DES$^{(-1)}$ eval's

$\Rightarrow$ insecure!

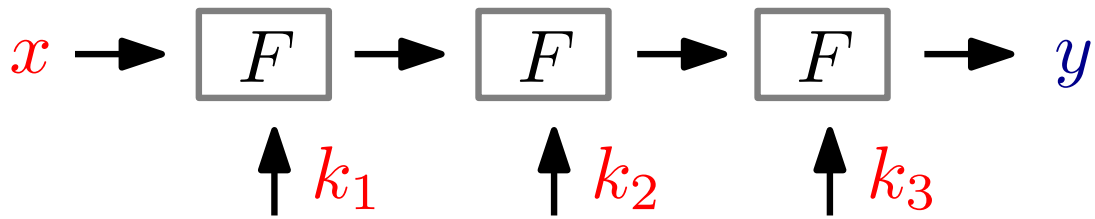
Triple DES

- Design withstood attacks, but *keys too short*
- Brute-force: given x and y find k such that $F_k(x) = y$
- **Longer keys!**
  - Encrypt *3 times*?



Triple DES

- Design withstood attacks, but *keys too short*
- Brute-force: given x and y find k such that $F_k(x) = y$
- **Longer keys!**
  - Encrypt *3 times?*



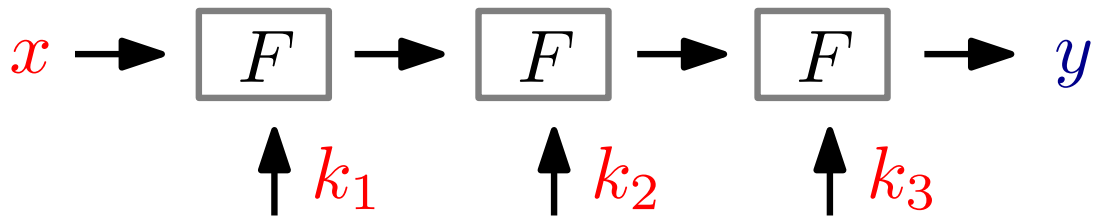
Triple DES: (still used)

$y = \text{DES}_{k_1}(\text{DES}^{-1}_{k_2}(\text{DES}_{k_1}(x)))$

Keys: 112 bits,
Security: 112 (NIST: 80) bits

Triple DES

- Design withstood attacks, but *keys too short*
- Brute-force: given x and y find k such that $F_k(x) = y$
- **Longer keys!**
  - Encrypt *3 times*?



Triple DES: (still used)

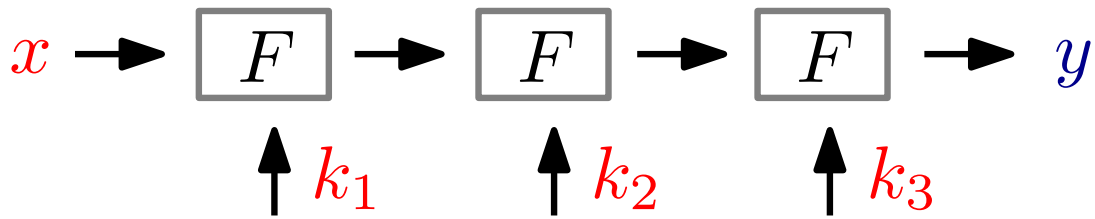
$y = \text{DES}_{k_1}(\text{DES}^{-1}_{k_2}(\text{DES}_{k_1}(x)))$

Keys: 112 bits,
Security: 112 (NIST: 80) bits

- Why only **two** keys?

Triple DES

- Design withstood attacks, but *keys too short*
- Brute-force: given x and y find k such that $F_k(x) = y$
- **Longer keys!**
  - Encrypt *3 times?*



Triple DES: (still used)

$y = \text{DES}_{k_1}(\text{DES}^{-1}_{k_2}(\text{DES}_{k_1}(x)))$

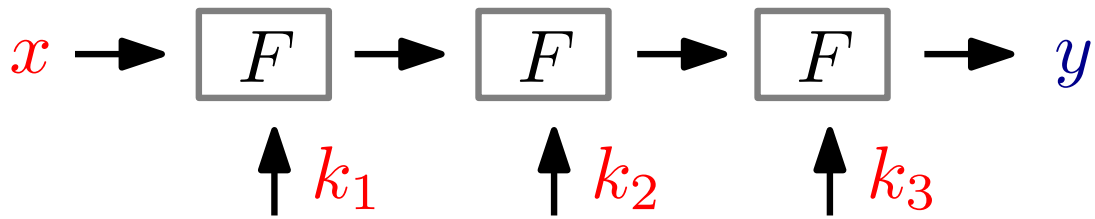
Keys: 112 bits,
Security: 112 (NIST: 80) bits

- Why only **two** keys?

  because meet-in-the-middle with 2^{112} possible anyway

Triple DES

- Design withstood attacks, but *keys too short*
- Brute-force: given x and y find k such that $F_k(x) = y$
- **Longer keys!**
  - Encrypt *3 times?*



Triple DES: (still used)

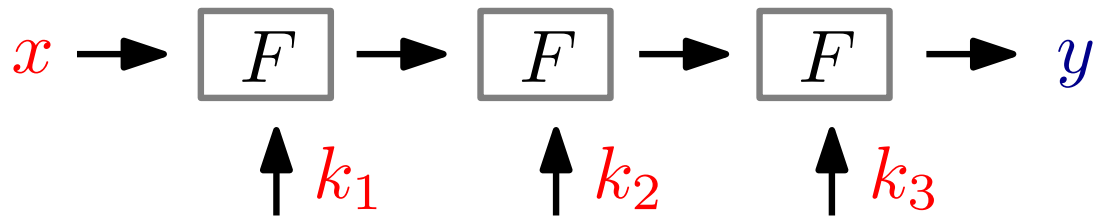
$y = \text{DES}_{k_1}\left(\text{DES}^{-1}_{k_2}(\text{DES}_{k_1}(x))\right)$

Keys: 112 bits,
Security: 112 (NIST: 80) bits

- Why only **two** keys?
  because meet-in-the-middle with 2^{112} possible anyway
- Why DES^{-1}?

Triple DES

- Design withstood attacks, but *keys too short*
- Brute-force: given x and y find k such that $F_k(x) = y$
- **Longer keys!**
  - Encrypt *3 times?*



Triple DES: (still used)

$y = \text{DES}_{k_1}(\text{DES}^{-1}_{k_2}(\text{DES}_{k_1}(x)))$

Keys: 112 bits,
Security: 112 (NIST: 80) bits

- Why only **two** keys?

  because meet-in-the-middle with 2^{112} possible anyway

- Why DES^{-1}?

  backwards compatibility: $k_1 = k_2 \Rightarrow$ DES

AES

DES versus AES

DES

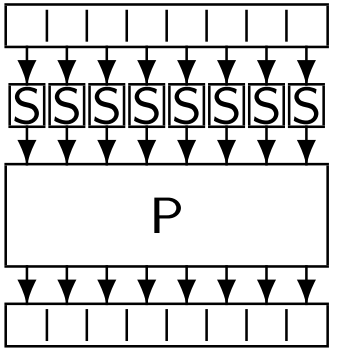
- S-P network, iterated cipher,
  Feistel structure
- 64-bit blocks
  56-bit keys
- 8 different S-boxes
- *non*-invertible round function
- optimized for hardware
  implementations
- secret design choice

DES versus AES

DES

- S-P network, iterated cipher, Feistel structure
- 64-bit blocks
  56-bit keys
- 8 different S-boxes
- *non*-invertible round function
- optimized for hardware implementations
- secret design choice

AES (Rijndael)



- S-P network,
  iterated cipher
- 128-bit blocks
  128- or 192- or 256-bit keys
- 1 S-box
- invertible round function
- optimized for "byte-orientated" implementations
- open design and evaluation
  (NIST competition,
  standardized in 2001)

AES

(Advanced Encryption Standard) – Rijndael

Key length: 128 or 192 or 256 bits

Number of rounds: 10 or 12 or 14

Rijmen, Daemen

AES

(Advanced Encryption Standard) – Rijndael

Key length: 128 or 192 or 256 bits

Number of rounds: 10 or 12 or 14

Round transformation consists of

- Byte substitution
- Shift rows
- Mix columns
- Round key addition

Rijmen, Daemen

AES

(Advanced Encryption Standard) – Rijndael

Key length: 128 or 192 or 256 bits

Number of rounds: 10 or 12 or 14

Round transformation consists of

- Byte substitution ← confusion
- Shift rows ← diffusion
- Mix columns ← diffusion
- Round key addition

Rijmen, Daemen

AES

(Advanced Encryption Standard) – Rijndael

Key length: 128 or 192 or 256 bits

Number of rounds: 10 or 12 or 14

Rijmen, Daemen

Round transformation consists of

- Byte substitution ← confusion
- Shift rows ← diffusion
- Mix columns ← diffusion
- Round key addition

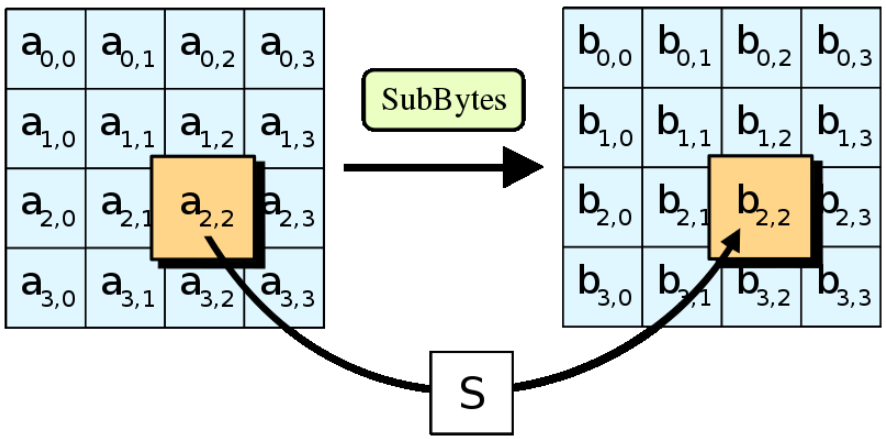
Plaintext block: 16 bytes $x_0, \ldots, x_{15}$

$$\begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix}$$

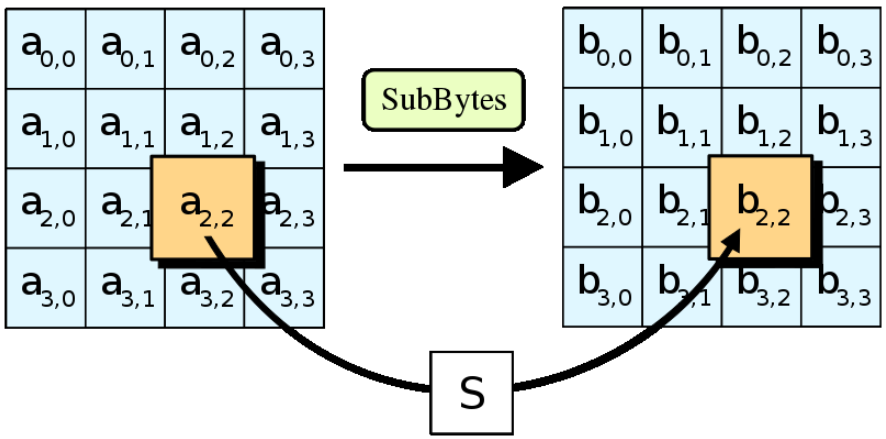
round key: $k_0, \ldots, k_{15}$

$$\begin{bmatrix} k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ k_8 & k_9 & k_{10} & k_{11} \\ k_{12} & k_{13} & k_{14} & k_{15} \end{bmatrix}$$

Rijndael – Substitution / Permutation



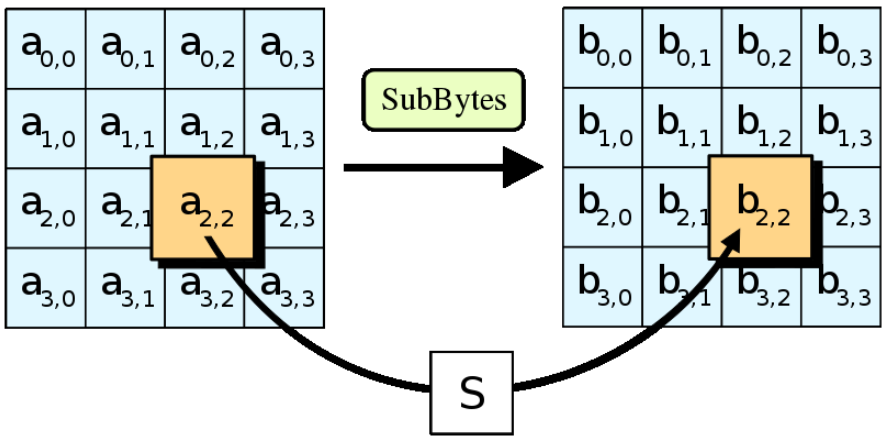
Rijndael – Substitution / Permutation



AES S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Rijndael – Substitution / Permutation



input byte: $b_0 b_1 b_2 b_3 \; b_4 b_5 b_5 b_7$

row column

AES S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Rijndael – Substitution / Permutation

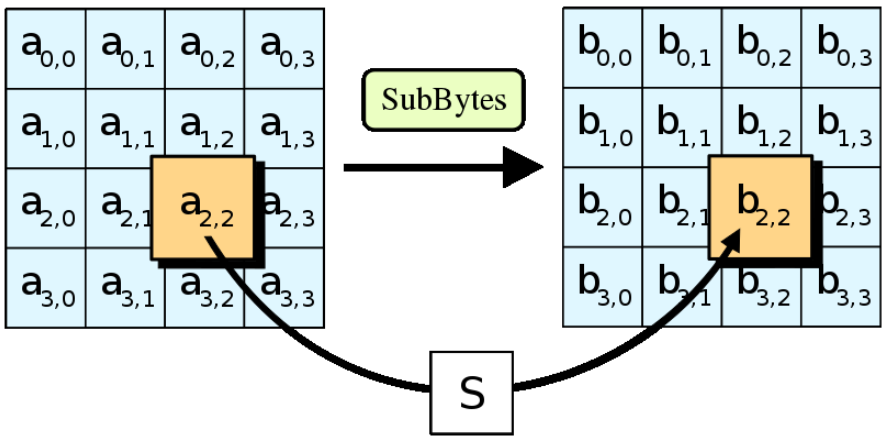
AES S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

AES S-box

04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
18	96	05	9a	07	12	80	e2	eb	27	b2	75
1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
22	2a	90	88	46	ee	b8	14	de	5e	0b	db
49	06	24	5c	c2	d3	ac	62	91	95	e4	79
8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Rijndael – Substitution / Permutation



Definition of S-Box:

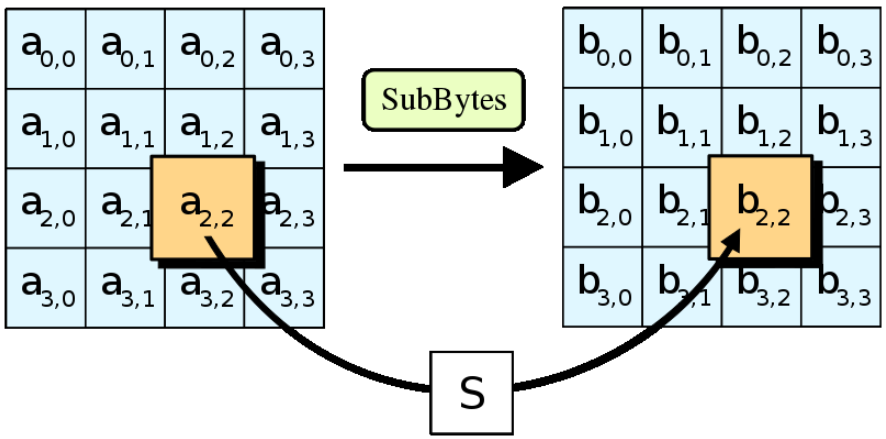
interpret bytes as elements from

$$\mathrm{GF}(2^8) = \mathbb{Z}_2[X]/(X^8+X^4+X^3+X+1)$$

AES S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Rijndael – Substitution / Permutation



AES S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

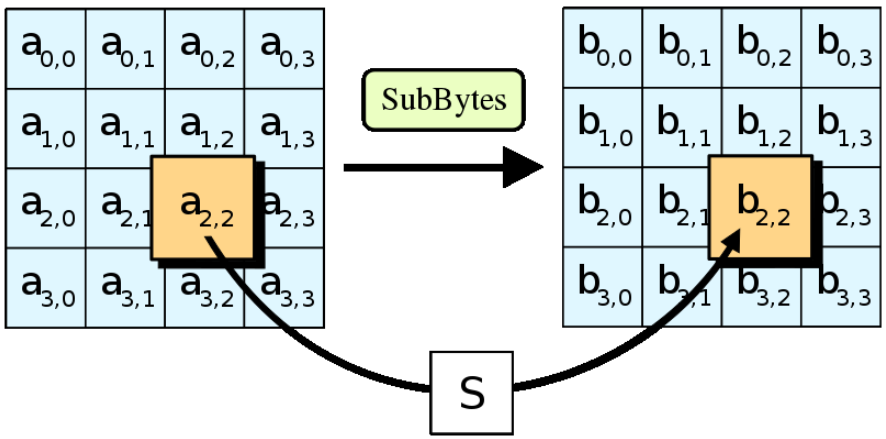
Definition of S-Box:

interpret bytes as elements from

$$\mathrm{GF}(2^8) = \mathbb{Z}_2[X]/(X^8+X^4+X^3+X+1)$$

$$a = (a_0, a_1, \ldots, a_7) = a_7X^7 + a_6X^6 + a_5X^5 + a_4X^4 + a_3X^3 + a_2X^2 + a_1X + a_0$$

Rijndael – Substitution / Permutation



AES S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

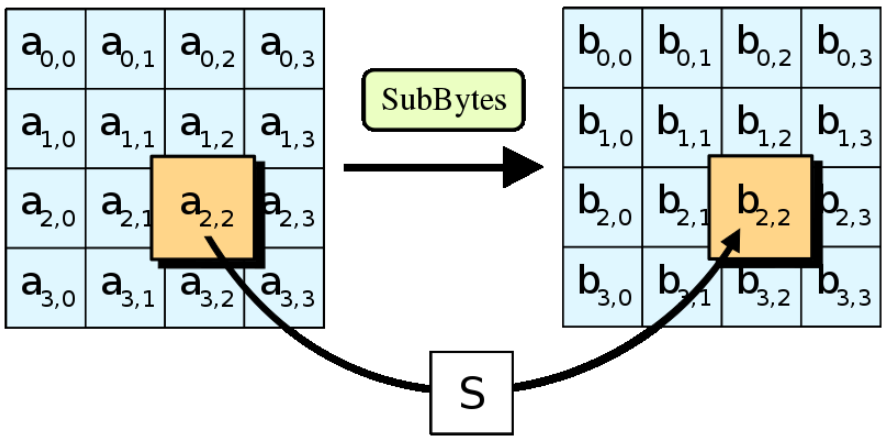
Definition of S-Box:

interpret bytes as elements from

$$\mathrm{GF}(2^8) = \mathbb{Z}_2[X]/(X^8+X^4+X^3+X+1)$$

- let a' be inverse of a in $\mathrm{GF}(2^8)$

Rijndael – Substitution / Permutation



AES S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Definition of S-Box:

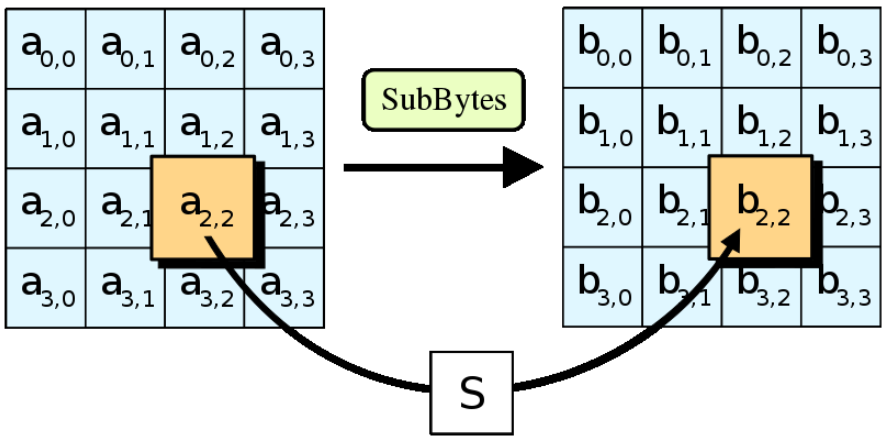
interpret bytes as elements from

$$\mathrm{GF}(2^8) = \mathbb{Z}_2[X]/(X^8+X^4+X^3+X+1)$$

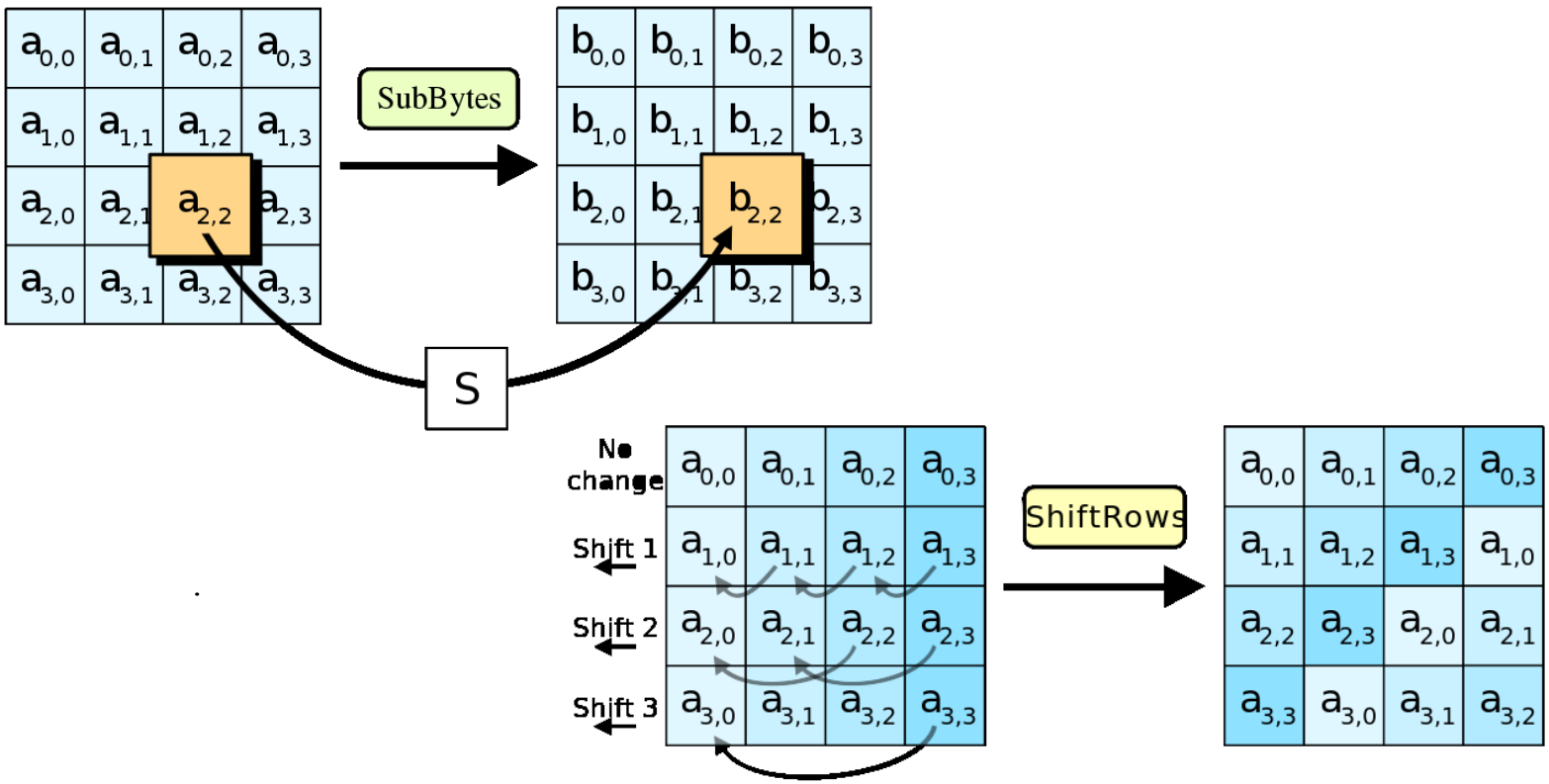
- let a' be inverse of a in $\mathrm{GF}(2^8)$
  - define b as

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a'_0 \\ a'_1 \\ a'_2 \\ a'_3 \\ a'_4 \\ a'_5 \\ a'_6 \\ a'_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

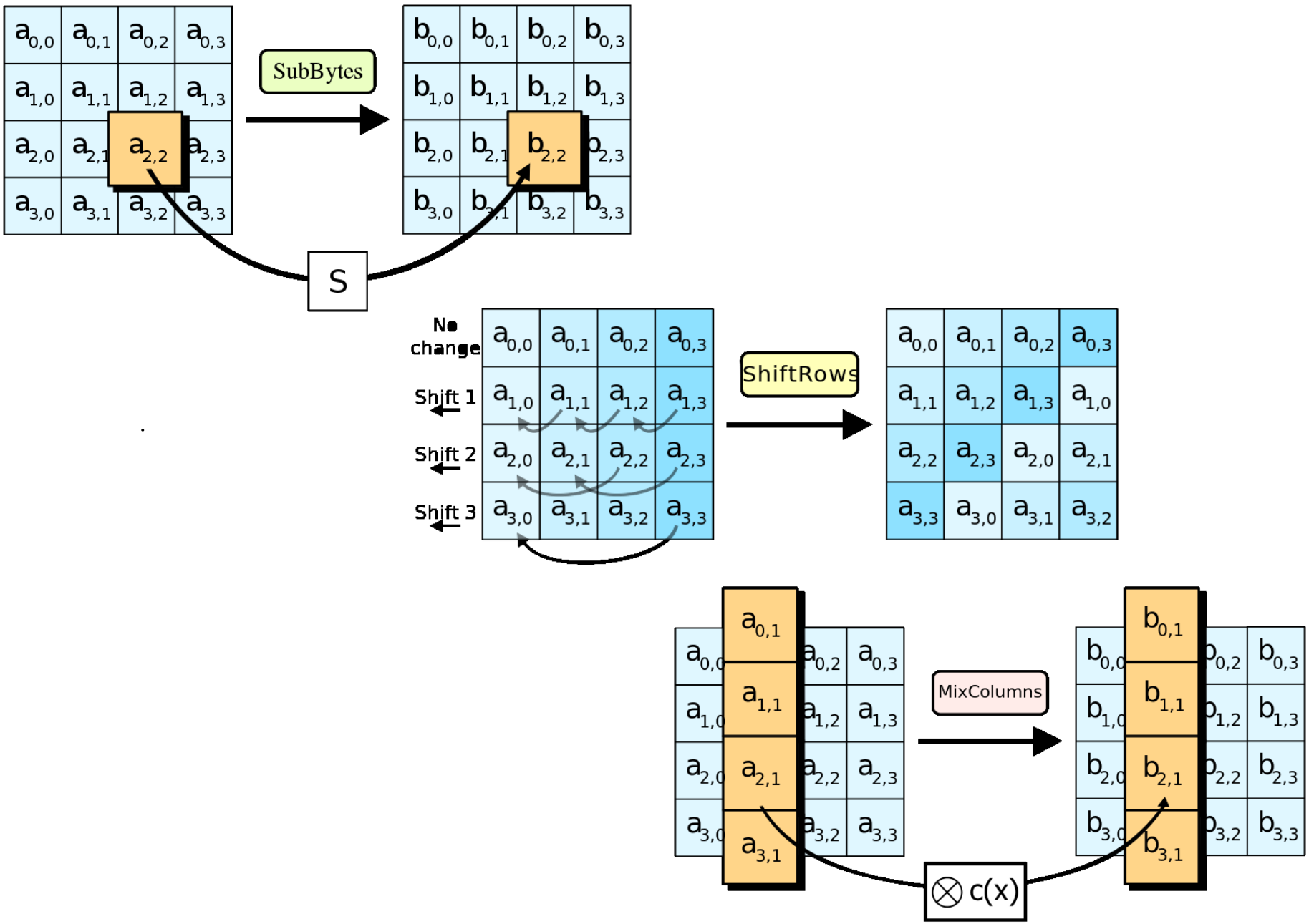
Rijndael – Substitution / Permutation



Rijndael – Substitution / Permutation



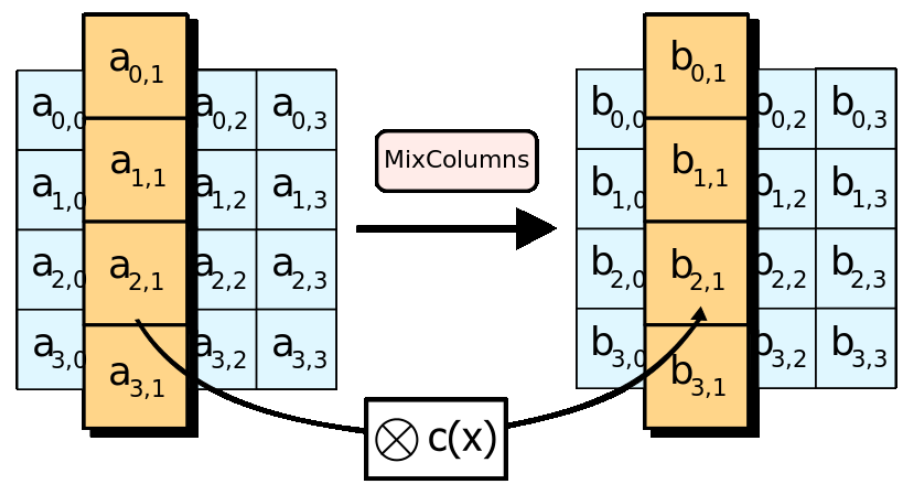
Rijndael – Substitution / Permutation



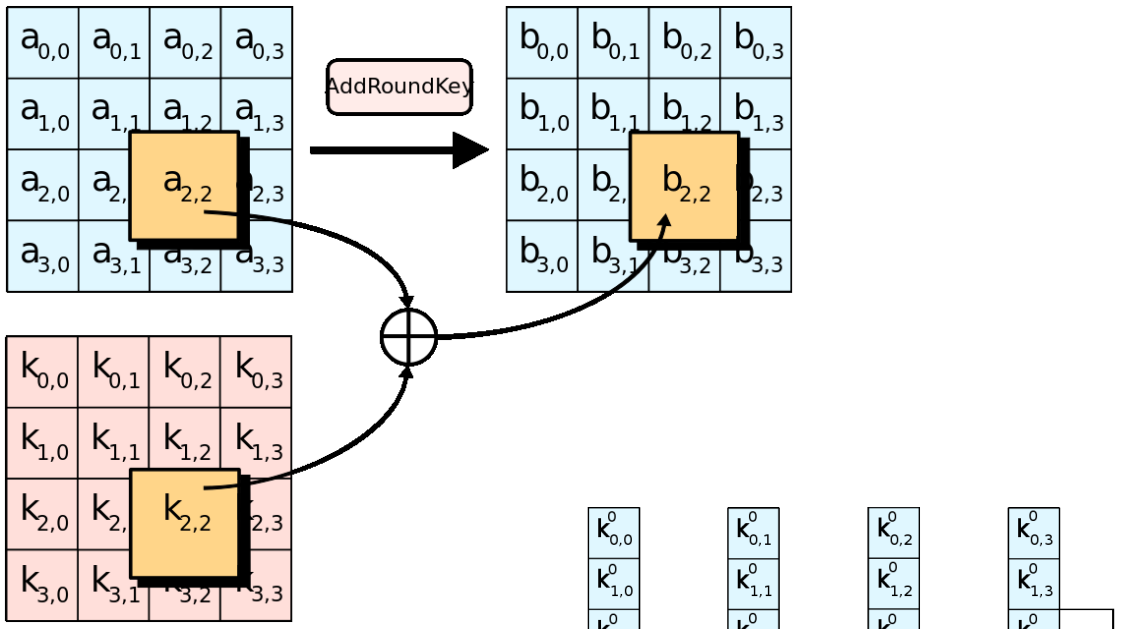
Rijndael – Substitution / Permutation

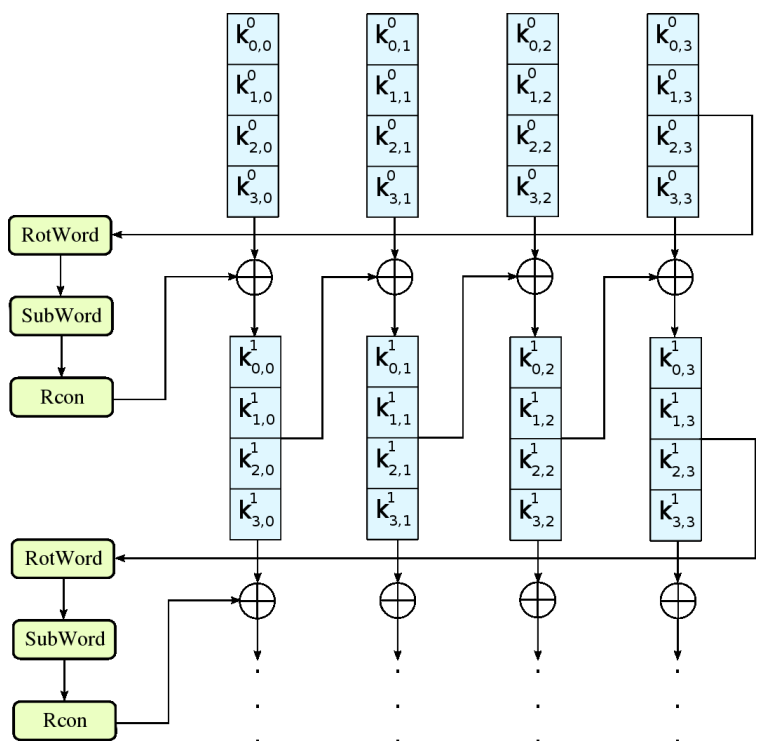
$$\begin{bmatrix} b_{0,j} \\ b_{1,j} \\ b_{2,j} \\ b_{3,j} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_{0,j} \\ a_{1,j} \\ a_{2,j} \\ a_{3,j} \end{bmatrix} \qquad 0 \leq j \leq 3$$

$$a_{i,j}, b_{i,j} \in \mathrm{GF}(2^8)$$



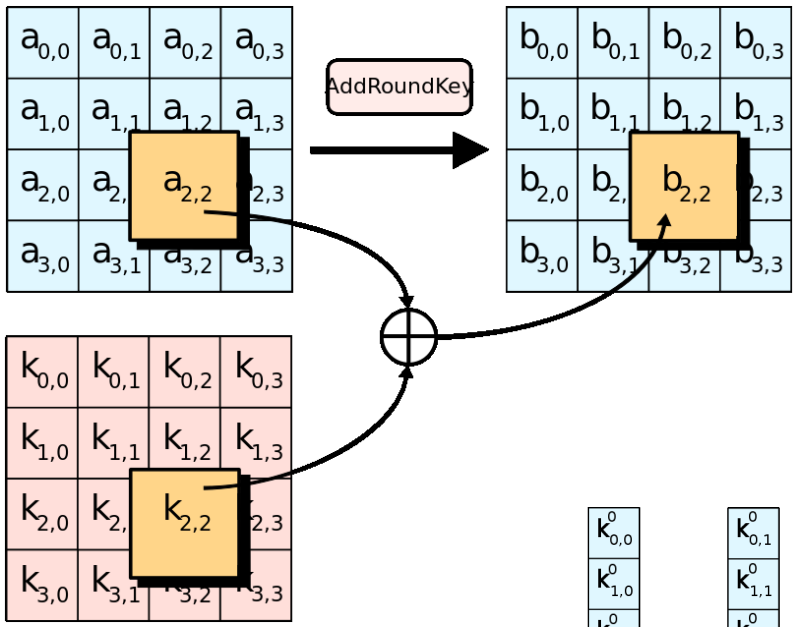
Rijndael – Round-key addition

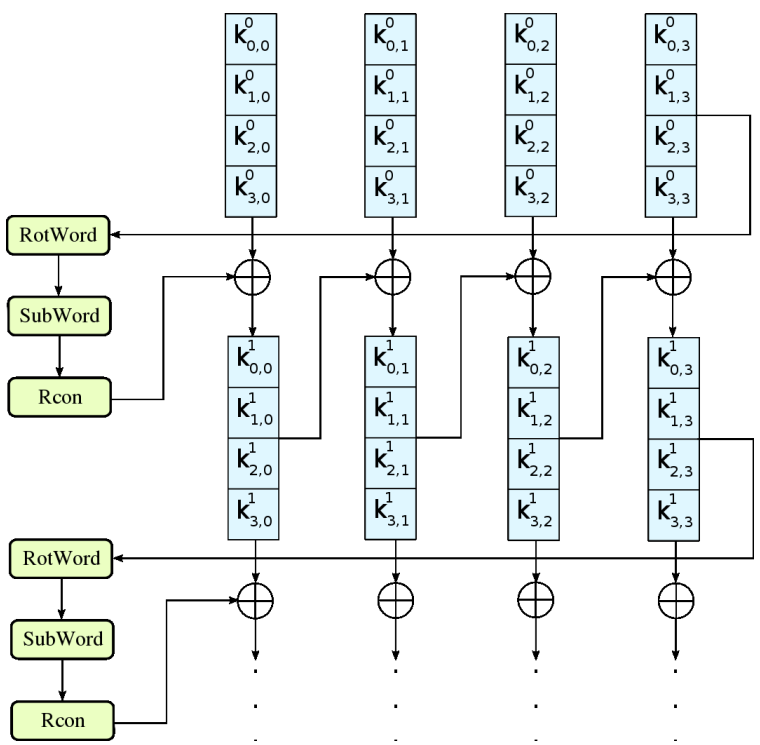




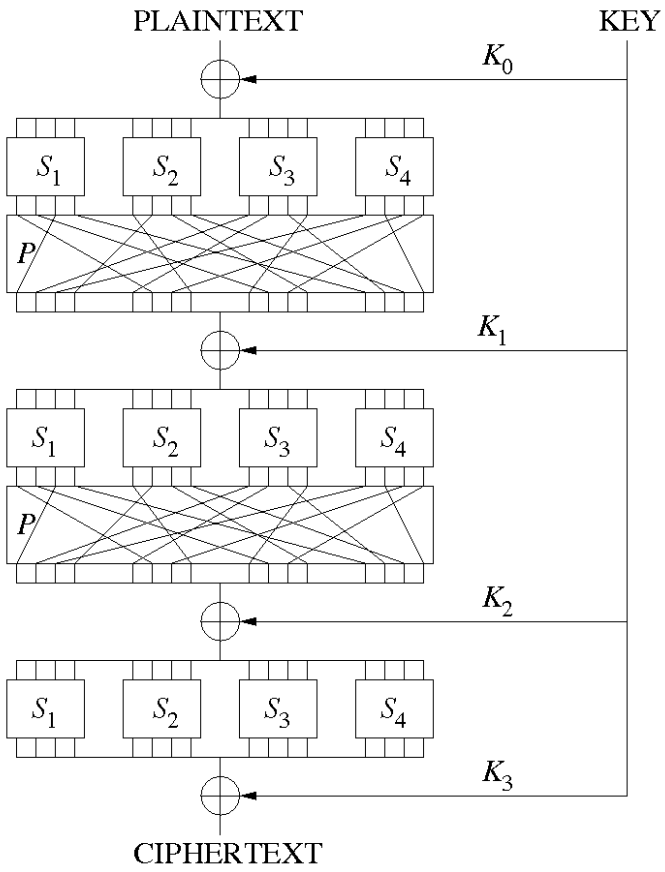
Derivation of round keys

Rijndael – Round-key addition





Derivation of round keys



Overall structure

Attacks

Cryptanalysis

- 2009, related-key attack with complexity $2^{99.5}$
- 2011, biclique attack (MIM) faster than brute force,
  $2^{126.2}$ AES calls to recover an AES-128 key

Introduction to Cryptography

(Lecture 4: Stream ciphers and computational security)

Elena Andreeva

TUW 2025W

Stream ciphers

§7.1

Stream ciphers

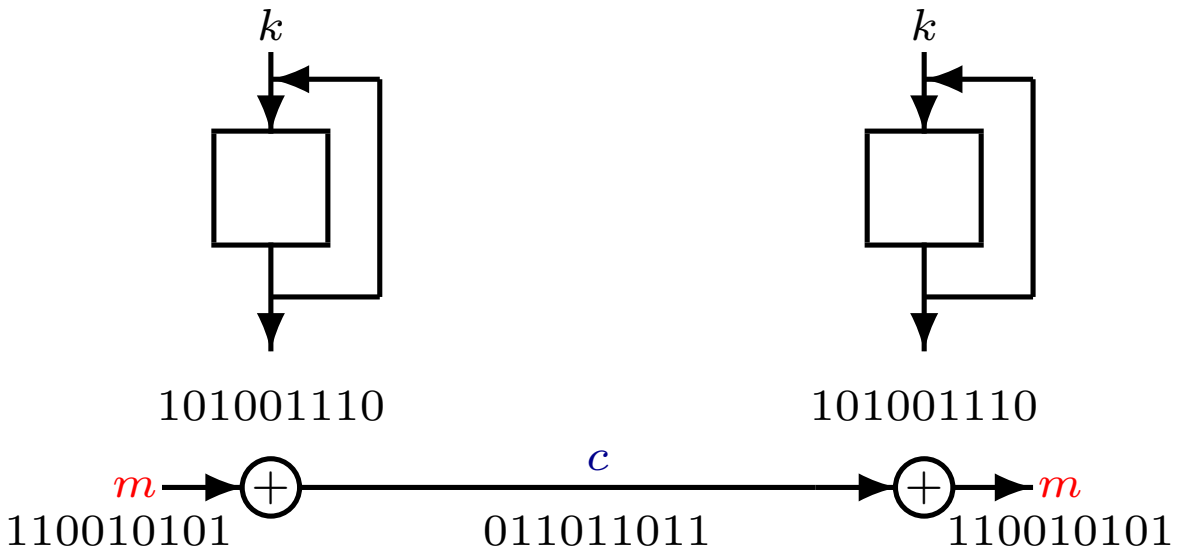
Key stream generator:

- takes a random string (the key)
- produces a potentially infinite string that *seems* random

Encryption: like one-time pad, but:

- replace (long) random key by

  a **pseudo-random** string generated by a random **seed**



Encryption: $c_i = m_i \oplus s_i$

key stream

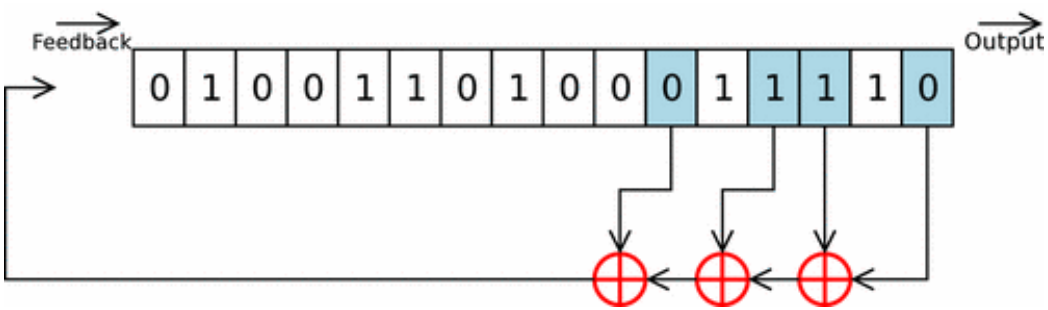
Decryption: $m_i = c_i \oplus s_i$

Stream ciphers

Key stream generator:

- takes a random string (the key)
- produces a potentially infinite string that *seems* random

Linear-feedback shift register (LFSR):



$$
\begin{array}{cccccccccccccccc}
0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1
\end{array}
$$

- very efficient in hardware
- linear $\Rightarrow$ attacks

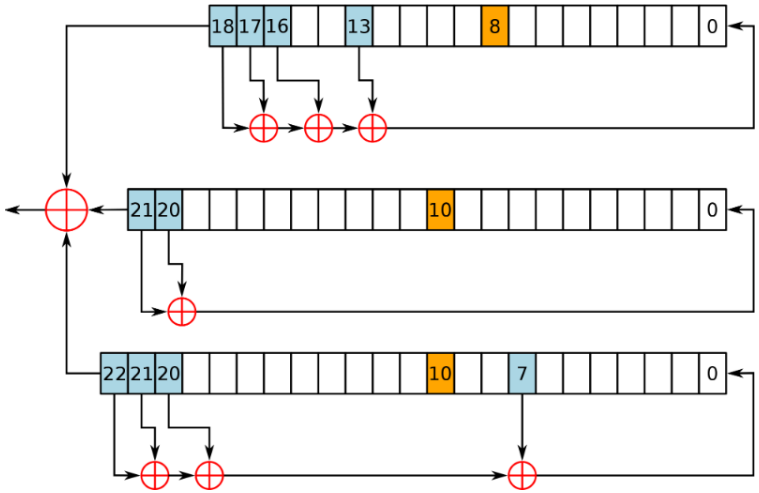
Stream ciphers

Key stream generator:

- takes a random string (the key)
- produces a potentially infinite string that *seems* random

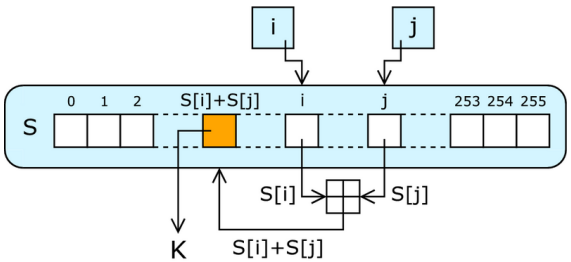
Implementation:

- Combination of LFSRs
  (e.g. A5/1 (GSM)) or



. . . broken

- RC4 (used in WEP) or



- Block cipher mode of operation ($\rightarrow$ later)

Computational security

§3.1

Modern Cryptography

Provable Security (a.k.a. reductionist security):

- **Definitions:** What is *security goal*, what is *threat model* (e.g. indistinguishability under chosen-plaintext attack)
- **Assumptions:** Computational assumptions (e.g. factoring large integers is hard)
- **Security proof:** Mathematical proof that construction achieves *definition* under *assumptions*.

Private-key encryption

Syntax of encryption schemes:

$\mathcal{K}$ ... key space
$\mathcal{M}$ ... plaintext space
$\mathcal{C}$ ... ciphertext space

Gen: $\to \mathcal{K}$	*key-generation*	(randomized)
Enc: $\mathcal{K} \times \mathcal{M} \to \mathcal{C}$	*encryption algorithm*	(possibly randomized)
Dec: $\mathcal{K} \times \mathcal{C} \to \mathcal{M}$	*decryption algorithm*	(deterministic)

Correctness: $\forall\, k \in \mathcal{K}\ \forall\, m \in \mathcal{M} : \mathsf{Dec}_k\big(\mathsf{Enc}_k(m)\big) = m$

Security?

- Threat model (adversary's capabilities)
- Adversary's goal

previously...

Kerckhoffs' Principle: The adversary knows the scheme

Auguste Kerckhoffs: *La cryptographie militaire* (1883)

- Adversary's **goals:**
  - ~~Find the key?~~
  - ~~Recover the plaintext~~
  - ~~*Guess* a single letter of the plaintext~~
  - Obtain *any* information about the plaintext

- Adversary's **power:**
  - Sees ciphertexts (one/many)
  - Has seen plaintext/ciphertext pairs
  - Has chosen the plaintexts

    ...and can ask for *decryption*

previously...

Kerckhoffs' Principle: The adversary knows the scheme

Auguste Kerckhoffs: *La cryptographie militaire* (1883)

- Adversary's **goals:**
  - ~~Find the key?~~
  - ~~Recover the plaintext~~
  - ~~*Guess* a single letter of the plaintext~~
  - Obtain *any* information about the plaintext

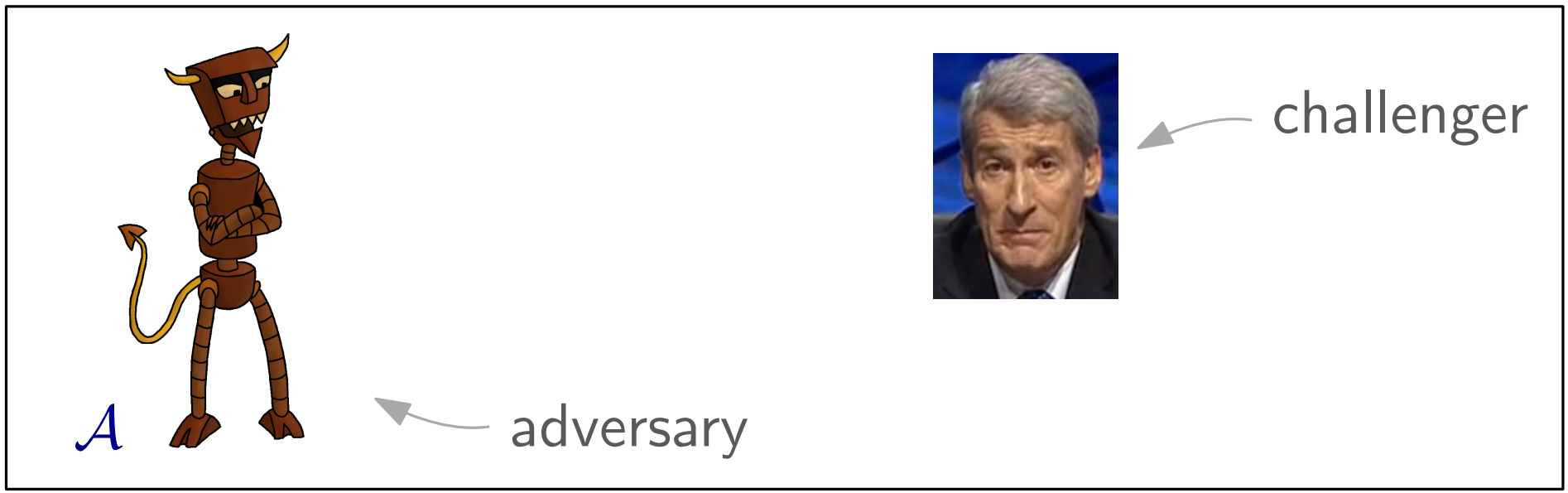
- Adversary's **power:**
  - *ciphertext-only attack ("eavesdropping")*
  - Has seen plaintext/ciphertext pairs
  - *chosen-plaintext attack*
  - *chosen-ciphertext attack*

Indistinguishability

The adversarial *indistinguishability* experiment

$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$

where $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$



Indistinguishability

The adversarial *indistinguishability* experiment

$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$

where $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$

$\mathcal{A}$

$m_0, m_1 \in \mathcal{M}$

c

b'

$k \leftarrow \mathsf{Gen}$

$b \leftarrow \{0,1\}$

$c \leftarrow \mathsf{Enc}_k(m_b)$

$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1$ iff $b = b'$ (and 0 otherwise)

Def. 2.6. $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* **perfectly indistinguishable** *if for every* $\mathcal{A}$*:*

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1\right] = \frac{1}{2}$$

Indistinguishability

Def. 2.3. Π is **perfectly secret** if for every probability distribution over $\mathcal{M}$, every $m \in \mathcal{M}$ and every ciphertext c with $\Pr[C = c] > 0$:

$$\Pr\left[M = m \,\middle|\, C = c\right] = \Pr\left[M = m\right]$$

Lemma 2.7. *An encryption scheme is perfectly secret*

if and only if

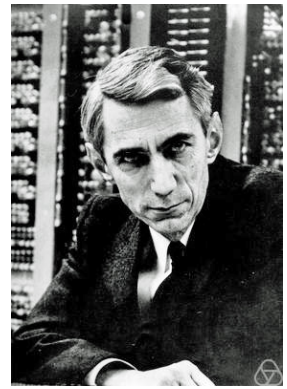
it is perfectly indistinguishable.

$\Leftrightarrow$

Def. 2.6. $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* **perfectly indistinguishable** *if for every* $\mathcal{A}$*:*

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1\right] = \frac{1}{2}$$

Computational Security



Perfect secrecy requires: (Shannon)

- key as **long** as message
- key can only be used **once** $\Rightarrow$ not practical

Computational security relaxes this:

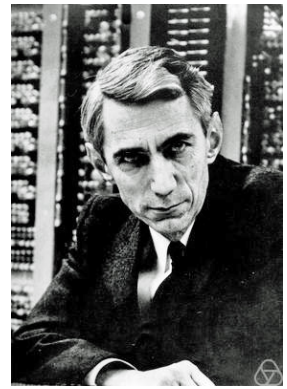
1. only considers *computationally bounded* adversaries

Example: brute-force key space (one key per clock cycle)

- PC $\approx 2^{32} \cdot 2^{25} = 2^{57}$ keys p.a.
- supercomputer $\approx 2^{60} \cdot 2^{25}$ p.a.
- . . . since Big Bang $\approx 2^{85} \cdot 2^{33} = 2^{118}$

Year	Supercomputer	Rmax (TFlop/s)	Location
2022	Cray/HPE Frontier	1,102,000.0	Oak Ridge, U.S.

Computational Security



Perfect secrecy requires: (Shannon)

- key as **long** as message
- key can only be used **once**

Computational security relaxes this:

2. adversaries could succeed with *very small probability*

Example: Security fails with probability 2^{-60}

- sender is struck by lightning: more probable
- 1 event per second: wait 100 000 000 000 years

Computational Security

Concrete approach: "any adversary running in time t can succeed with probability at most ε"

Example: $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is (t, ε)-*indistinguishable* if for all adversaries $\mathcal{A}$ running in time at most t, we have:

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1\right] \leq \frac{1}{2} + \varepsilon$$

- $\boxed{+}$ relevant for real-world applications
- $\boxed{-}$ security cannot be ajusted
- $\boxed{-}$ not robust (which computational model?)

Computational Security

Asymptotic approach:

- assumes **security parameter** $n \in \mathbb{N}$, known to everyone

  (think key length; bigger $n \Rightarrow$ more secure)

- running times (users and $\mathcal{A}$) and success probability are *functions* of n

Computational Security

$\boxed{t}$ Adversaries (and users) assumed to run in **probabilistic polynomial time in** n

- there exists polynomial p s.t. the adversary, on input of length n, runs in time at most $p(n)$
- adversary (like Gen) can make random choices

Computational Security

$\boxed{t}$ Adversaries (and users) assumed to run in
probabilistic polynomial time in n

Problem with **concrete** security:

- how to count running time t?
  - Turing machines?
  - random access machines?
  - which CPU; how many?

Strong Church-Turing thesis: every physically realizable computation can be simulated on a Turing machine *with at most polynomial slowdown*.

$\Rightarrow$ All that can be *"realistically"* computed
can be computed in (probabilistic) polynomial time

Computational Security

$\boxed{t}$ Adversaries (and users) assumed to run in **probabilistic polynomial time in** n

$\boxed{\varepsilon}$ Security holds except with probability **negligible in** n

Def. 3.4. $f : \mathbb{N} \to \mathbb{R}^+$ is *negligible* if for every positive polynomial p there exists N s.t. for all $n > N$: $f(n) < \frac{1}{p(n)}$

Examples:

- 2^{-n}
- $2^{-\sqrt{n}}$
- $n^{-\log(n)}$

$\Big\}$ $< n^{-c}$ for any c for sufficiently large n

Computational Security

$\boxed{t}$ Adversaries (and users) assumed to run in **probabilistic polynomial time in** n

$\boxed{\varepsilon}$ Security holds except with probability **negligible in** n

Def. 3.4. $f : \mathbb{N} \to \mathbb{R}^+$ is *negligible* if for every positive polynomial p there exists N s.t. for all $n > N$: $f(n) < \frac{1}{p(n)}$

Closure properties:

- $p(\cdot), q(\cdot)$ poly., then $p(\cdot) \times q(\cdot)$ and $p(q(\cdot))$ again poly.
- $p(\cdot)$ poly. and $\mu(\cdot)$ negl., then $p(\cdot) \times \mu(\cdot)$ negl.

Computational Security

- Increase $n \Rightarrow$ running time of algorithms increases
  moderately (polynomially)
  but adversary's success prob. decreases *fast*

Example 3.3. Suppose:

- Honest parties run in $10^6 \cdot n^2$ cycles
- Adversary, after $10^8 \cdot n^4$ cycles, succeeds with prob. $\leq 2^{-n/2}$

Assume everyone has 2GHz CPUs and $n = 80$

- Honest parties run in 3.2 sec.
- Adversary, after 3 weeks, succeeds with prob. 2^{-40}

Assume everyone has 8GHz CPUs and $n = 160$

- Honest parties again run in 3.2 sec.
- Adversary, after 13 weeks, succeeds with prob. 2^{-80}

Parametrized encryption schemes

(Revised) syntax of encryption schemes:

Definition 3.7. A **private-key encryption scheme** consists of three probabilistic polynomial-time (p.p.t.) algorithms:

$k \leftarrow \mathsf{Gen}(1^n)$**:** Gen takes security parameter in *unary*, returns key k with $|k| \geq n$

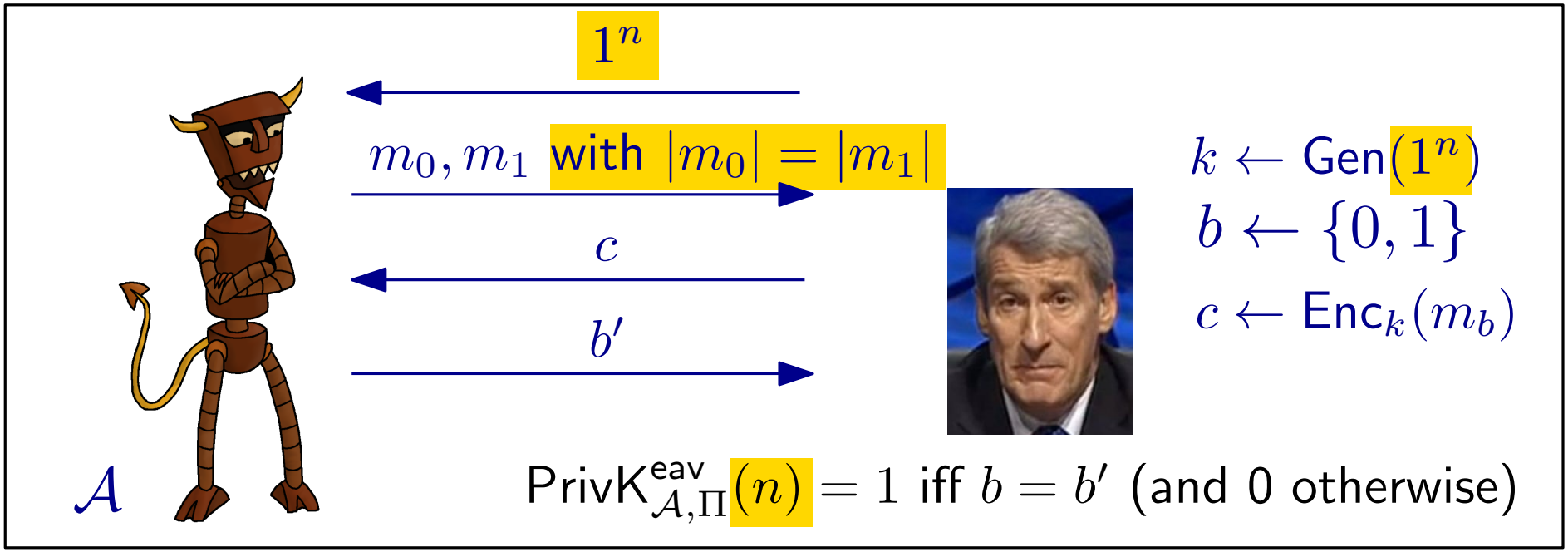
$c \leftarrow \mathsf{Enc}_k(m)$**:** Enc takes key and message $m \in \{0,1\}^*$, returns ciphertext

$m := \mathsf{Dec}_k(c)$**:** Dec is w.l.o.g. deterministic

Computational indistinguishability

The adversarial *indistinguishability* experiment

$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$ where $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$



Definition 3.8. Π *is (computationally)* **indistinguishable** *in the presence of an eavesdropper if for every p.p.t.* $\mathcal{A}$ *there exists negligible* $\varepsilon(\cdot)$*:*

function in n

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \varepsilon(n)$$

Plaintext length

- We want schemes that encrypt arbitrary-length messages
- Even one-time pad reveals message length
- Encryption, in general, does *not* hide the plaintext length
  (not possible)

  $\rightarrow$ reflected in the definition

Leaking the plaintext length **can be a problem!**

- yes/no
- database search ...

  $\Rightarrow$ take care of by other means

Semantic security

- What guarantees does *indistinguishability* yield?
- Can formalize notion that "ciphertext leaks nothing to poly.-time adversary except with negl. probability"

→ **"Semantic security"** (see Def. 3.12 in Katz-Lindell)

Theorem 3.13. A private-key encryption scheme is semantically secure (in the presence of an eavesdropper) if and only if it satisfies computational indistinguishability (in the presence of an eavesdropper)

Introduction to Cryptography

(Lecture 5: Pseudorandomness, security proofs)

Georg Fuchsbauer

TUW 2025W

Pseudorandomness

§3.3.1

What is random?

Which bitstring is *uniformly* random?

- 0000000000000000
- 0110110010110110

If sampled *uniformly* from $\{0,1\}^{16}$ then both sampled with prob. 2^{-16}

Randomness is a property of *distributions*:

A **distribution** (on n-bit strings) is a function

$$D\colon \{0,1\}^n \to [0,1] \quad \text{with} \quad \sum_{x\in\{0,1\}^n} D(x) = 1$$

The **uniform distribution**:

$$U_n\colon \{0,1\}^n \to [0,1]$$
$$x \mapsto 2^{-n}$$

Pseudorandomness

- *"cannot be distinguished from uniform"*

Notation. $x \leftarrow D$ means *"sample x according to D"*

Historically, a distribution D was considered *pseudorandom* if it passed statistical tests: $T_i(\cdot)$

$$\Pr_{x \leftarrow D}\left[T_i(x) = 1\right] \approx \Pr_{x \leftarrow U_n}\left[T_i(x) = 1\right] \quad \text{for } i = 1, \ldots$$

Cryptography: don't know which test adversary uses

$\Rightarrow$ must pass *all efficient* tests

Pseudorandomness

Pseudorandomness (concrete)

- Let $D\colon \{0,1\}^n \to [0,1]$ be a distribution
- D is (t,ε)-**pseudorandom** if for all $\mathcal{A}$ running in time $\leq t$:

$$\left| \Pr_{x \leftarrow D}\left[\mathcal{A}(x)=1\right] - \Pr_{x \leftarrow U_n}\left[\mathcal{A}(x)=1\right] \right| \leq \varepsilon$$

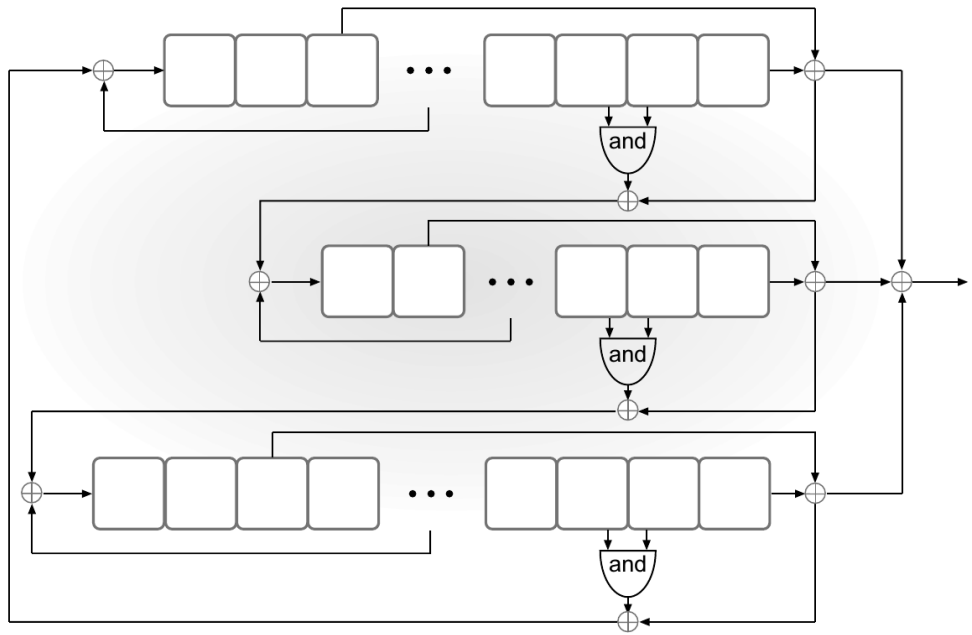
Pseudorandomness (asymptotic)

$\leftarrow$ security parameter n

- Consider *sequence* of distributions $\{D_n\}_{n\in\mathbb{N}} = \{D_1, D_2, \dots\}$ with $D_n\colon \{0,1\}^{\ell(n)} \to [0,1]$ for polynomial ℓ
- $\{D_n\}_{n\in\mathbb{N}}$ is **pseudorandom** if for all p.p.t. $\mathcal{A}$ there exists negligible $\varepsilon(\cdot)$ s.t.

$$\left| \Pr_{x \leftarrow D_n}\left[\mathcal{A}(x)=1\right] - \Pr_{x \leftarrow U_{\ell(n)}}\left[\mathcal{A}(x)=1\right] \right| \leq \varepsilon(n)$$

PRGs



Trivium

$\{0,1\}^{80} \to \{0,1\}^{2^{64}}$

A **pseudorandom** (number) **generator** is a (deterministic) poly.-time algorithm, which

- takes a *uniform seed* and
- expands it into a (longer) *pseudorandom* output

From a "few" true random bits

$\to$ get lots of pseudorandom bits

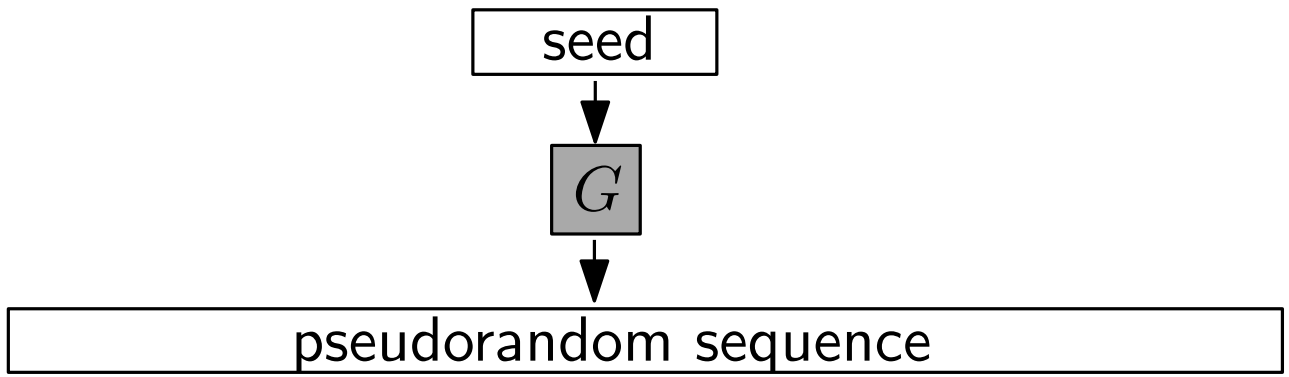
PRGs

Definition 3.14. A (deterministic) poly.-time algorithm G with $|G(x)| = \ell(|x|)$ for all x is a **pseudorandom generator** if it is:

- *expanding:* $\ell(n) > n$ for all n, and
- *pseudorandom:* $\{G(U_n)\}_{n \in \mathbb{N}}$ is

  (a sequence of distributions which is) pseudorandom

uniform distrib. over n-bit strings



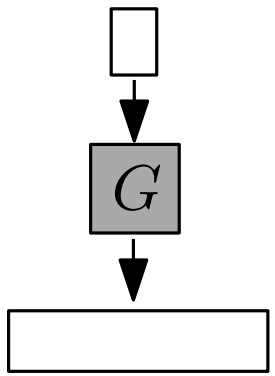
PRGs

Definition 3.14. A (deterministic) poly.-time algorithm G with $|G(x)| = \ell(|x|)$ for all x is a **pseudorandom generator** if it is:

- *expanding:* $\ell(n) > n$ for all n, and
- *pseudorandom:* $\{G(U_n)\}_{n \in \mathbb{N}}$ is

  (a sequence of distributions which is) pseudorandom

G defines *sequence of distributions*: $G(U_1)$,



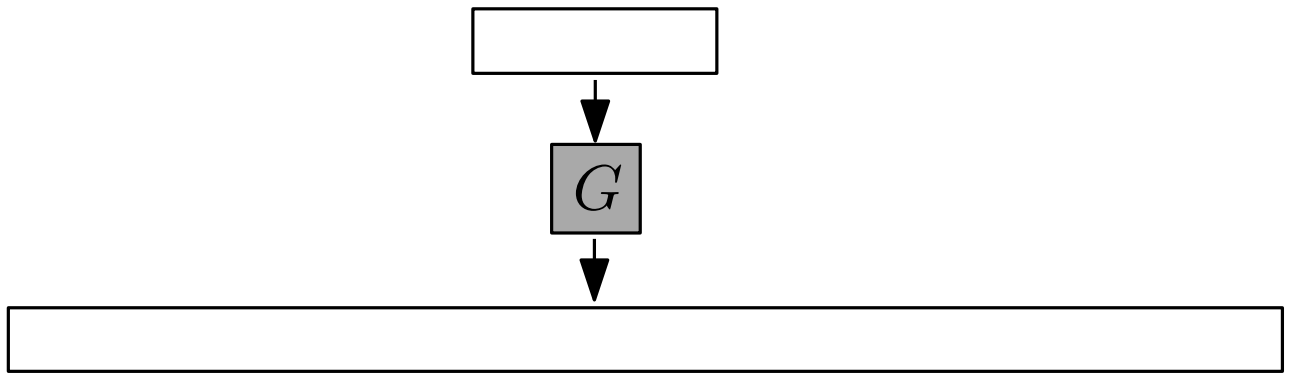
PRGs

Definition 3.14. A (deterministic) poly.-time algorithm G with $|G(x)| = \ell(|x|)$ for all x is a **pseudorandom generator** if it is:

- *expanding:* $\ell(n) > n$ for all n, and
- *pseudorandom:* $\{G(U_n)\}_{n \in \mathbb{N}}$ is

  (a sequence of distributions which is) pseudorandom

G defines *sequence of distributions*: $G(U_1), G(U_2), \ldots$



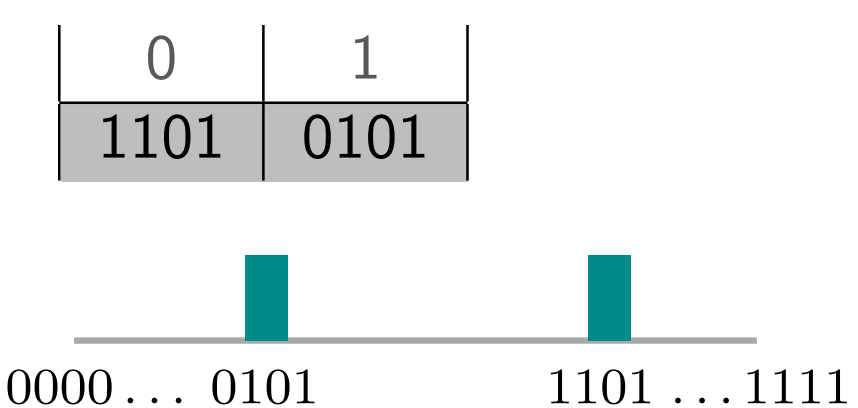
PRGs

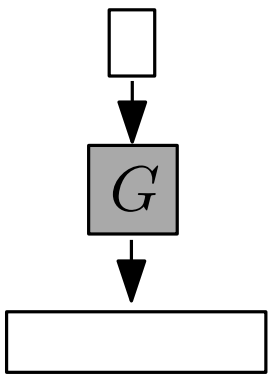
Definition 3.14. A (deterministic) poly.-time algorithm G with $|G(x)| = \ell(|x|)$ for all x is a **pseudorandom generator** if it is:

- *expanding:* $\ell(n) > n$ for all n, and
- *pseudorandom:* $\{G(U_n)\}_{n\in\mathbb{N}}$ is

  (a sequence of distributions which is) pseudorandom

G defines *sequence of distributions*: $G(U_1)$,





PRGs

Definition 3.14. A (deterministic) poly.-time algorithm G with $|G(x)| = \ell(|x|)$ for all x is a **pseudorandom generator** if it is:

- *expanding:* $\ell(n) > n$ for all n, and
- *pseudorandom:* $\{G(U_n)\}_{n\in\mathbb{N}}$ is

  (a sequence of distributions which is) pseudorandom

Recall: $\{D_n\}$ pseudorandom if
for all p.p.t. $\mathcal{A}$ there exists negligible $\varepsilon(\cdot)$ s.t.

$$\left| \Pr_{x\leftarrow D_n}\left[\mathcal{A}(x)=1\right] - \Pr_{x\leftarrow U_{\ell(n)}}\left[\mathcal{A}(x)=1\right]\right| \leq \varepsilon(n)$$

PRGs

Definition 3.14. A (deterministic) poly.-time algorithm G with $|G(x)| = \ell(|x|)$ for all x is a **pseudorandom generator** if it is:

- *expanding:* $\ell(n) > n$ for all n, and
- *pseudorandom:* $\{G(U_n)\}_{n\in\mathbb{N}}$ is

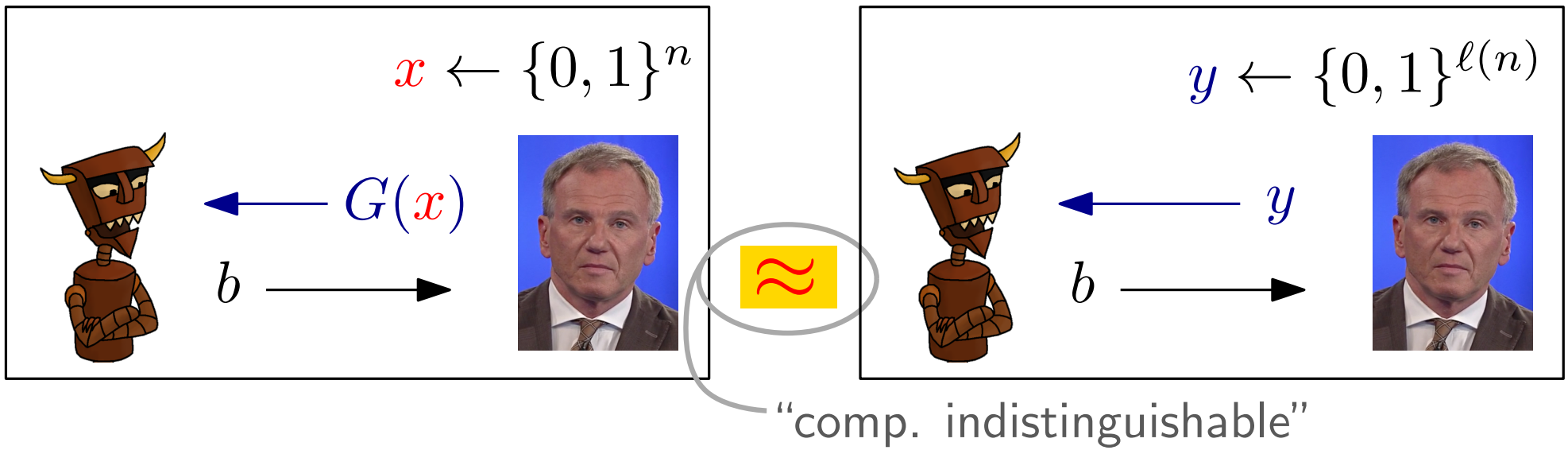
  (a sequence of distributions which is) pseudorandom

  that is,

  for all p.p.t. $\mathcal{A}$ there exists negligible $\varepsilon(\cdot)$ s.t.

$$\left| \Pr_{x\leftarrow U_n}\left[\mathcal{A}(G(x)) = 1\right] - \Pr_{x\leftarrow U_{\ell(n)}}\left[\mathcal{A}(x) = 1\right]\right| \leq \varepsilon(n)$$

PRGs



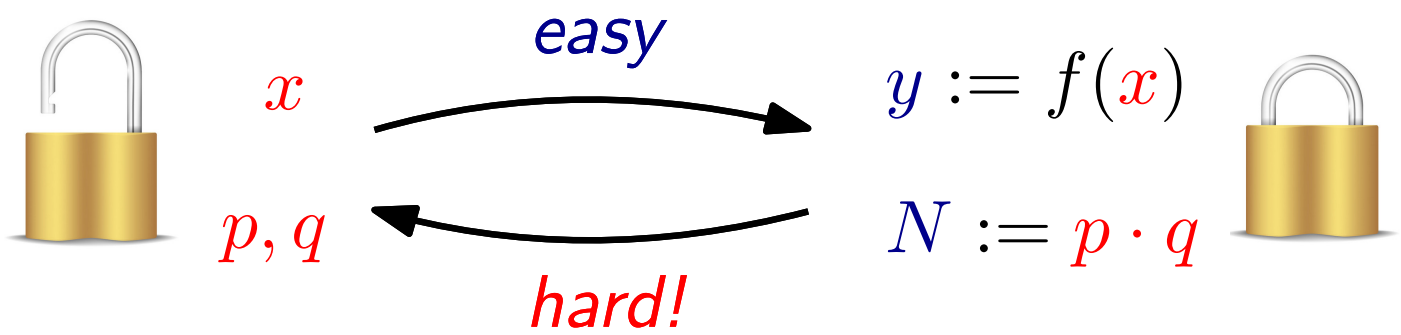
for all p.p.t. $\mathcal{A}$ there exists negligible $\varepsilon(\cdot)$ s.t.

$$\left| \Pr_{x \leftarrow U_n} \left[\mathcal{A}(G(x)) = 1\right] - \Pr_{x \leftarrow U_{\ell(n)}} \left[\mathcal{A}(x) = 1\right] \right| \leq \varepsilon(n)$$

PRGs

- Do PRGs exist? We don't know ($\exists$ PRG $\Rightarrow P \neq NP$)
- PRGs can be constructed from weak assumptions

  PRGs $\Leftarrow$ **one-way functions** (e.g. factoring problem)



This lecture:

- We **assume** certain constructions are PRGs
- Then *use* PRGs to construct encryption schemes

  overcoming shortcomings of perfectly secret schemes

Proofs of security

§3.3.2

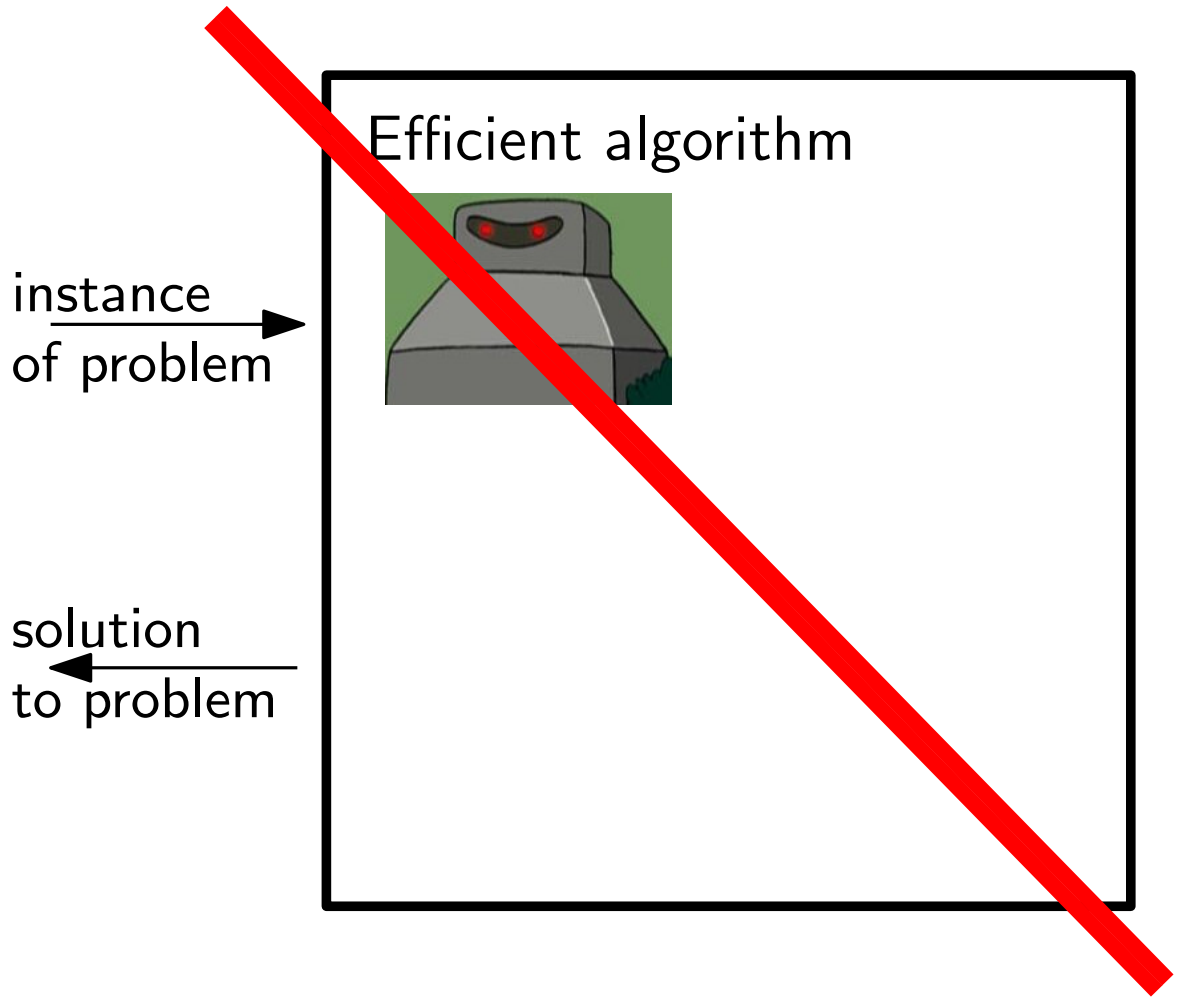
Proofs by reduction

- Want to show that:

  problem hard to solve $\Rightarrow$ *construction hard to break*

Assume:

– problem hard



Proofs by reduction

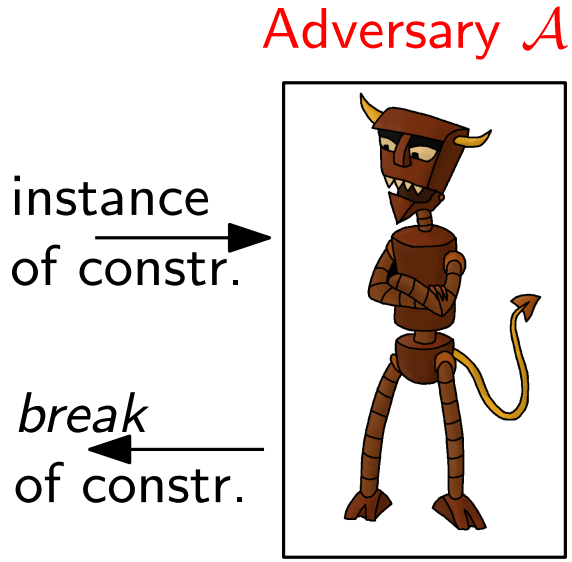
- Want to show that:

  problem hard to solve $\Rightarrow$ *construction hard to break*

Assume:

– problem hard

– exists efficient adversary $\mathcal{A}$ against construction



Proofs by reduction

- Want to show that:

  problem hard to solve $\Rightarrow$ *construction hard to break*

Assume:

– problem hard

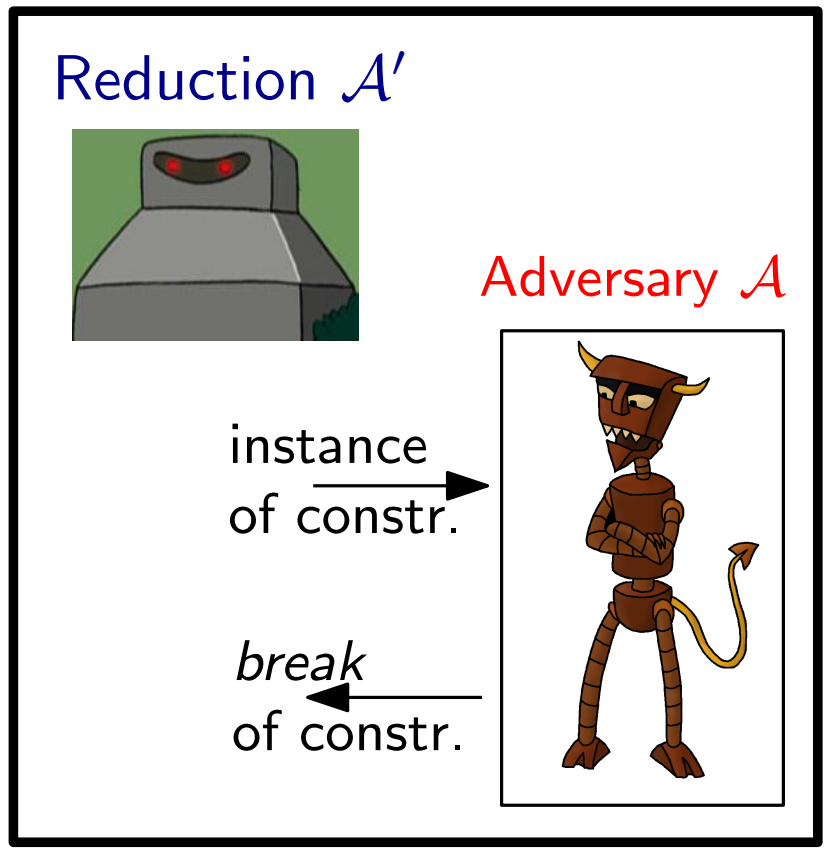
– exists efficient adversary $\mathcal{A}$ against construction

Construct:

– algorithm $\mathcal{A}'$ against problem using $\mathcal{A}$

$\Rightarrow$ No such $\mathcal{A}$ exists

Q.E.D.

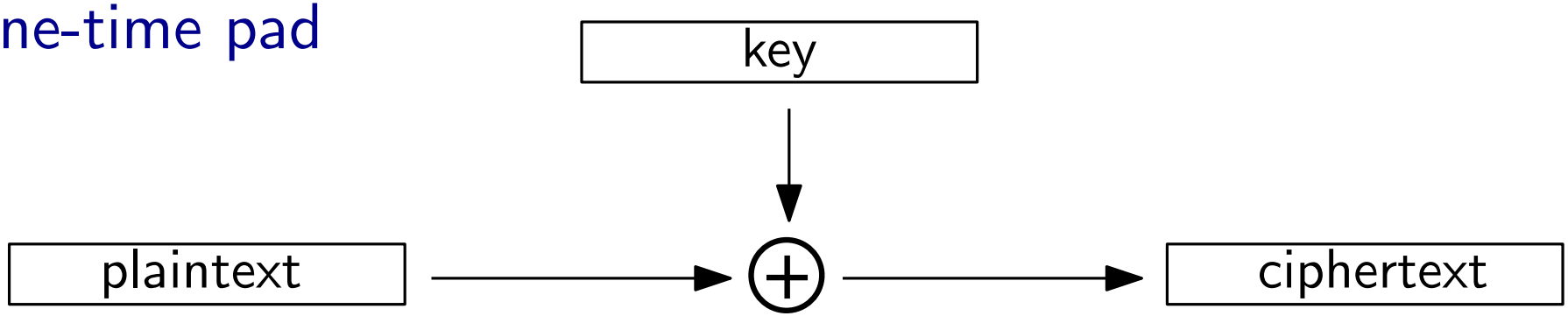


The pseudo one-time pad

§3.3.3

Encryption with short keys

one-time pad



Encryption with short keys

Construction 3.17. Let G be a PRG with $|G(k)| = \ell(|k|)$

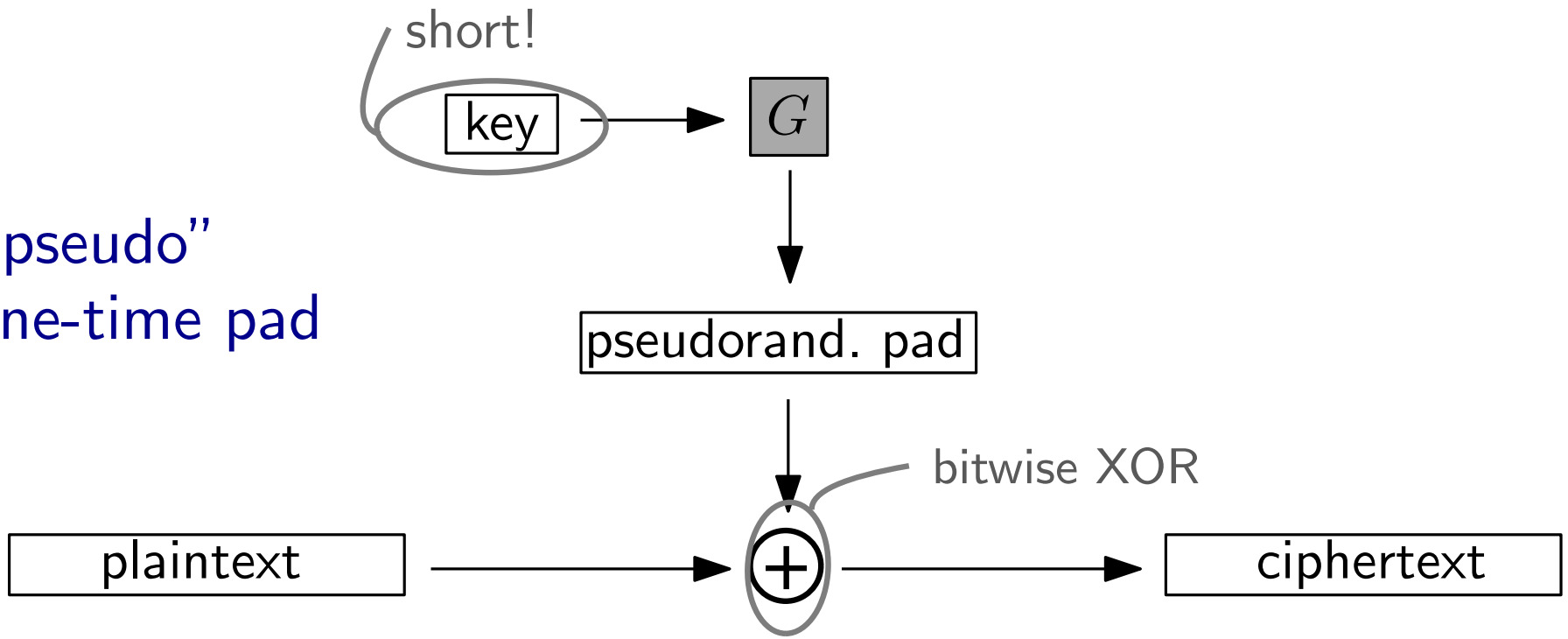
$\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$; return k

$\mathsf{Enc}_k(m)$: return $c := G(k) \oplus m$

$\mathsf{Dec}_k(c)$: return $m := G(k) \oplus c$

sec.par $= n$

$\Rightarrow \mathcal{M} = \{0,1\}^{\ell(n)}$



"pseudo"
one-time pad

Proof of the pseudo one-time pad

- Want to show that:

  If G *is a pseudorandom generator*

  $\Rightarrow$ *the pseudo one-time pad* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is computationally indistinguishable*

Definition 3.8. Π *is computationally* **indistinguishable** *in the presence of an eavesdropper if for every p.p.t.* $\mathcal{A}$ *there exists negligible* $\varepsilon(\cdot)$*:*

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \varepsilon(n)$$

Proof of the pseudo one-time pad

$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$

$k \leftarrow \mathsf{Gen}(1^n)$

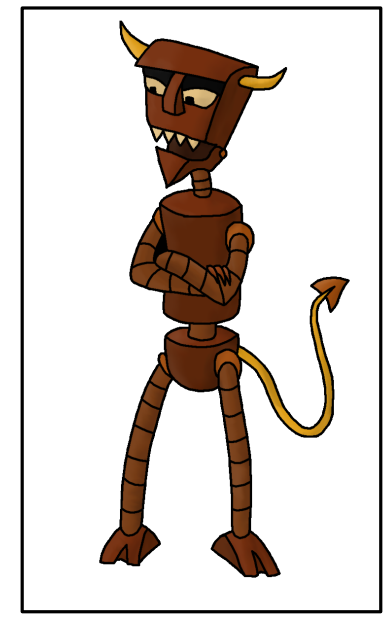
$b \leftarrow \{0,1\}$

$c \leftarrow \mathsf{Enc}_k(m_b)$

m_0, m_1 ←

c →

b' ←



Return 1 iff $b = b'$

Proof of the pseudo one-time pad

Consider any p.p.t. $\mathcal{A}$ playing in PrivK for the pseudo OTP Π

$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$

$k \leftarrow \{0,1\}^n$

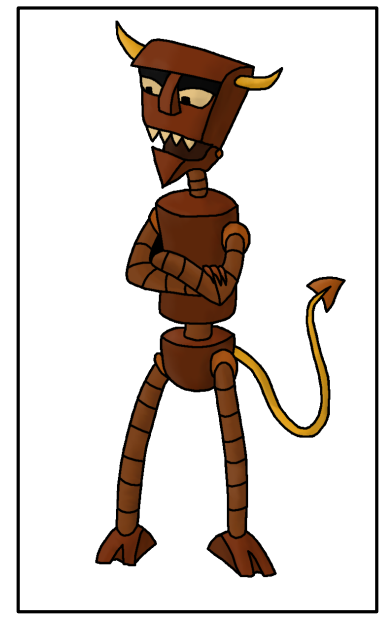
$b \leftarrow \{0,1\}$

$c := G(k) \oplus m_b$

m_0, m_1 ←

c →

b' ←



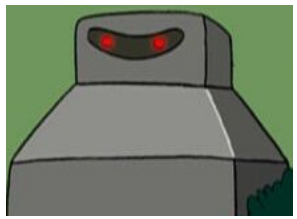
Return 1 iff $b = b'$

Proof of the pseudo one-time pad

Define reduction $\mathcal{A}'$ against pseudorandomness of G

1. $k \leftarrow \{0,1\}^n$;
   $y := G(k)$

Reduction $\mathcal{A}'$



2. $y \leftarrow \{0,1\}^{\ell(n)}$

β

G pseudorandom

$\Rightarrow$ prob. that $\beta = 1$ close in cases 1. and 2., that is:

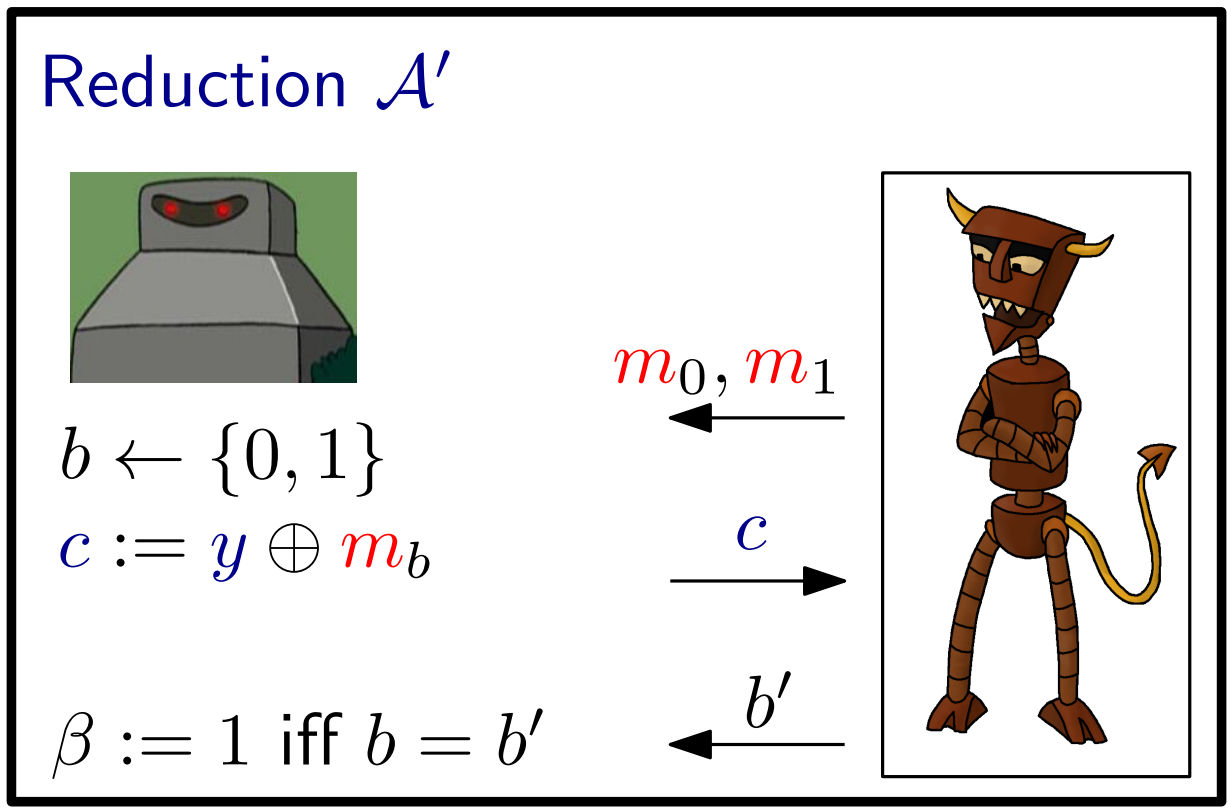
For any p.p.t. $\mathcal{A}'$ there exists negl. $\varepsilon(\cdot)$ s.t.

$$\left| \Pr_{k \leftarrow U_n}\left[\mathcal{A}'(G(k)) = 1\right] - \Pr_{y \leftarrow U_{\ell(n)}}\left[\mathcal{A}'(y) = 1\right] \right| \leq \varepsilon(n)$$

Proof of the pseudo one-time pad

Define reduction $\mathcal{A}'$ against pseudorandomness of G using $\mathcal{A}$

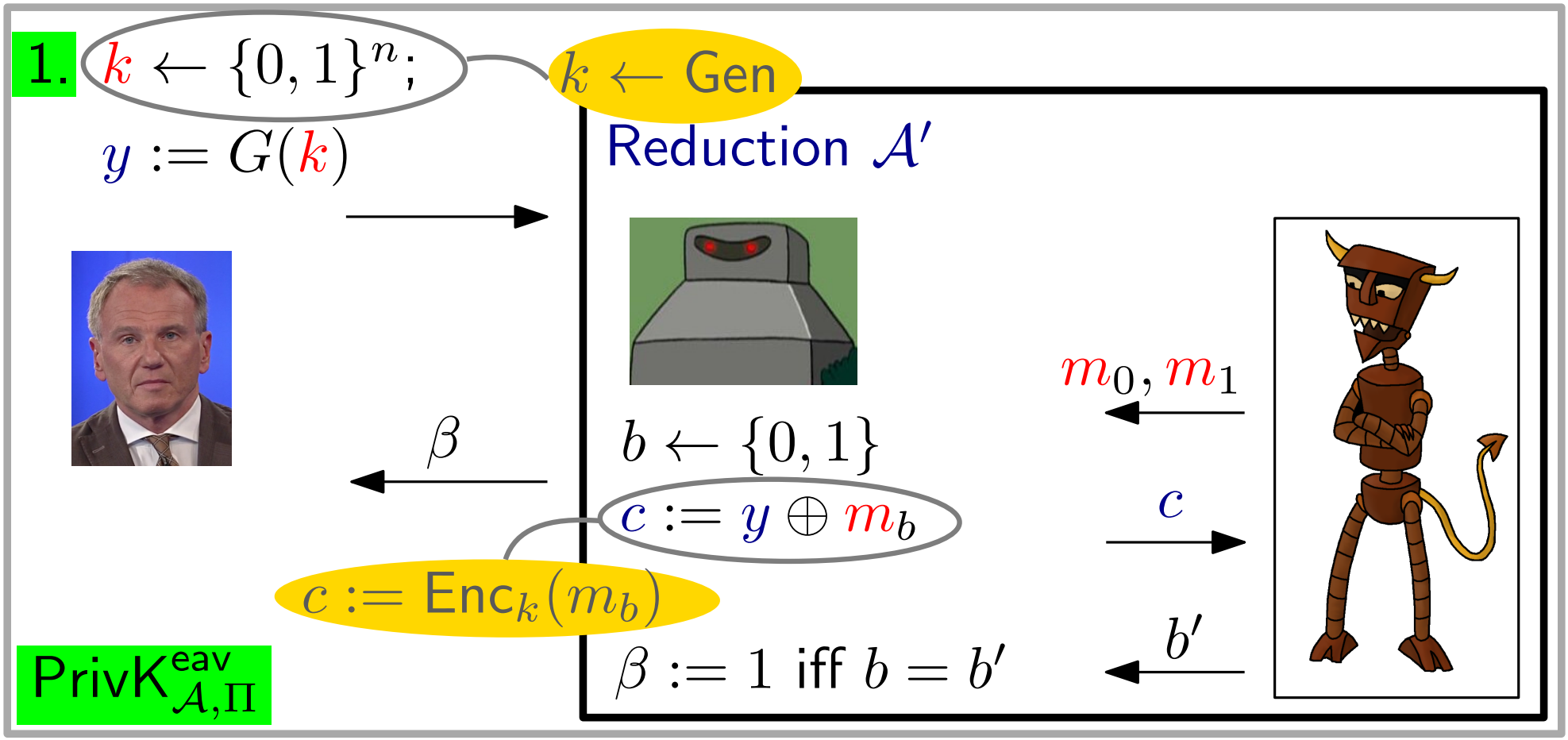
1. $k \leftarrow \{0,1\}^n$;
   $y := G(k)$
2. $y \leftarrow \{0,1\}^{\ell(n)}$



$\mathcal{A}$ p.p.t. $\Rightarrow$ $\mathcal{A}'$ p.p.t.

Proof of the pseudo one-time pad

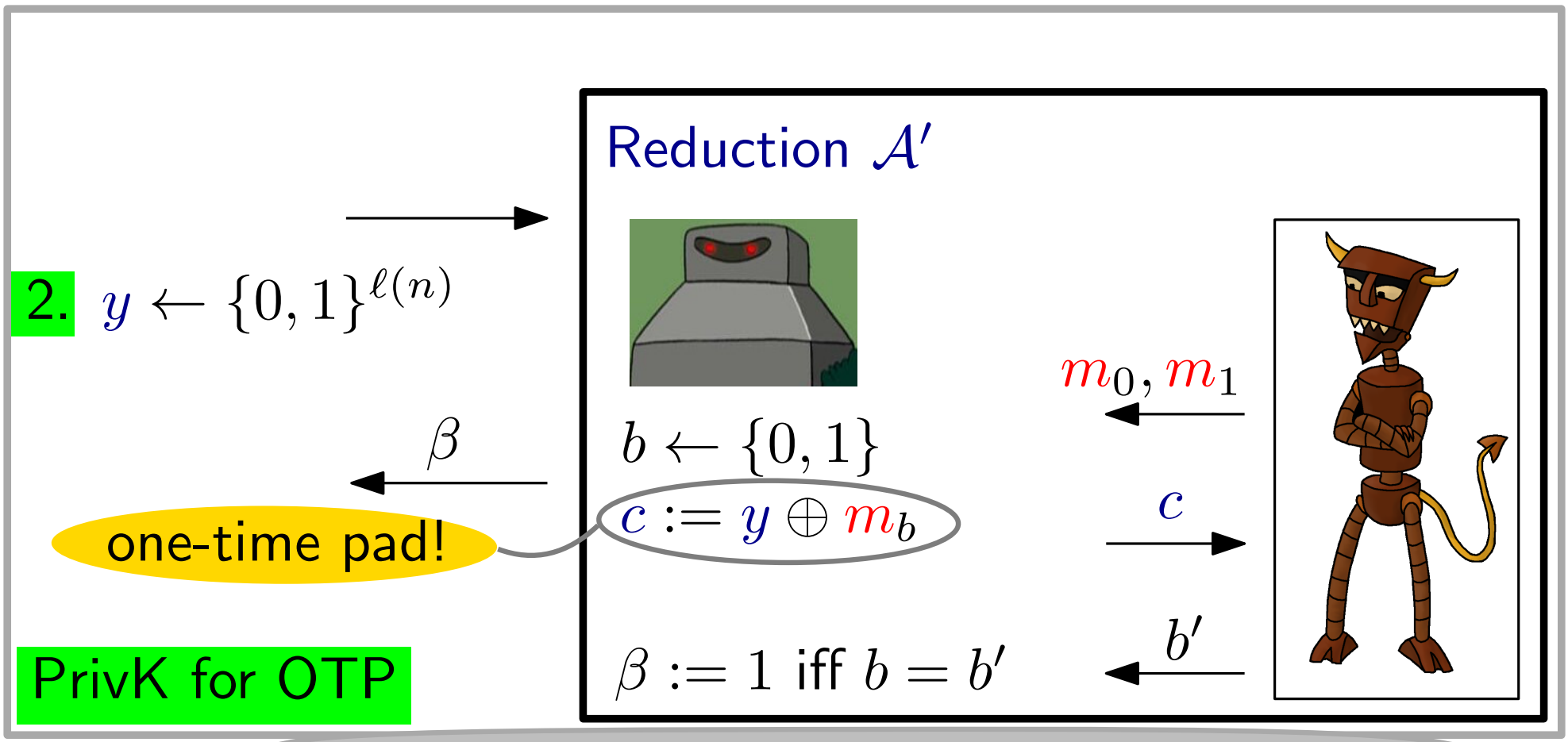
Consider the two cases:



$$\Pr_{k \leftarrow U_n}\left[\mathcal{A}'(G(k)) = 1\right] = \Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right]$$

Proof of the pseudo one-time pad

Consider the two cases:



$$\Pr_{k \leftarrow U_n}\left[\mathcal{A}'(G(k)) = 1\right] = \Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right]$$

$$\Pr_{y \leftarrow U_{\ell(n)}}\left[\mathcal{A}'(y) = 1\right] = \frac{1}{2}$$

Proof of the pseudo one-time pad

- We haved showed: for any p.p.t. $\mathcal{A}$ exists p.p.t. $\mathcal{A}'$ s.t.:

$$\Pr_{k\leftarrow U_n}\left[\mathcal{A}'(G(k))=1\right]=\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1\right]$$
$$\Pr_{y\leftarrow U_{\ell(n)}}\left[\mathcal{A}'(y)=1\right]=\tfrac{1}{2}$$

- G PRG $\Rightarrow$ for any (and thus for) $\mathcal{A}'$ there exists negl. $\varepsilon(\cdot)$:

$$\left|\Pr_{k\leftarrow U_n}\left[\mathcal{A}'(G(k))=1\right]-\Pr_{y\leftarrow U_{\ell(n)}}\left[\mathcal{A}'(y)=1\right]\right|\leq\varepsilon(n)$$

- Together:

$$\left|\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1\right]-\tfrac{1}{2}\right]\leq\varepsilon(n)$$ and thus:

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1\right]\leq\tfrac{1}{2}+\varepsilon(n)$$

$\Rightarrow$ Π satisfies *computational indistinguishability*

Proof of the pseudo one-time pad

Theorem 3.16. If G is a PRG, then the "pseudo one-time pad" (Construction 3.17) is computationally indistinguishable in the presence of an eavesdropper.

- Assumption

  what if G is not pseudorandom?

- Definition

  what if definition is not strong enough?

What if key used more than once?

$c = G(k) \oplus m$

$c' = G(k) \oplus m'$

$\Rightarrow \quad c \oplus c' = m \oplus m'$

m : "B" $=$ 100 0010 or "S" $=$ 101 0011
("buy" or "sell")

m': "Y" $=$ 101 0110 or "N" $=$ 100 1110
("yes" or "no")

Introduction to Cryptography

(Lecture 6: PRFs, CPA-secure encryption)

Georg Fuchsbauer

TUW 2025W

Chosen-plaintext attacks

§3.4.2

Private-key encryption

("Symmetric encryption")

m Enc c Dec m

k secure channel k

Gen

Single-message security:

PrivK$^{\text{eav}}$

Private-key encryption

("Symmetric encryption")

$m_1, m_2, \ldots$ → Enc → $c_1, c_2, \ldots$ → Dec → $m_1, m_2, \ldots$

k → Enc

k → Dec

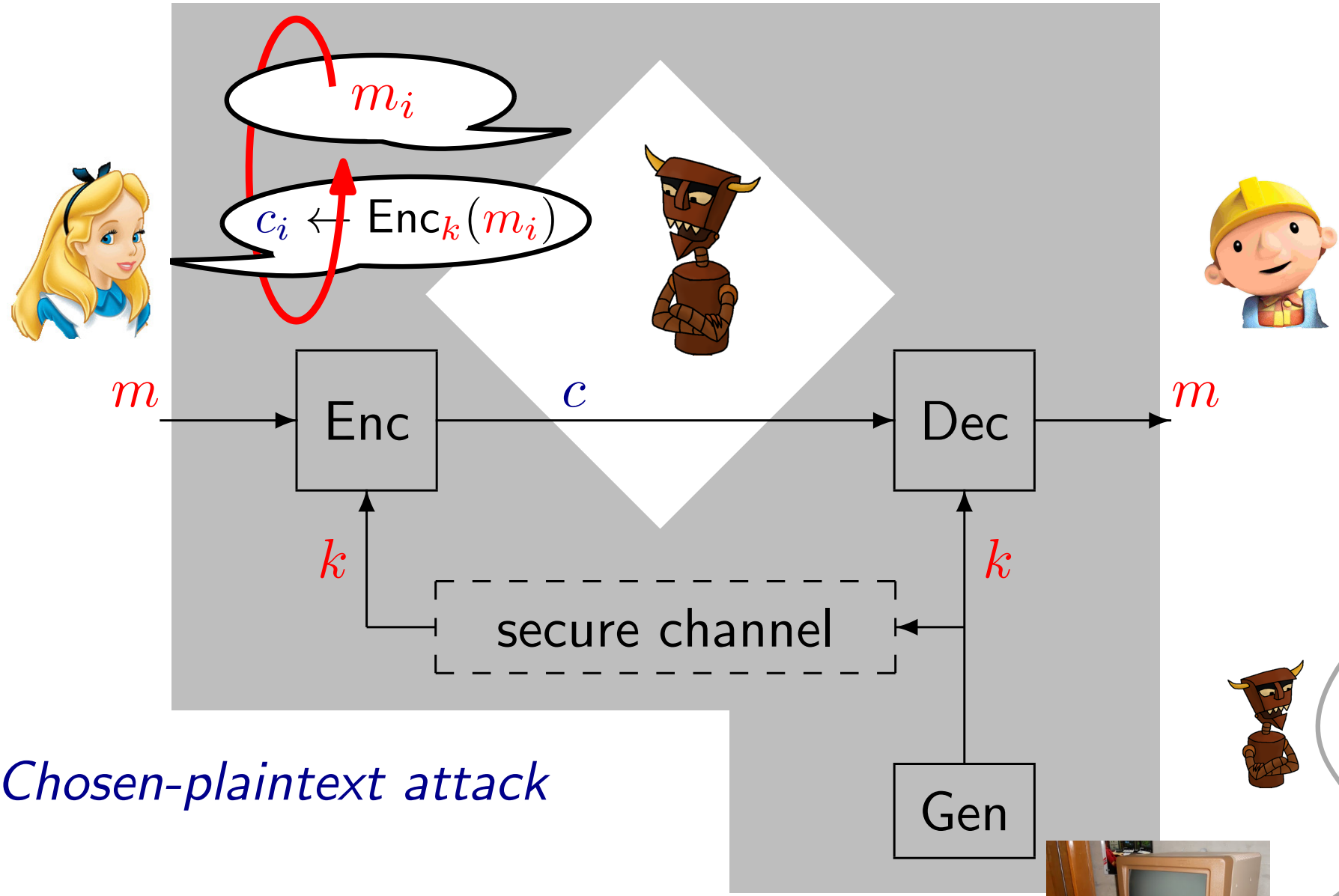
secure channel

Gen

(Multi-message security)

define **stronger** notion instead

CPA-security



Chosen-plaintext attack

unrealistic? no!

Recall. . .

Kerckhoffs' Principle: The adversary knows the scheme

Auguste Kerckhoffs: *La cryptographie militaire* (1883)

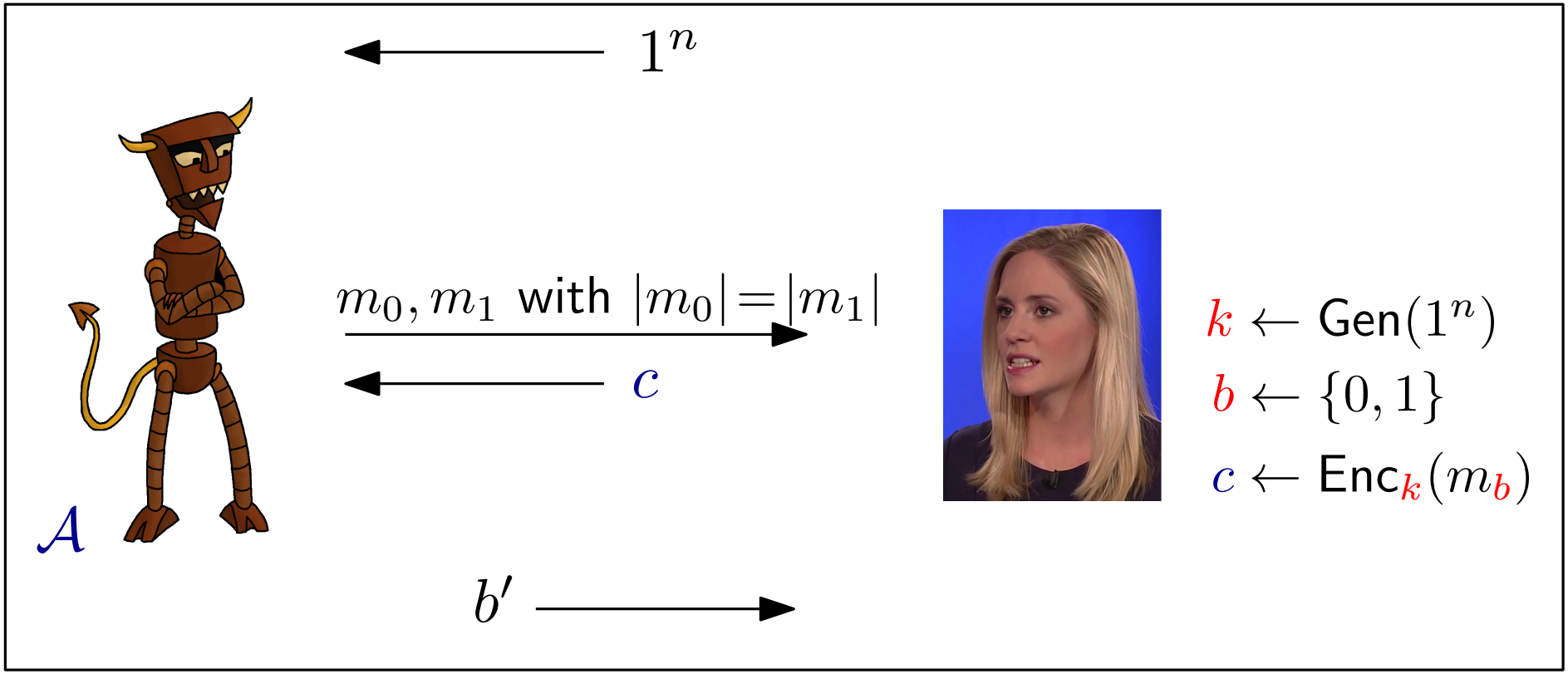
- Adversary's **goals:**
  - Find the key?
  - Recover the plaintext
  - *Guess* a single letter of the plaintext
  - Obtain *any* information about the plaintext

- Adversary's **power:**
  - Sees ciphertexts (one/many)
  - Has seen plaintext/ciphertext pairs
  - Has chosen the plaintexts

    . . . and can ask for *decryption*

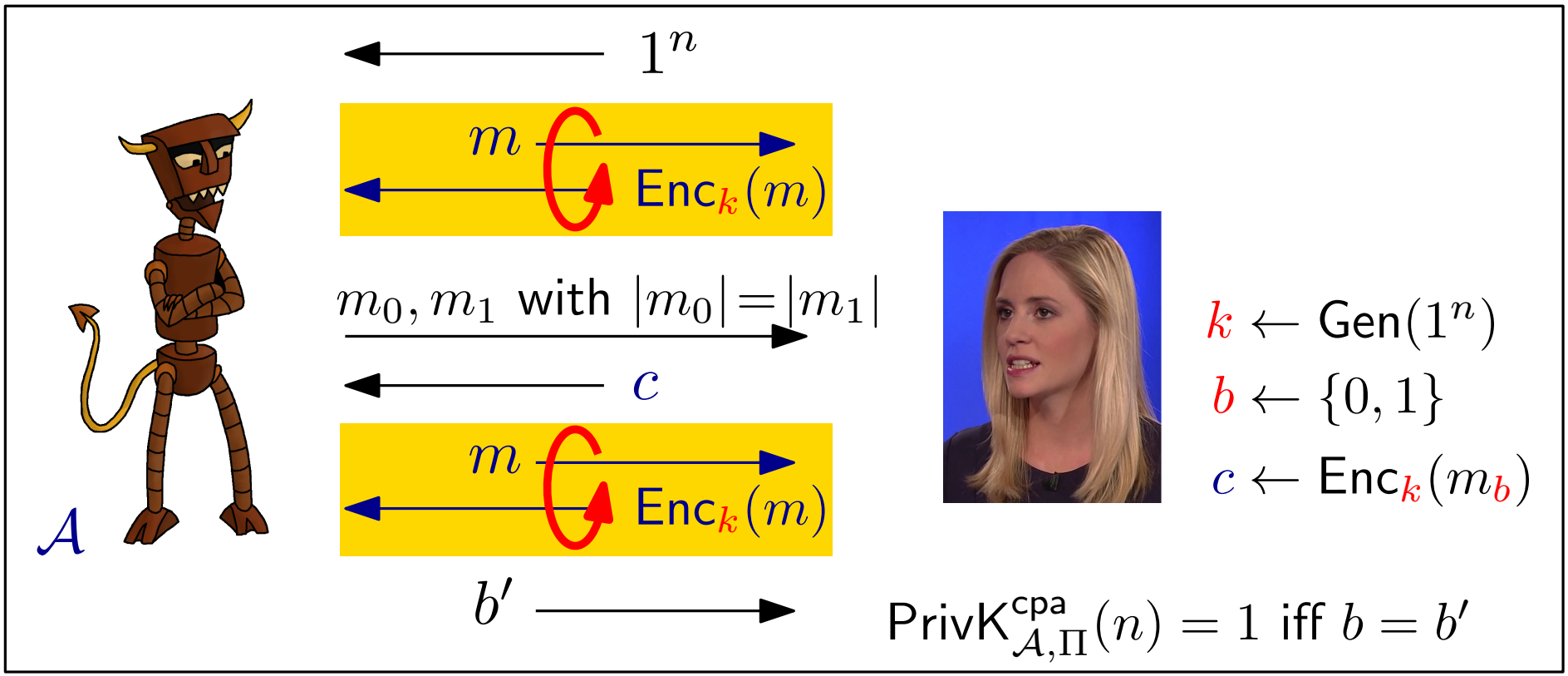
Recall...

$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$ for $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$



CPA-security

$\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$ for $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$



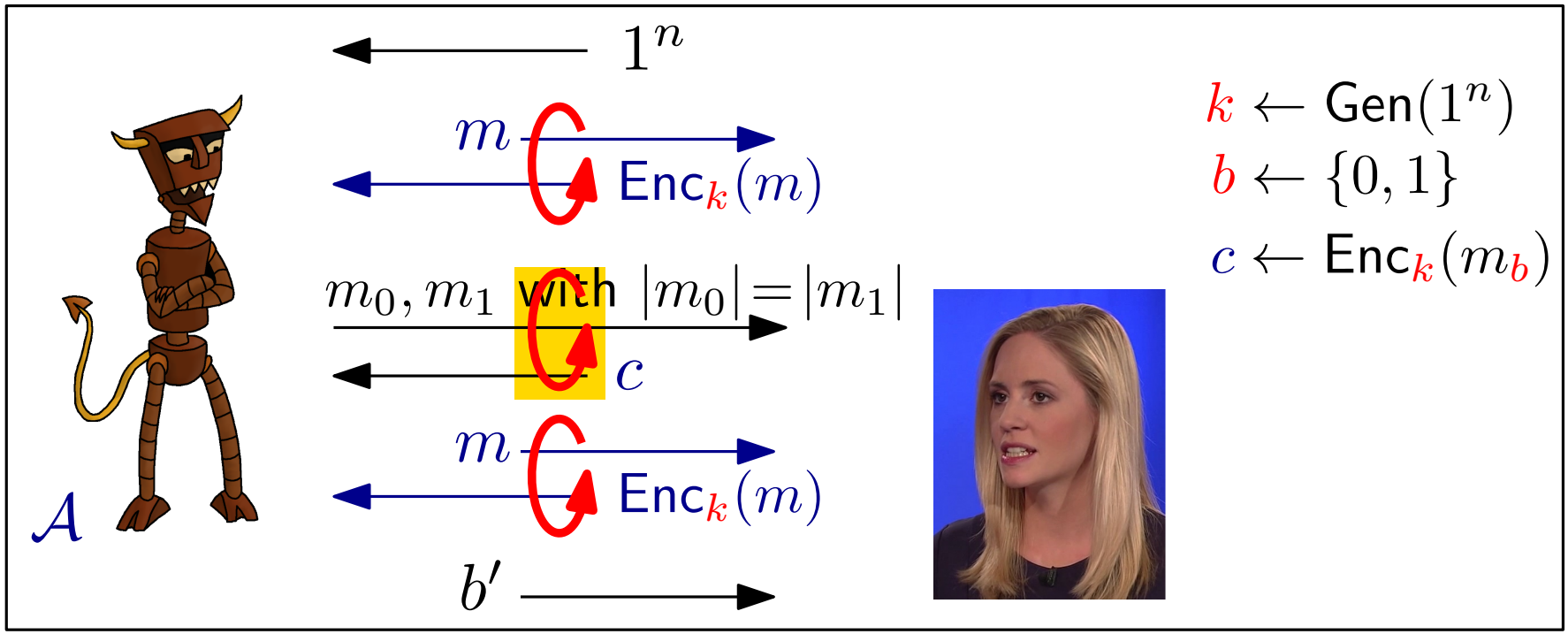
Definition 3.21. Π *is* **secure against chosen-plaintext attacks** *if for every p.p.t.* $\mathcal{A}$ *there exists negligible* $\varepsilon(\cdot)$*:*

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \varepsilon(n)$$

Multi-message security

Could define game where adversary

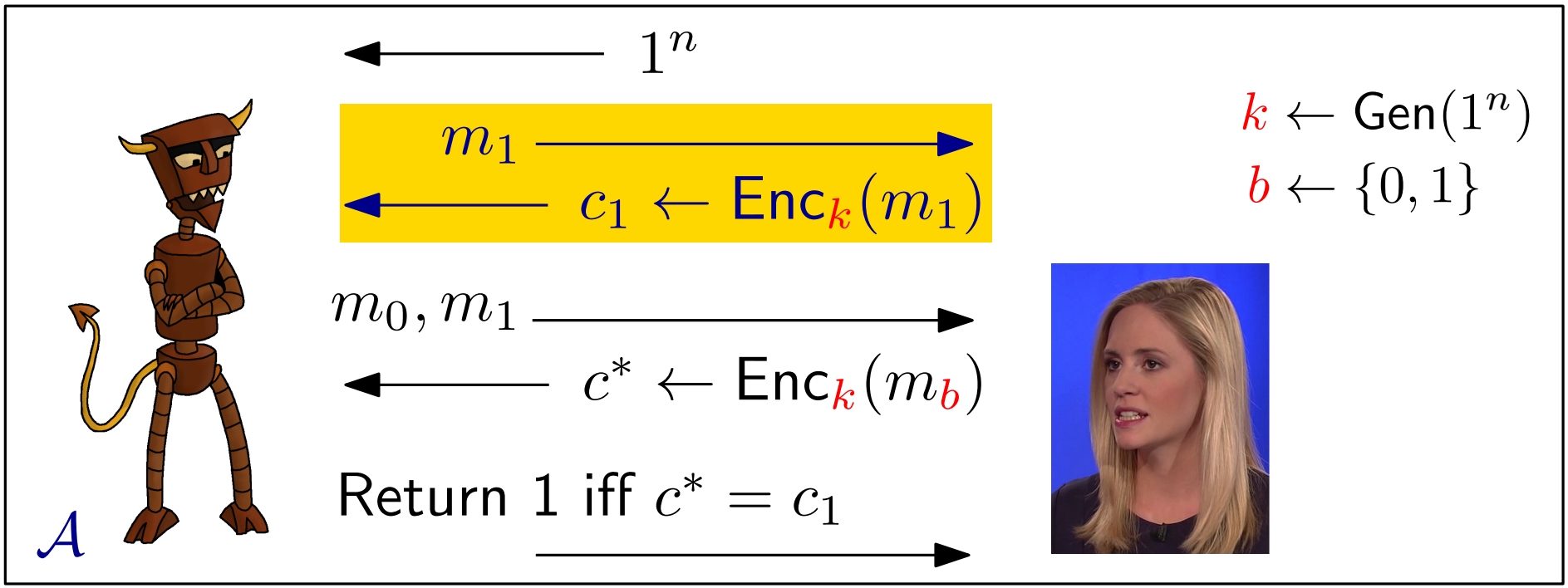
- **gets multiple challenge ciphertexts**



Theorem 3.23. *If a private-key encryption scheme is CPA-secure then it is also* **CPA-secure for multiple encryptions**.

Achievable?

Consider the following adversary:



$\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1\right] = 1$?

Attack works for any **deterministic** *encryption scheme*

Only **randomized** encryption schemes can be CPA-secure!

Pseudorandom functions

§3.5.1

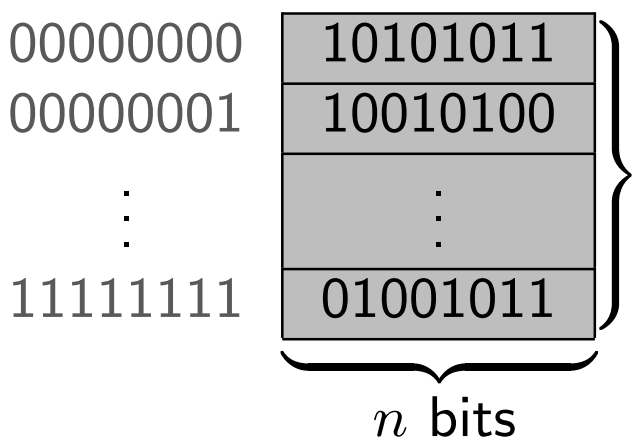
Pseudorandom functions

A **pseudorandom** function *looks like a random function*

Random functions: (recall ideal cipher)

$$\mathsf{Func}_n := \left\{ f : \{0,1\}^n \to \{0,1\}^n \right\}$$

Writing down f:



$\Rightarrow n \cdot 2^n$ bits to write down function

- every $(n\,2^n)$-bit string *uniquely* defines function
- choosing $f \leftarrow \mathsf{Func}_n$ uniformly is same as:
  - for all $x \in \{0,1\}^n$: choose $f(x) \leftarrow \{0,1\}^n$

Pseudorandom functions

Recall: Pseudorandom generator: input random *seed*

- returns string indistinguishable from **random string**

Definitions.

Keyed function: $F\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$

- key $k \in \{0,1\}^*$ defines function $F_k : \{0,1\}^* \to \{0,1\}^*$
  $x \mapsto F(k,x)$

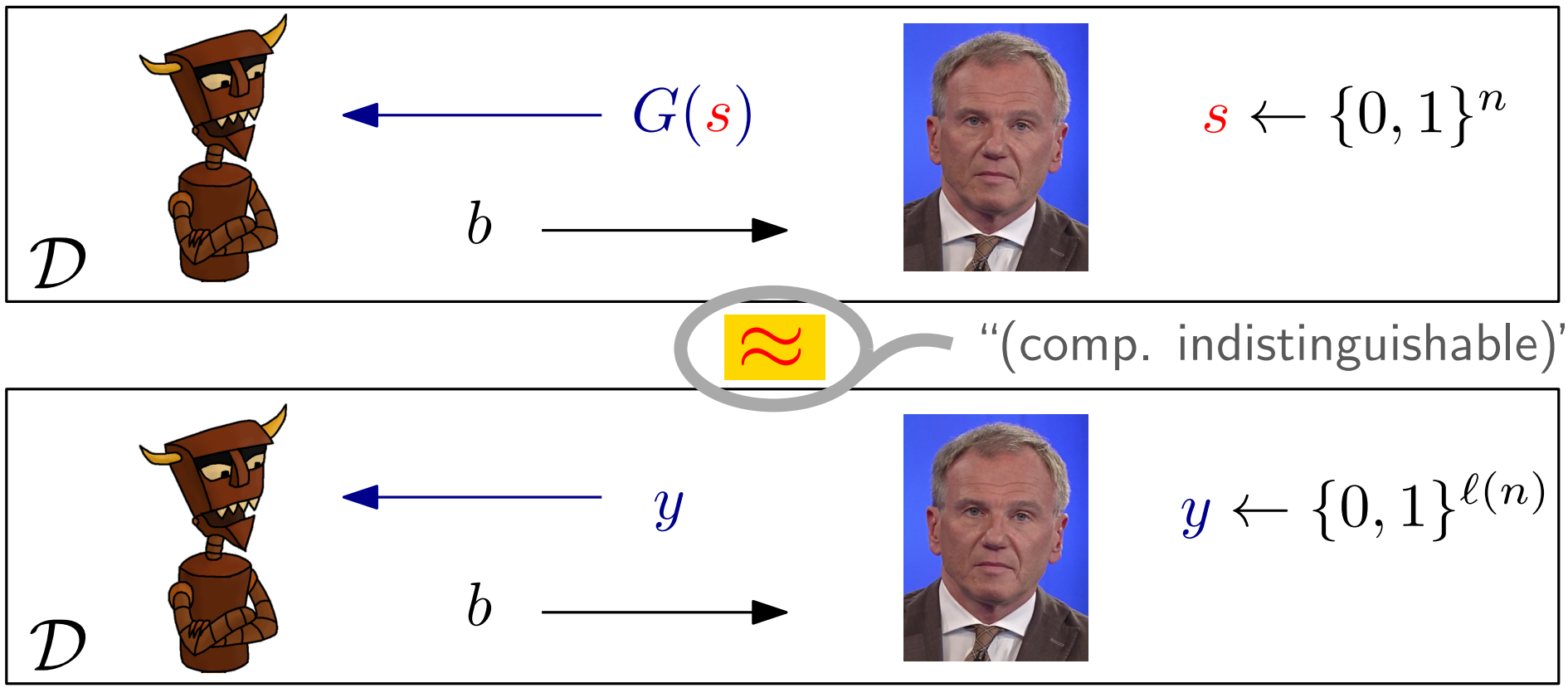
Length-preserving: : $|F(k,x)| = |k| = |x|$

- key $k \in \{0,1\}^n$ defines function $F_k : \{0,1\}^n \to \{0,1\}^n$

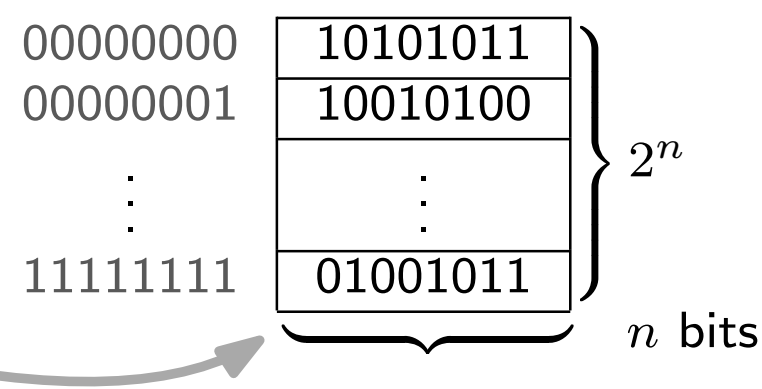
Pseudorandom function (PRF):

- efficient (poly.-time) keyed function
- for uniform k: F_k indistinguishable from **random function**

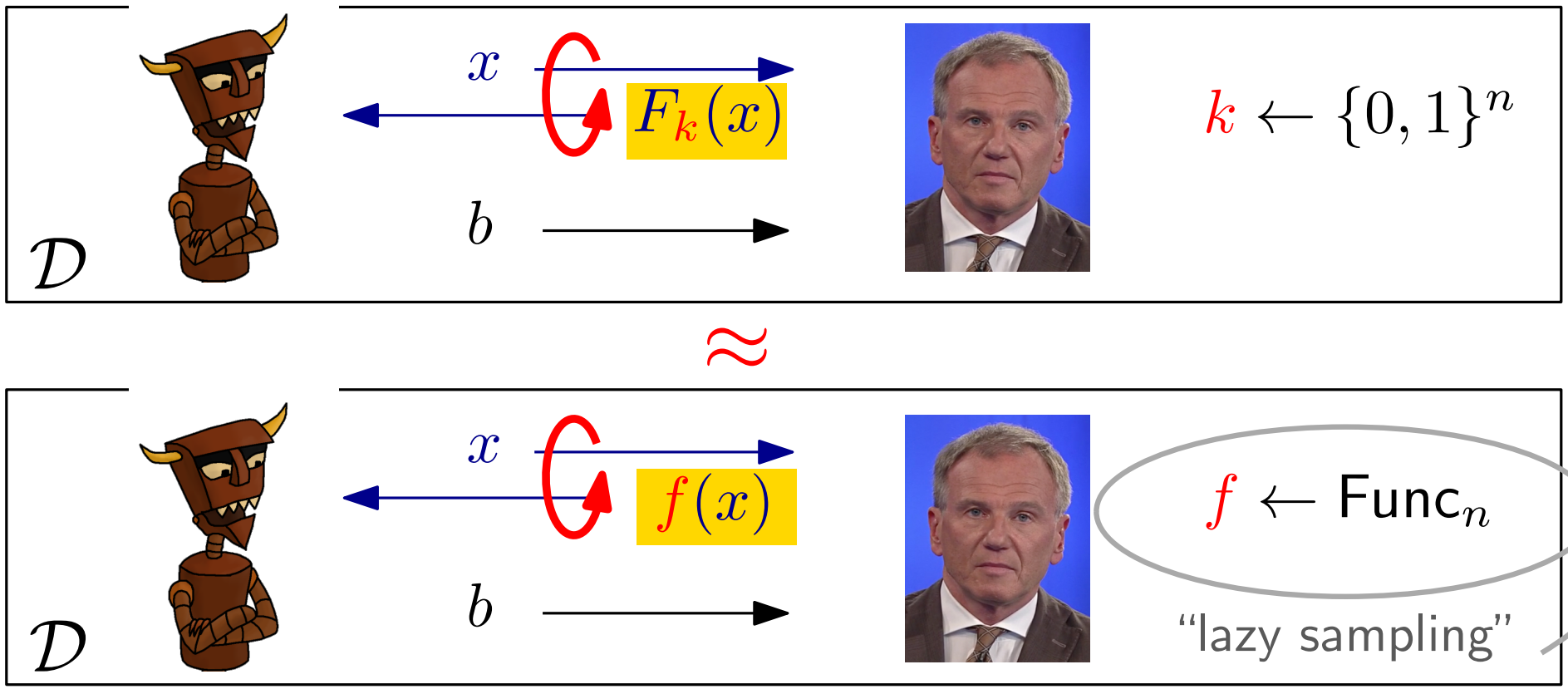
Recall: Pseudorandom generators



PRF: "give function to adversary"?



Pseudorandom functions



Definition 3.24. A keyed function F is a **pseudorandom function** (PRF) if for all p.p.t. $\mathcal{D}$ there exists negl. $\varepsilon(\cdot)$:

$$\left| \Pr_{k \leftarrow \{0,1\}^n} \left[\mathcal{D}^{F_k(\cdot)}(1^n) = 1 \right] - \Pr_{f \leftarrow \mathsf{Func}_n} \left[\mathcal{D}^{f(\cdot)}(1^n) = 1 \right] \right| \leq \varepsilon(n)$$

Pseudorandom permutations

Consider permutations over $\{0,1\}^n$: $\mathsf{Perm}_n \subset \mathsf{Func}_n$

Keyed permutation:

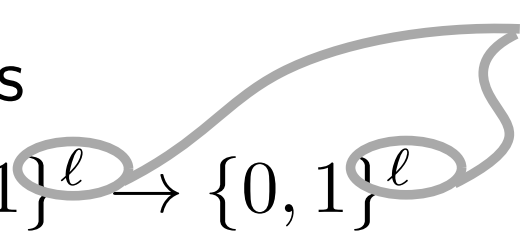
- for every k: $F_k : \{0,1\}^n \to \{0,1\}^n$ is bijective
- F_k^{-1} is *efficiently* computable (in addition to F_k)

Pseudorandom permutation (PRP):

- F_k, for uniform $k \leftarrow \{0,1\}^n$, is indistinguishable from a uniform $f \leftarrow \mathsf{Perm}_n$

Block ciphers:

- Practical constructions of PRPs
- fixed length: $F : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^\ell$



Best distinguishing attack should take time $\approx 2^n$

CPA-secure encryption schemes

§3.5.2

CPA-secure encryption

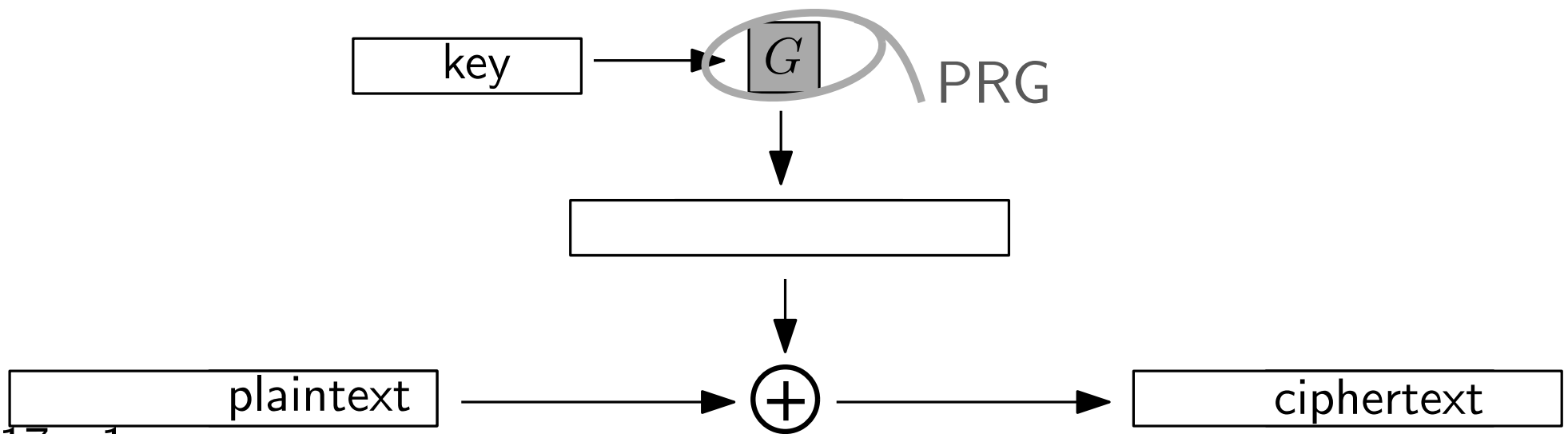
- Encryption using pseudo-random permutation (blockcipher)
- First idea: given PRP $F\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$,
  define:
  - $\mathrm{Enc}_k(m) := F_k(m)$
  - $\mathrm{Dec}_k(c) := F_k^{-1}(c)$
- Secure?
  - passively secure (indist. in presence of eavesdropper)
  - CPA-secure?

    no, since *deterministic*!

$\Longrightarrow$ construct *probabilistic* encryption scheme

CPA-secure encryption from PRFs

Recall: "pseudo" one-time pad



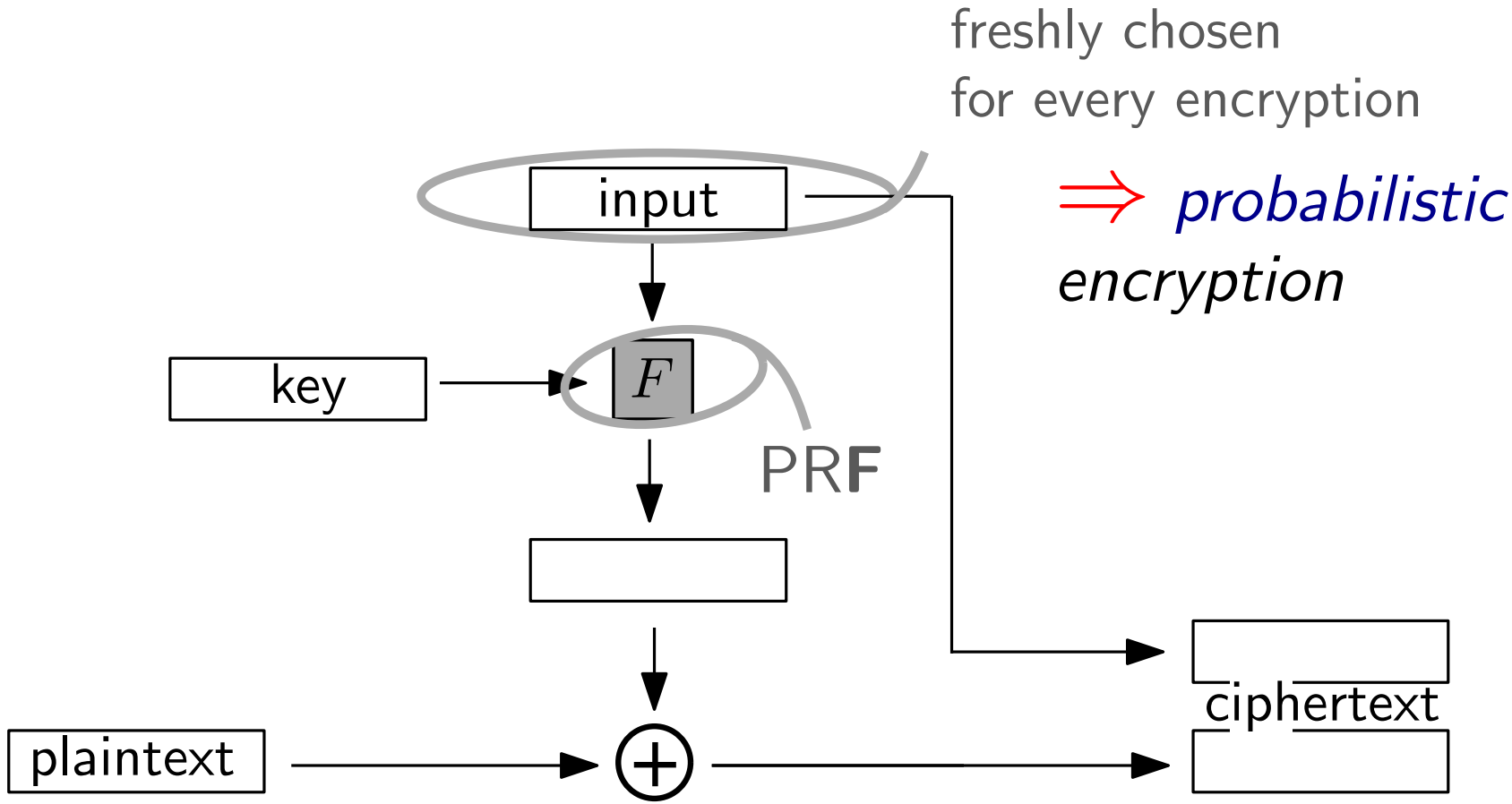
CPA-secure encryption from PRFs

Construction 3.28. Let $F\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$

$\mathsf{Gen}(1^n)$: sample $k \leftarrow \{0,1\}^n$; return k

$\mathsf{Enc}_k(m)$: sample $r \leftarrow \{0,1\}^n$; return $c := \big(r,\ F_k(r) \oplus m\big)$

$\mathsf{Dec}_k((r,s))$: return $m := F_k(r) \oplus s$



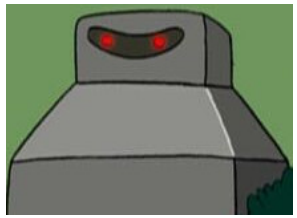
Proof that Construction 3.28 is CPA-secure

Proof of CPA security

Reduction $\mathcal{D}$ against security of PRF F

$k \leftarrow \{0,1\}^n$

Reduction $\mathcal{D}$



r

$y = \begin{matrix} F_k(r) \\ \hline f(r) \end{matrix}$

β

F PRF $\Rightarrow$

For any p.p.t. $\mathcal{D}$ there exists negl. $\varepsilon(\cdot)$ s.t.

$$\left|\Pr_{k\leftarrow\{0,1\}^n}\left[\mathcal{D}^{F_k(\cdot)}=1\right]-\Pr_{f\leftarrow\mathsf{Func}_n}\left[\mathcal{D}^{f(\cdot)}=1\right]\right|\leq\varepsilon(n)$$

Proof of CPA security

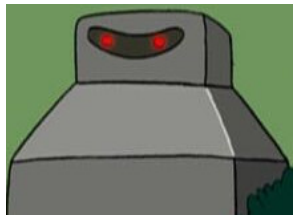
Reduction $\mathcal{D}$ against security of PRF F using $\mathcal{A}$

$k \leftarrow \{0,1\}^n$

r ←

$y = F_k(r)$ → $f(r)$

Reduction $\mathcal{D}$



$r \leftarrow \{0,1\}^n$

m ←

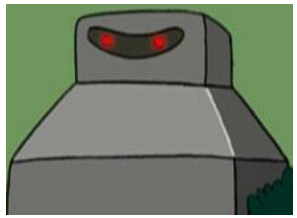
$c = (r, y \oplus m)$ →

Proof of CPA security

Reduction $\mathcal{D}$ against security of PRF F using $\mathcal{A}$

$k \leftarrow \{0,1\}^n$

Reduction $\mathcal{D}$



Challenge query

r ←

$y = F_k(r)$ →

$r \leftarrow \{0,1\}^n$

$b \leftarrow \{0,1\}$

← m_0, m_1

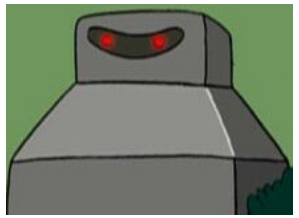
$c = (r, y \oplus m_b)$ →

Proof of CPA security

Reduction $\mathcal{D}$ against security of PRF F using $\mathcal{A}$

$k \leftarrow \{0,1\}^n$

Reduction $\mathcal{D}$



Encrypt. query

r ←

$y = F_k(r)$ →

$r \leftarrow \{0,1\}^n$

← m

$c = (r, y \oplus m)$ →

← β

$\beta := 1$ iff $b = b'$

← b'

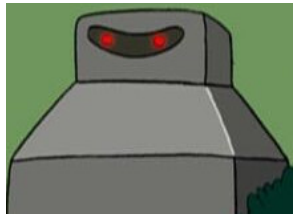
$\mathcal{A}$ p.p.t. $\Rightarrow$ $\mathcal{D}$ p.p.t.

Proof of CPA security

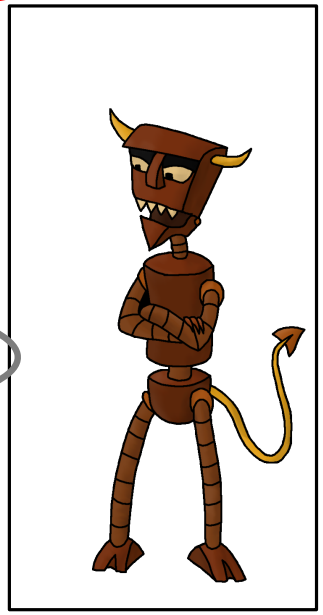
Consider two cases:

$k \leftarrow \{0,1\}^n$

Reduction $\mathcal{D}$



Challenge query



r

m_0, m_1

$y = F_k(r)$

$r \leftarrow \{0,1\}^n$

$b \leftarrow \{0,1\}$

$c = (r, y \oplus m_b)$

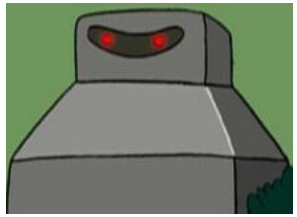
β

$c \leftarrow \mathsf{Enc}_k(m_b)$

Proof of CPA security

Consider two cases:

$k \leftarrow \{0,1\}^n$



Challenge query

r

$y = F_k(r)$

m_0, m_1

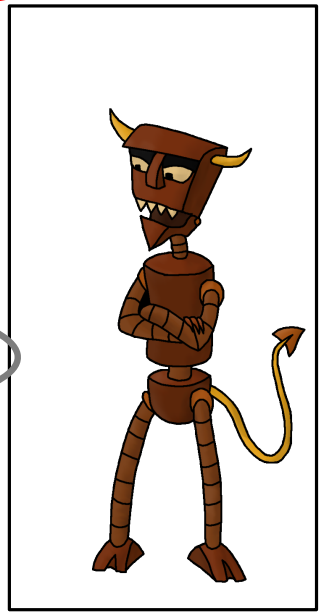
$r \leftarrow \{0,1\}^n$

$b \leftarrow \{0,1\}$

$c = (r, y \oplus m_b)$

$\beta := 1$ iff $b = b'$

$c \leftarrow \mathsf{Enc}_k(m_b)$



$\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}$

$$\Pr_k\left[\mathcal{D}^{F_k(\cdot)} = 1\right] = \Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1\right]$$

Proof of CPA security

Consider two cases:

$k \leftarrow \{0,1\}^n$

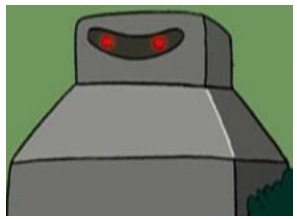
r

$y =$ $f(r)$

unif. random for every r

β

Reduction $\mathcal{D}$



$r \leftarrow \{0,1\}^n$

$b \leftarrow \{0,1\}$

Challenge query

m_0, m_1

$c = (r, y \oplus m_b)$

- probability of winning $= \frac{1}{2}$?? Let $c^* = f(r^*) \oplus m_b$
- what if r^* chosen another time? $\Rightarrow c \oplus c^* = m \oplus m_b$

Let R ("repeat") denote event; let $q(n)$ upper bound on queries

Proof of CPA security

Consider two cases:

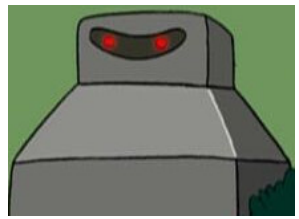
$k \leftarrow \{0,1\}^n$

r

$y =$ $f(r)$

β

Reduction $\mathcal{D}$



$r \leftarrow \{0,1\}^n$

$b \leftarrow \{0,1\}$

Challenge query

m_0, m_1

$c = (r, y \oplus m_b)$

$$\Pr_f\left[\mathcal{D}^{f(\cdot)} = 1\right] = \Pr_f\left[\mathcal{D}^{f(\cdot)} = 1 \wedge R\right] + \Pr_f\left[\mathcal{D}^{f(\cdot)} = 1 \wedge \overline{R}\right]$$

$$\leq \Pr\left[R\right] + \Pr_f\left[\mathcal{D}^{f(\cdot)} = 1 \,\middle|\, \overline{R}\right] \leq \frac{q(n)}{2^n} + \frac{1}{2}$$

$\{0,1\}^n \Rightarrow \leq q(n)$ points

Proof of CPA-security

Putting everything together: For all p.p.t. $\mathcal{D}$ exists negl. ε:

$$\underbrace{\left|\Pr_{k}\left[\mathcal{D}^{F_k(\cdot)}=1\right]\right.}_{=\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)=1\right]}\underbrace{\left.-\Pr_{f}\left[\mathcal{D}^{f(\cdot)}=1\right]\right|}_{\leq \frac{q(n)}{2^n}+\frac{1}{2}} \leq \varepsilon(n)$$

$\Rightarrow$ For all p.p.t. $\mathcal{A}$ exists negl. $\varepsilon'(n) := \varepsilon(n) + q(n)\cdot 2^{-n}$ with

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)=1\right] \leq \frac{1}{2}+\varepsilon'(n)$$

$\Rightarrow$ Π is CPA-secure $\square$

Theorem 3.29. If F is a pseudorandom function, then Construction 3.28 is a CPA-secure private-key encryption scheme for messages of length n.

Proof idea

$\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$ for $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$

$\mathcal{A}$

1^n

$k \leftarrow \{0,1\}^n$

$b \leftarrow \{0,1\}$

m'_i

$(r_i, F_k(r_i) \oplus m'_i)$

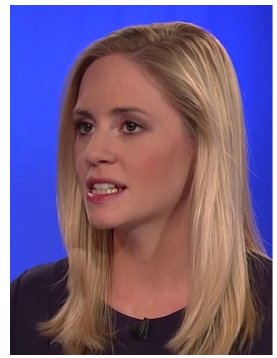
for $r_i \leftarrow \{0,1\}^n$

m_0, m_1

$(r^*, F_k(r^*) \oplus m_b)$

for $r^* \leftarrow \{0,1\}^n$

$\ldots$



b' $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1$ iff $b = b'$

Proof idea

"Ideal game" for $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$

$\mathcal{A}$

1^n

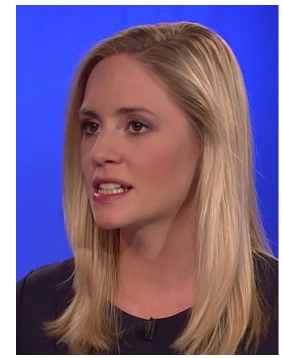
m'_i

$(r_i, f(r_i) \oplus m'_i)$

for $r_i \leftarrow \{0,1\}^n$

$f \leftarrow \mathsf{Func}_n$

$b \leftarrow \{0,1\}$



m_0, m_1

$(r^*, f(r^*)) \oplus m_b)$

for $r^* \leftarrow \{0,1\}^n$

$\ldots$

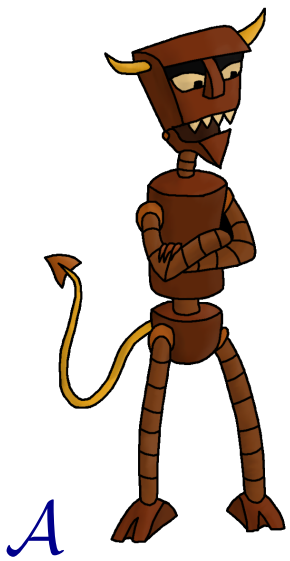
b' $\longrightarrow$ $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1$ iff $b = b'$

. . . "indistinguishable" by PRF security

$\Delta \leq \varepsilon(n)$

Proof idea

"Ideal game" for $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$



$\mathcal{A}$

1^n

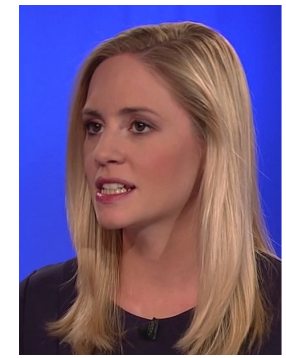
m_i'

$(r_i, f(r_i) \oplus m_i')$

for $r_i \leftarrow \{0,1\}^n$

$f \leftarrow \mathsf{Func}_n$

$b \leftarrow \{0,1\}$



m_0, m_1

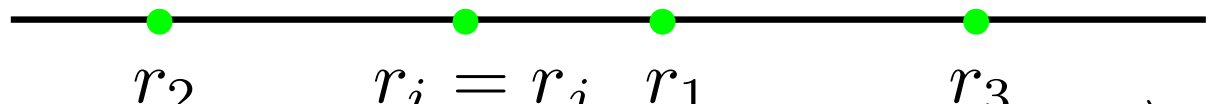
$(r^*, f(r^*)) \oplus m_b)$

for $r^* \leftarrow \{0,1\}^n$

$\ldots$

b' $\longrightarrow$ $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1$ iff $b = b'$

can $\mathcal{A}$ still win? ...not if $f(r^*)$ is never used elsewhere!

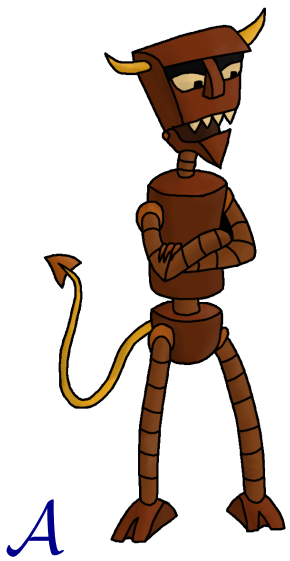


$\Rightarrow \leq q$ points

$\Rightarrow$ probability of collision $\leq q/2^n$

Proof idea

"Ideal game" for $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$



$\mathcal{A}$

1^n

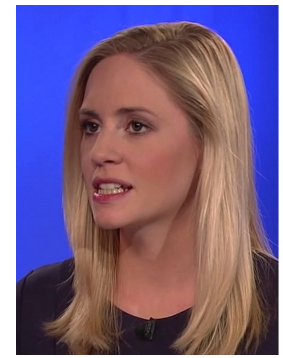
m_i'

$(r_i, f(r_i) \oplus m_i')$

for $r_i \leftarrow \{0,1\}^n$

$f \leftarrow \mathsf{Func}_n$

$b \leftarrow \{0,1\}$



m_0, m_1

$(r^*, f(r^*)) \oplus m_b)$

for $r^* \leftarrow \{0,1\}^n$

$\ldots$

b' $\longrightarrow$ $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1$ iff $b = b'$

Theorem. For all p.p.t. $\mathcal{A}$:

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \underbrace{\varepsilon(n) + q(n)/2^n}_{\text{negligible}}$$

Introduction to Cryptography

(Lecture 7: modes of operation, CCA security)

Elena Andreeva

TUW 2025W

Recall. . .

Kerckhoffs' Principle: The adversary knows the scheme

Auguste Kerckhoffs: *La cryptographie militaire* (1883)

- Adversary's **goals:**

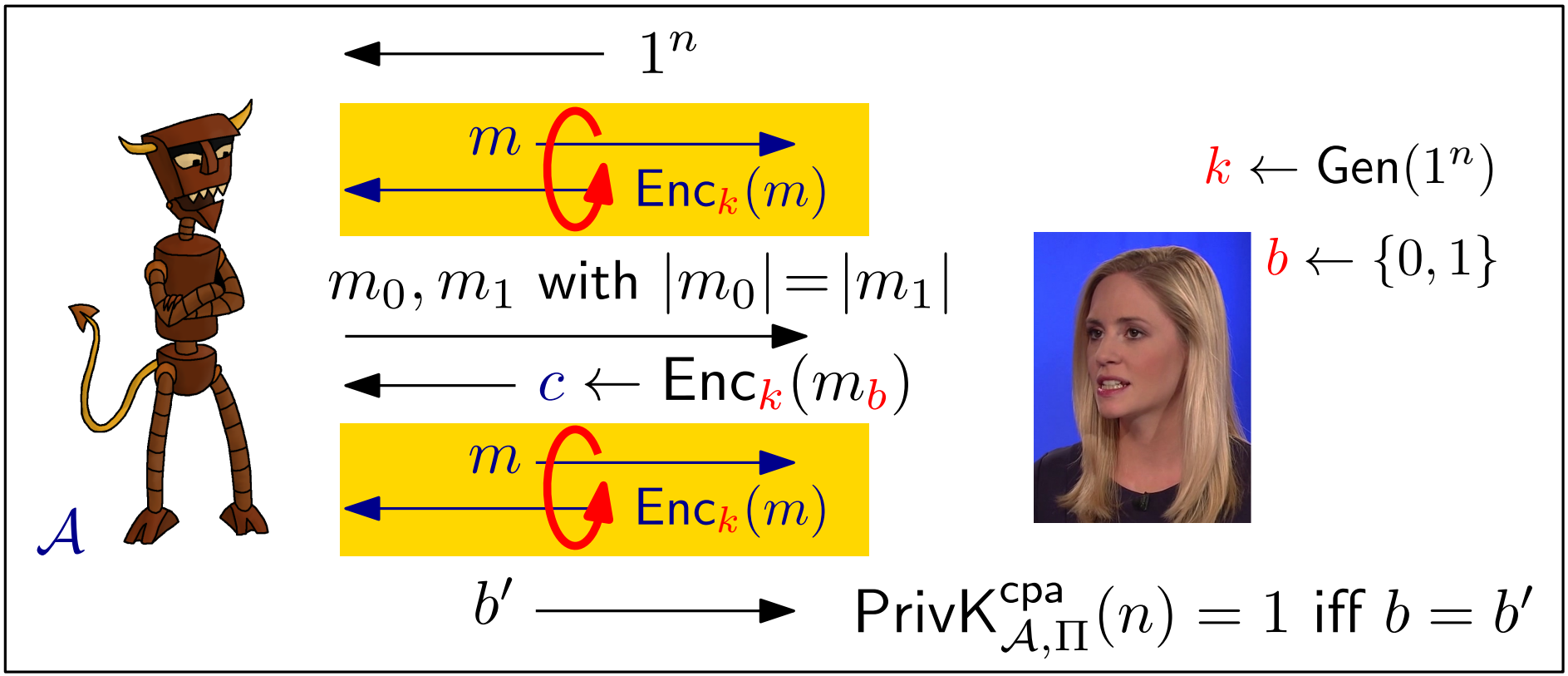
  - Find the key?
  - Recover the plaintext
  - *Guess* a single letter of the plaintext
  - Obtain *any* information about the plaintext

- Adversary's **power:**
  - Sees ciphertexts (one/many)
  - Has seen plaintext/ciphertext pairs
  - Has chosen the plaintexts

    . . . and can ask for *decryption*

Security of encryption

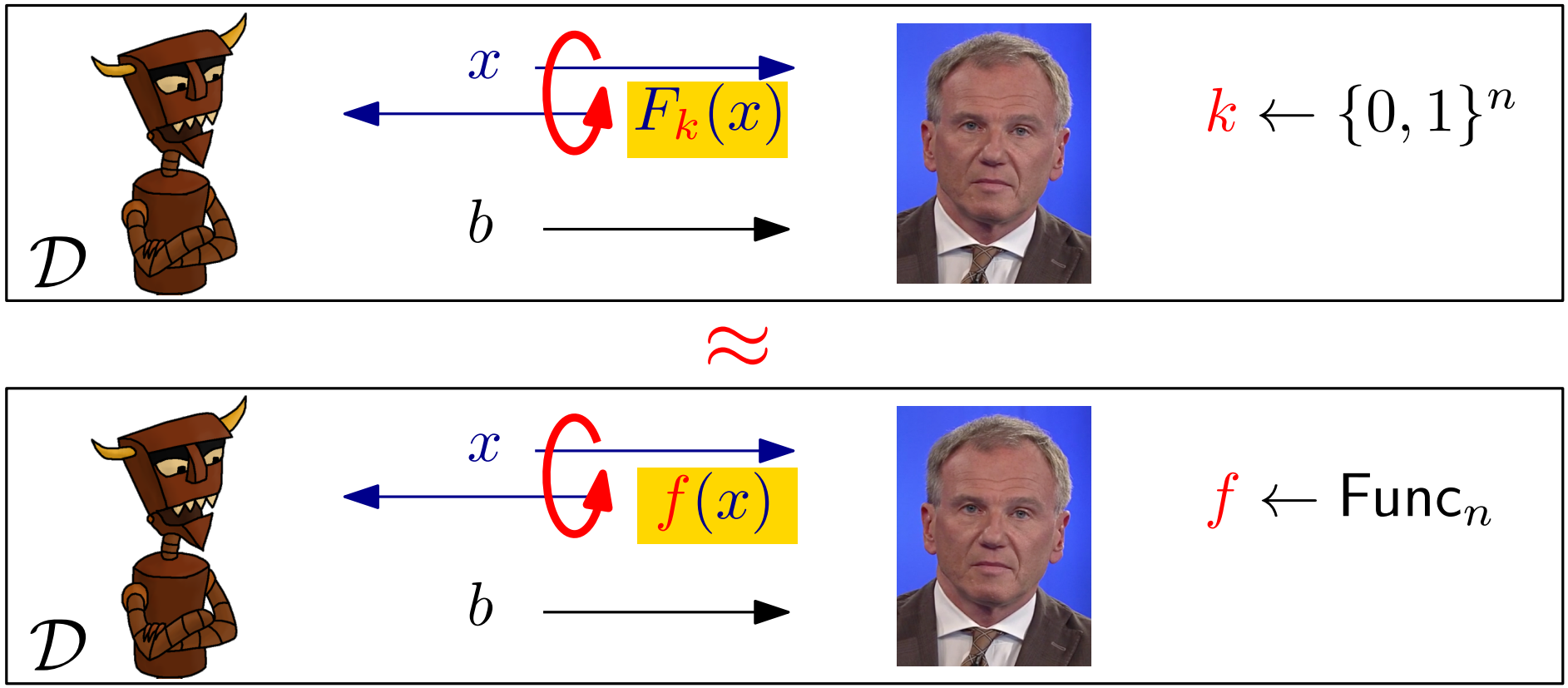
$\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$ for $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$



Definition 3.21. Π *is* **secure against chosen-plaintext attacks** *if for every p.p.t.* $\mathcal{A}$ *there exists negligible* $\varepsilon(\cdot)$*:*

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \varepsilon(n)$$

Pseudorandom functions



Definition 3.24. A keyed function F is a **pseudorandom function** (PRF) if for all p.p.t. $\mathcal{D}$ there exists negl. $\varepsilon(\cdot)$:

$$\left| \Pr_{k \leftarrow \{0,1\}^n} \left[\mathcal{D}^{F_k(\cdot)}(1^n) = 1 \right] - \Pr_{f \leftarrow \mathsf{Func}_n} \left[\mathcal{D}^{f(\cdot)}(1^n) = 1 \right] \right| \leq \varepsilon(n)$$

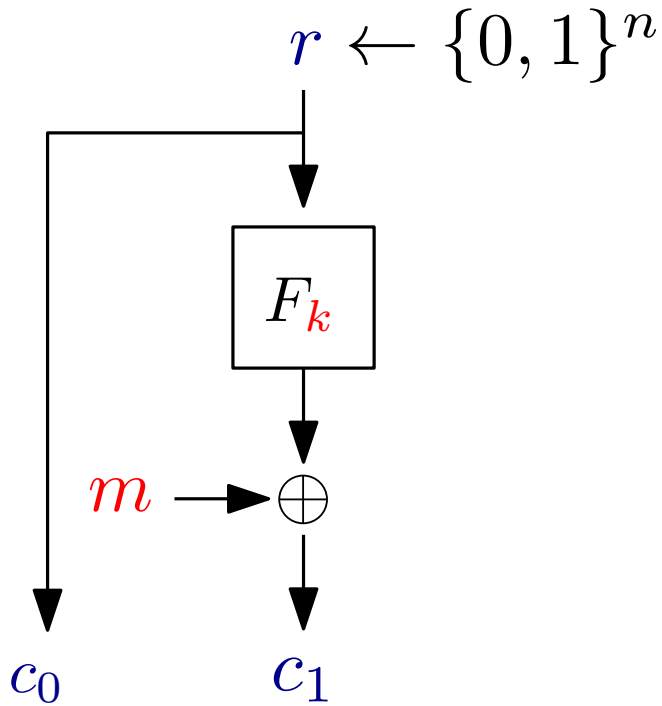
CPA-secure encryption

Construction 3.28. Let $F\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ PRF

$\mathsf{Gen}(1^n)$: return $k \leftarrow \{0,1\}^n$

$\mathsf{Enc}_k(m)$: $r \leftarrow \{0,1\}^n$; return $c := \big(r,\, F_k(r) \oplus m\big)$

$\mathsf{Dec}_k((c_0, c_1))$: return $m := F_k(c_0) \oplus c_1$



Proof idea

$\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$ for $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$

$\mathcal{A}$

1^n

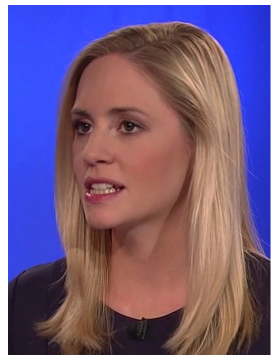
$k \leftarrow \{0,1\}^n$

m'_i

$(r_i, F_k(r_i) \oplus m'_i)$

for $r_i \leftarrow \{0,1\}^n$

$b \leftarrow \{0,1\}$



m_0, m_1

$(r^*, F_k(r^*) \oplus m_b)$

for $r^* \leftarrow \{0,1\}^n$

$\ldots$

b' $\longrightarrow$ $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1$ iff $b = b'$

Proof idea

"Ideal game" for $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$

1^n

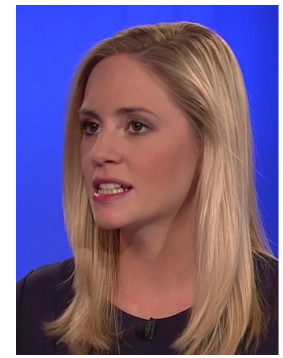
m'_i

$(r_i, f(r_i) \oplus m'_i)$

for $r_i \leftarrow \{0,1\}^n$

$f \leftarrow \mathsf{Func}_n$

$b \leftarrow \{0,1\}$



m_0, m_1

$(r^*, f(r^*)) \oplus m_b)$

for $r^* \leftarrow \{0,1\}^n$

$\ldots$

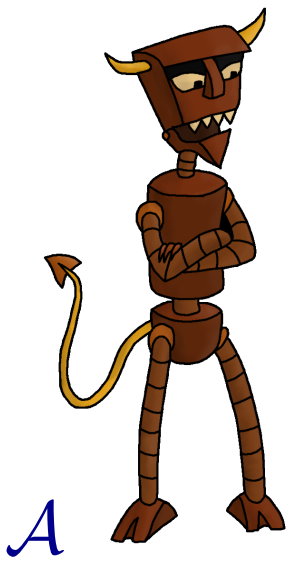
b' $\longrightarrow$ $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1$ iff $b = b'$

. . . "indistinguishable" by PRF security

$\Delta \leq \varepsilon(n)$

Proof idea

"Ideal game" for $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$



$\mathcal{A}$

1^n

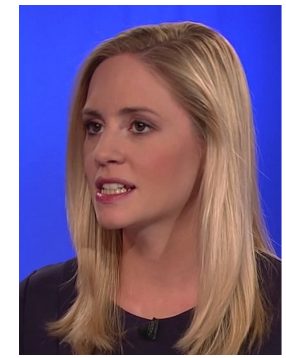
m_i'

$(r_i, f(r_i) \oplus m_i')$

for $r_i \leftarrow \{0,1\}^n$

$f \leftarrow \text{Func}_n$

$b \leftarrow \{0,1\}$



m_0, m_1

$(r^*, f(r^*)) \oplus m_b)$

for $r^* \leftarrow \{0,1\}^n$

$\ldots$

b' $\longrightarrow$ $\text{PrivK}^{\text{cpa}}_{\mathcal{A},\Pi}(n) = 1$ iff $b = b'$

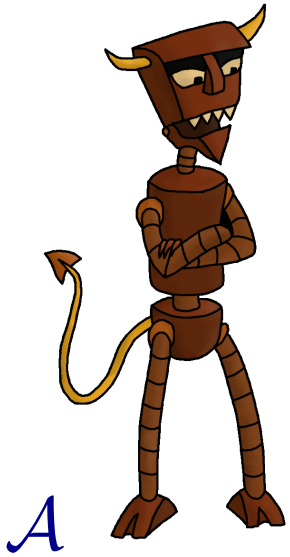
can $\mathcal{A}$ still win? ...not if $f(r^*)$ is never used elsewhere!

$\Rightarrow$ $\leq q$ points

$\Rightarrow$ probability of collision $\leq q/2^n$

Proof idea

"Ideal game" for $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$



$\mathcal{A}$

1^n

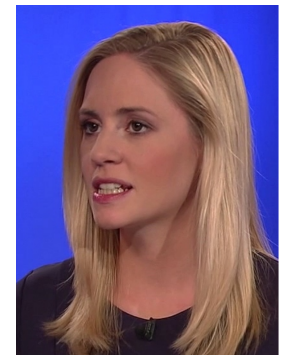
m'_i

$(r_i, f(r_i) \oplus m'_i)$

for $r_i \leftarrow \{0,1\}^n$

$f \leftarrow \mathsf{Func}_n$

$b \leftarrow \{0,1\}$



m_0, m_1

$(r^*, f(r^*)) \oplus m_b)$

for $r^* \leftarrow \{0,1\}^n$

$\ldots$

b' $\longrightarrow$ $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1$ iff $b = b'$

Theorem 3.29. For all p.p.t. $\mathcal{A}$:

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \underbrace{\varepsilon(n) + q(n)/2^n}_{\text{negligible}}$$

CPA-secure encryption

Construct CPA-secure encryption for n-bit messages from PRF (or blockcipher)

- Encrypting longer messages? – split m into n-bit blocks
  – $\mathsf{Enc}_k(m_1, \ldots, m_t) = \mathsf{Enc}_k(m_1), \ldots, \mathsf{Enc}_k(m_t)$

Recall: CPA-security implies multi-message security

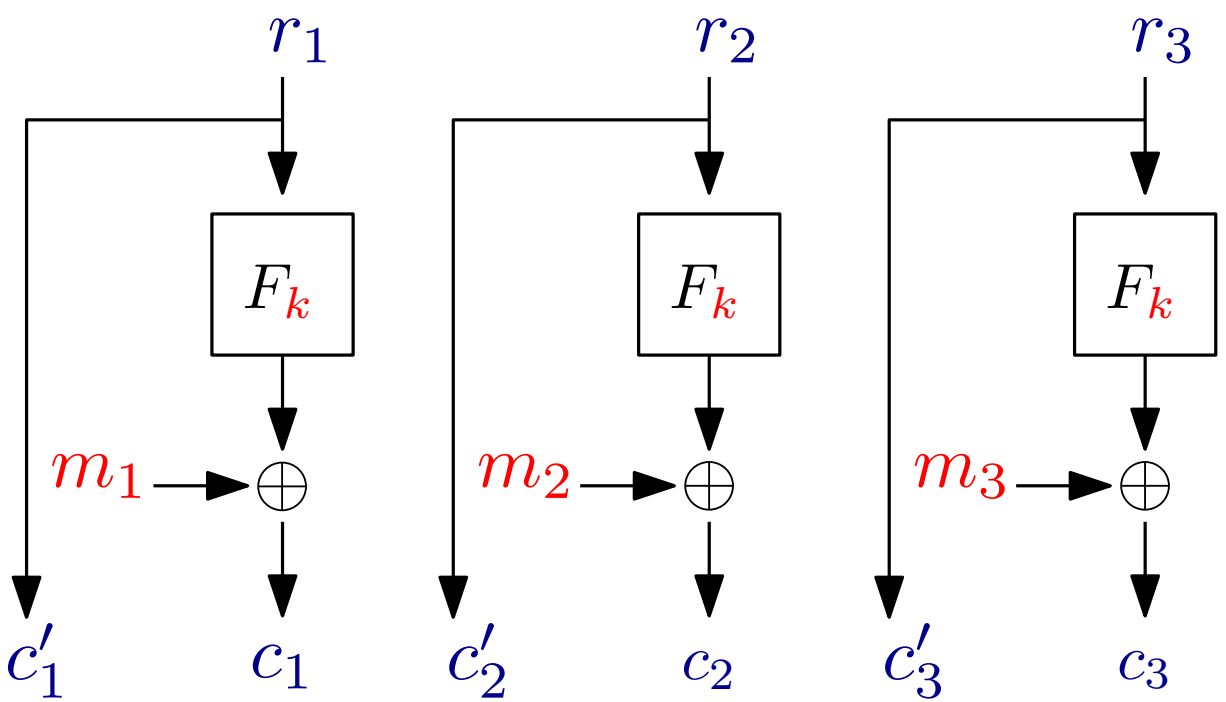
- (CPA-)**secure?** ✓
- **efficient?**

Ciphertext expansion:

$|c| = 2 \cdot |m|$

(not optimal)

Expansion necessary?



Modes of operation

§3.6.3

Modes of operation

To encrypt messages that are longer than n bits with a PR**P**/block cipher F, one uses a

block-cipher mode of operation. Here:

- ECB *(electronic code book)*
- CBC *(cipher block chaining)*
- CTR *(counter mode)*

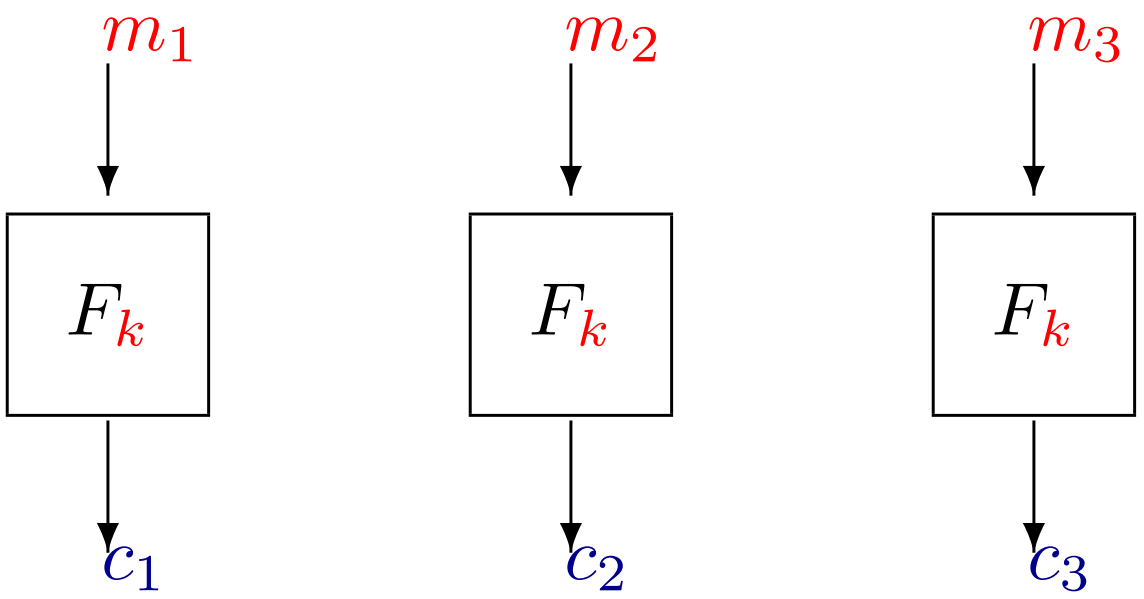
ECB (old standard)

- divide m into t blocks of n bits: $m_1, m_2, \ldots, m_t$
- use *padding* if necessary

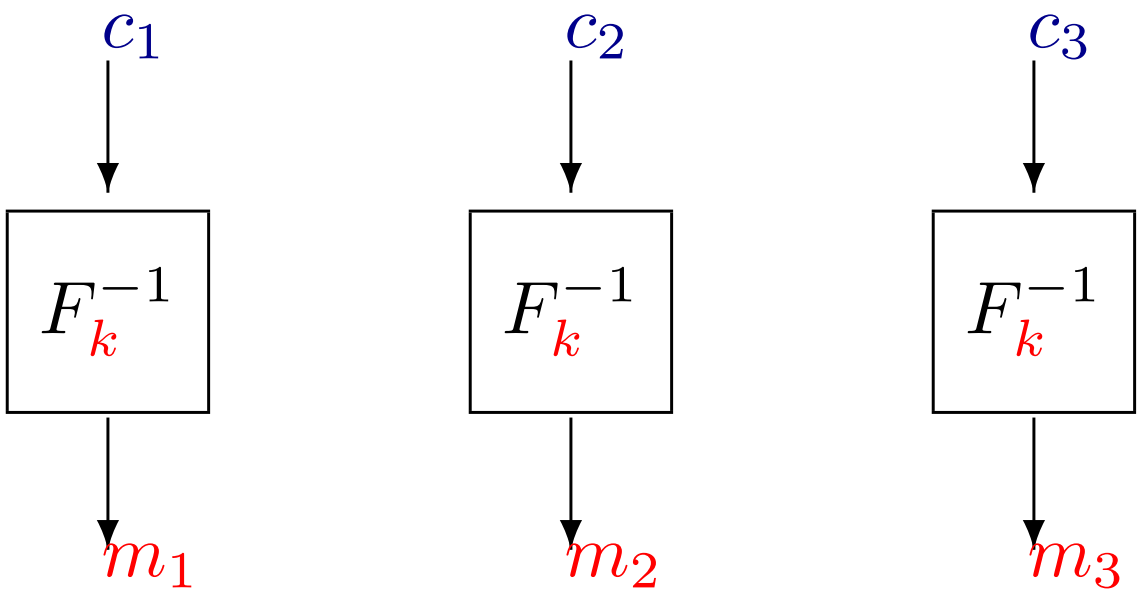
$\text{Enc}_k(m_1, \ldots, m_t) = F_k(m_1), \ldots, F_k(m_t)$

ECB mode

Encryption:



Decryption:

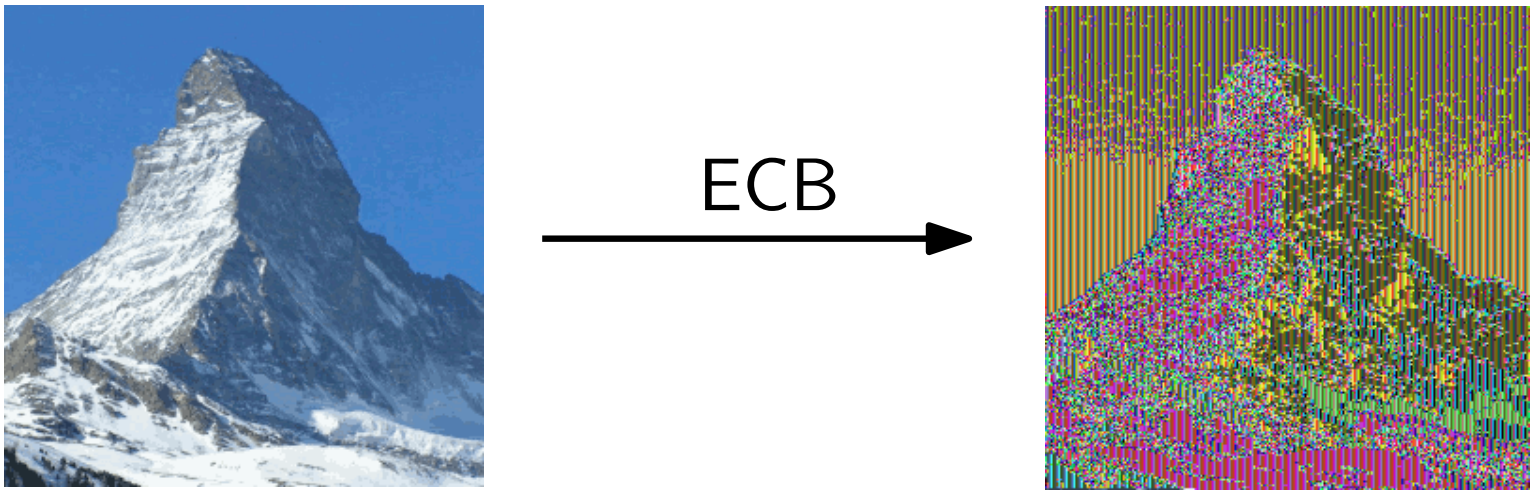


ECB mode

Secure? deterministic $\Rightarrow$ not CPA-secure! EAV-secure?

Problem: if $m_i = m_j$, then $c_i = c_j$

$\Rightarrow$ patterns appear in the ciphertext



$\Rightarrow$ Not even secure *in presence of EAVesdropper*

Do not use!

(define $2n$-bit challenge messages:
$m_0 := 0 \ldots 00 \ldots 0$,
$m_1 := 0 \ldots 01 \ldots 1$)

CPA-secure encryption

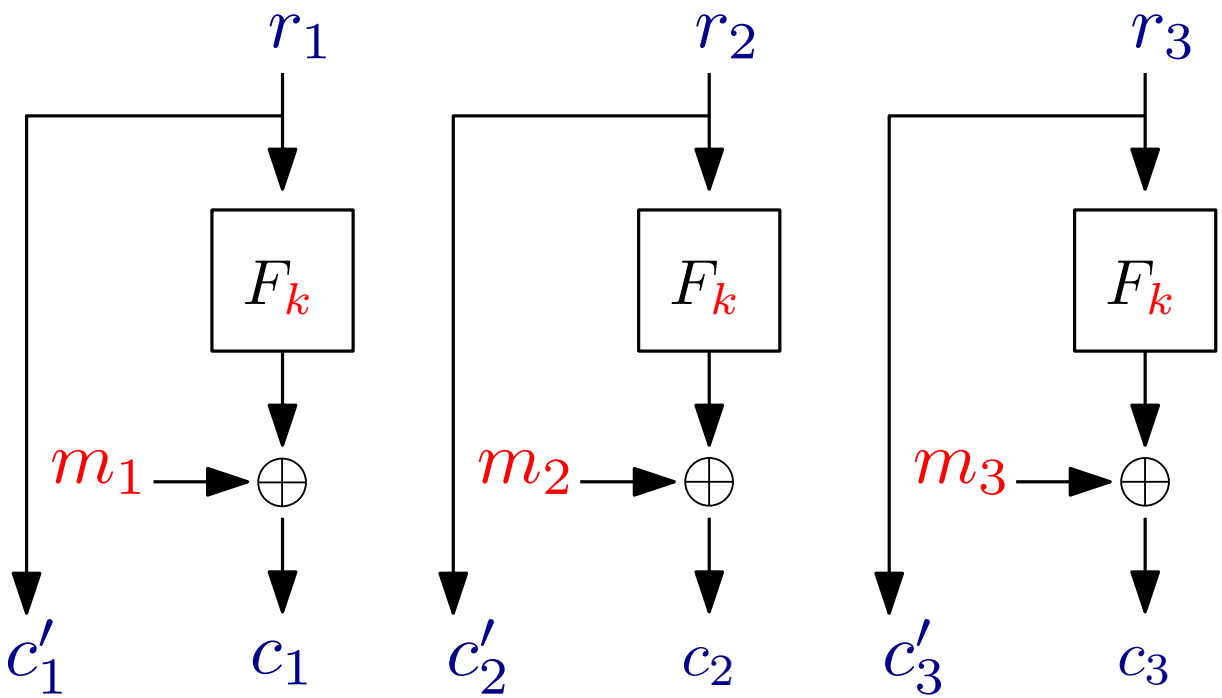
to our secure scheme . . .

- **efficient?**

Ciphertext expansion:

$|c| = 2 \cdot |m|$

(not optimal)



CPA-secure encryption

to our secure scheme . . .

- **efficient**

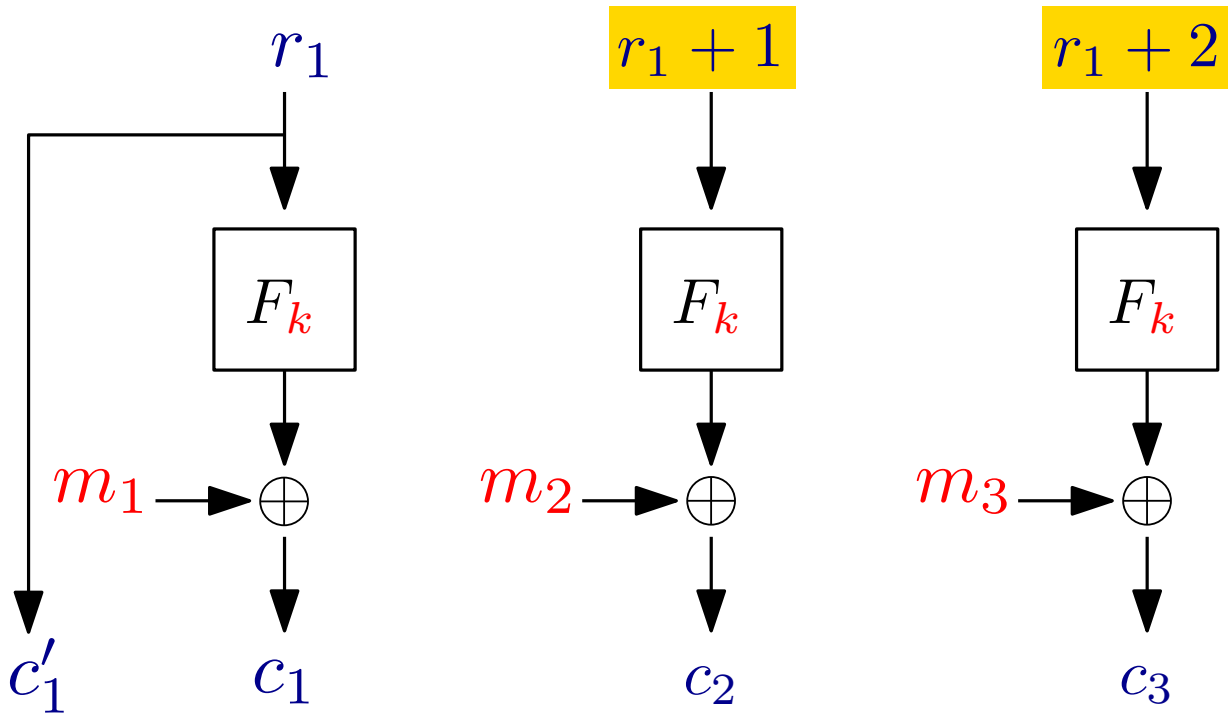
Ciphertext expansion:

~~$|c| = 2 \cdot |m|$~~

(not optimal)

$|c| = |m| + n$

(optimal!)



CTR mode

CTR (counter)

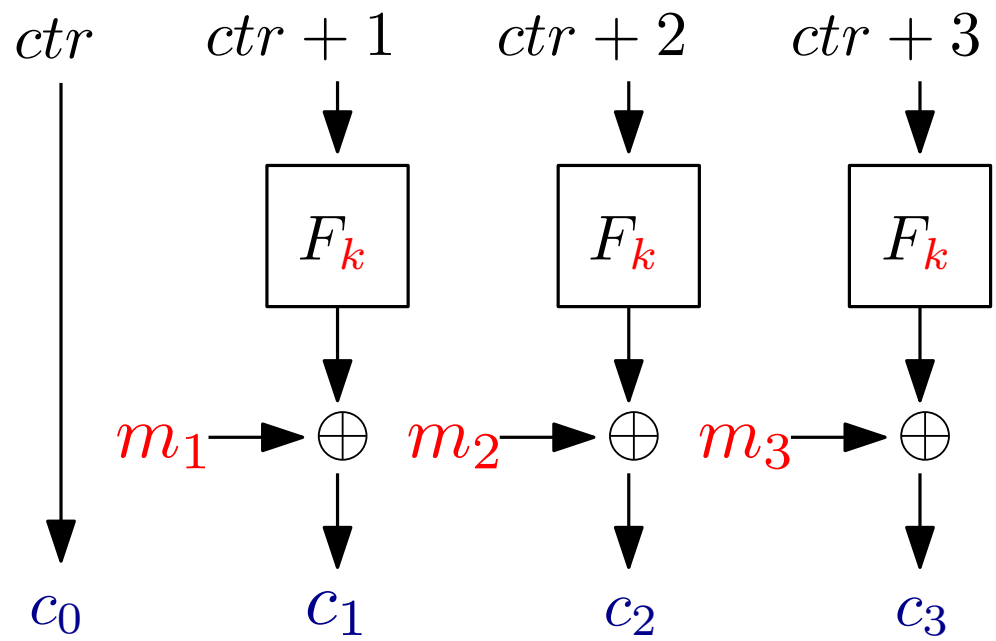
- Can be viewed as **stream cipher** $\rightarrow$ disk encryption, etc.

$\mathsf{Enc}_k(m_1, \ldots, m_t)$:

- sample $ctr \leftarrow \{0,1\}^n$
- For $i = 1 \ldots t$:
  $c_i := F_k(ctr + i) \oplus m_i$
- Return $c_0 := ctr, c_1, \ldots, c_t$

$\mathsf{Dec}_k(c_0, c_1, \ldots, c_t)$:

- For $i = 1 \ldots t$:
  $m_i := F_k(c_0 + i) \oplus c_i$
- Return $m_1, \ldots, m_t$



F need not be invertible!
(PR**F** suffices)

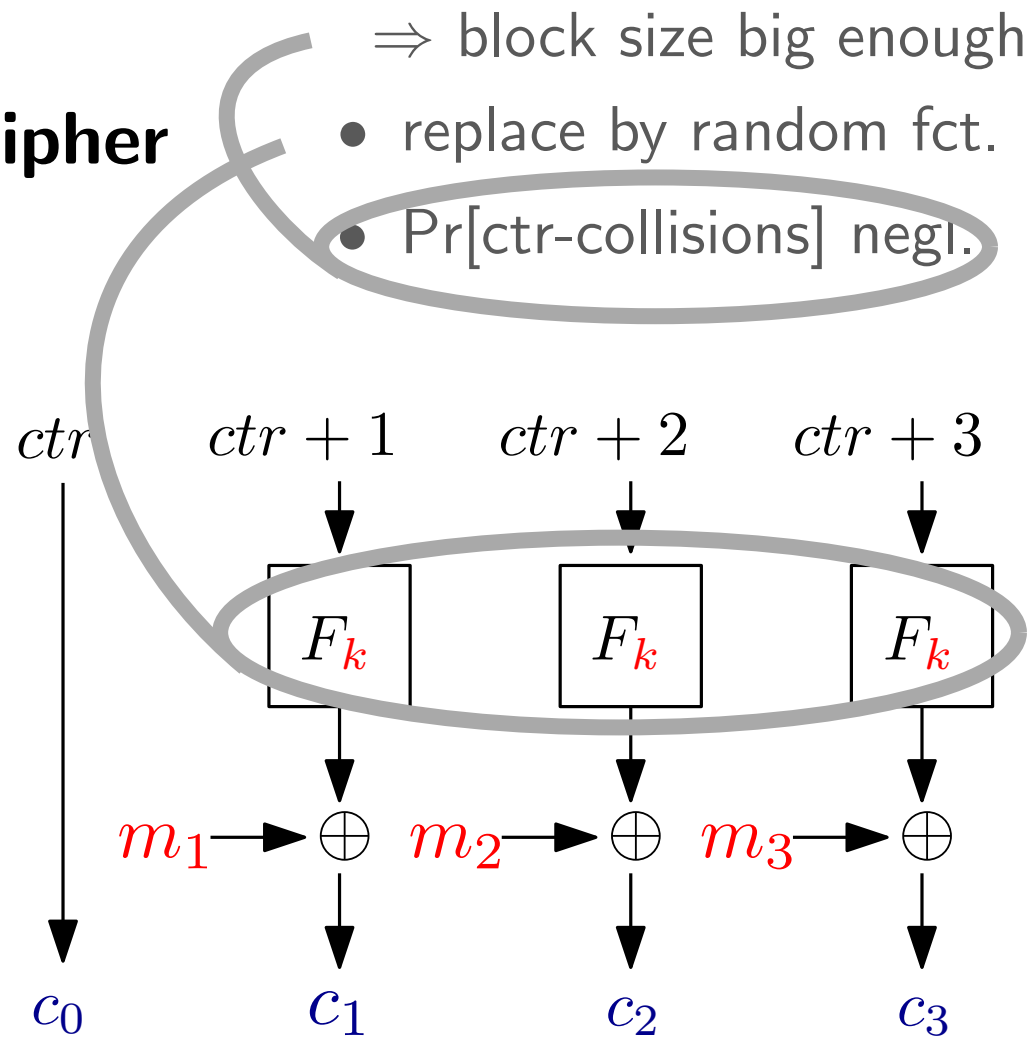
CTR mode

CTR (counter)

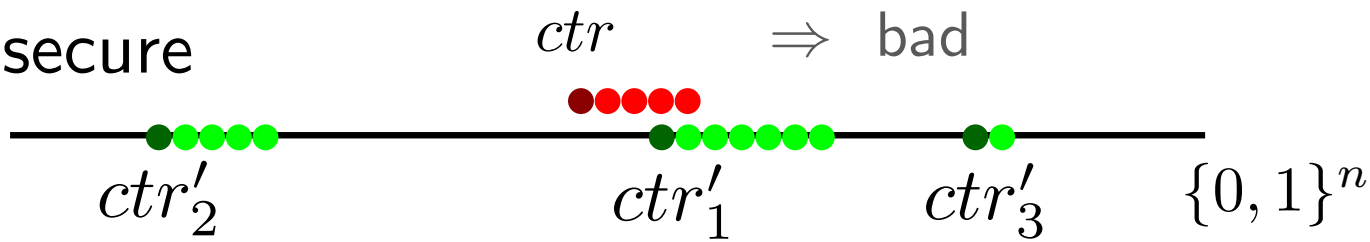
- Can be viewed as **stream cipher**

$\mathsf{Enc}_k(m_1, \ldots, m_t)$:

- sample $ctr \leftarrow \{0,1\}^n$
- For $i = 1 \ldots t$:
  $c_i := F_k(ctr + i) \oplus m_i$
- Return $c_0 := ctr, c_1, \ldots, c_t$



Theorem 3.33.
If F is pseudorandom then
CTR mode is CPA-secure



CBC mode

CBC (Cipher Block Chaining)

$\mathsf{Enc}_k(m_1, \ldots, m_t)$: for $m_i \in \{0,1\}^n$

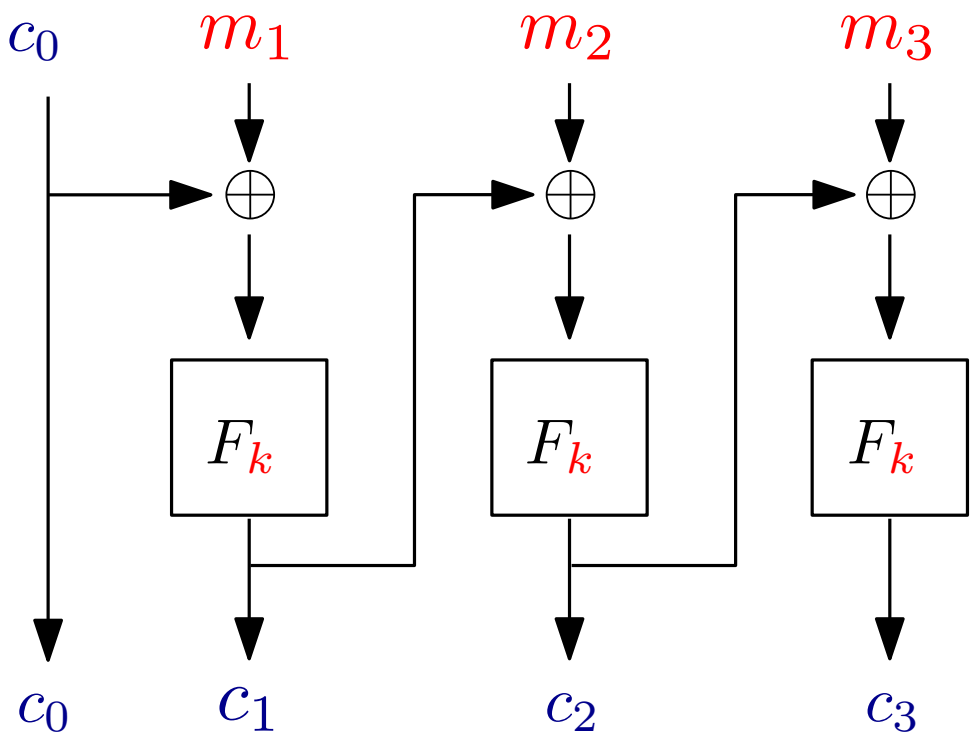
- sample $c_0 \leftarrow \{0,1\}^n$ (a.k.a. "initialization vector" (IV))
- For $i = 1 \ldots t$:

  $c_i := F_k(m_i \oplus c_{i-1})$
- Return $c_0, c_1, \ldots, c_t$

$\mathsf{Dec}_k(c_0, c_1, \ldots, c_t)$:

- For $i = 1 \ldots t$:

  $m_i := F_k^{-1}(c_i) \oplus c_{i-1}$
- Return $m_1, \ldots, m_t$



CBC mode

CBC (Cipher Block Chaining)

$\mathsf{Enc}_k(m_1, \ldots, m_t)$: for $m_i \in \{0,1\}^n$

- sample $c_0 \leftarrow \{0,1\}^n$ (a.k.a. "initialization vector" (IV))
- For $i = 1 \ldots t$:

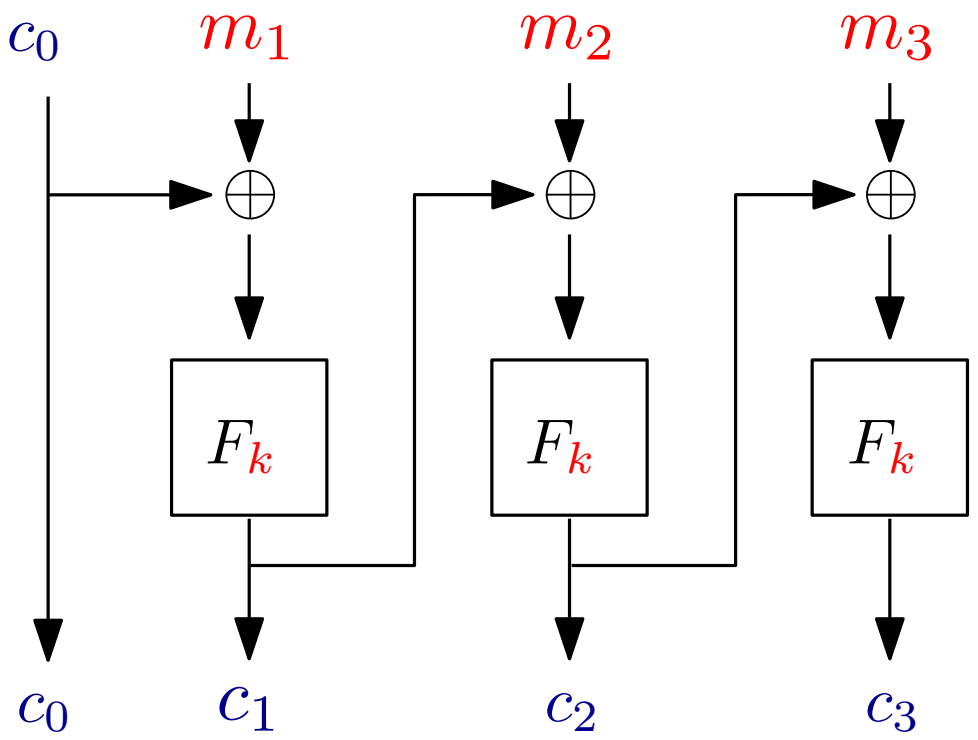
$$c_i := F_k(m_i \oplus c_{i-1})$$

- Return $c_0, c_1, \ldots, c_t$

Ciphertext expansion: one block

If F is a pseudorandom permutation *then CBC mode is CPA-secure (Thm. 3.32)*

CBC is used in SSL/TLS



Chosen-ciphertext attacks

§5.1

Recall. . .

Kerckhoffs' Principle: The adversary knows the scheme

Auguste Kerckhoffs: *La cryptographie militaire* (1883)

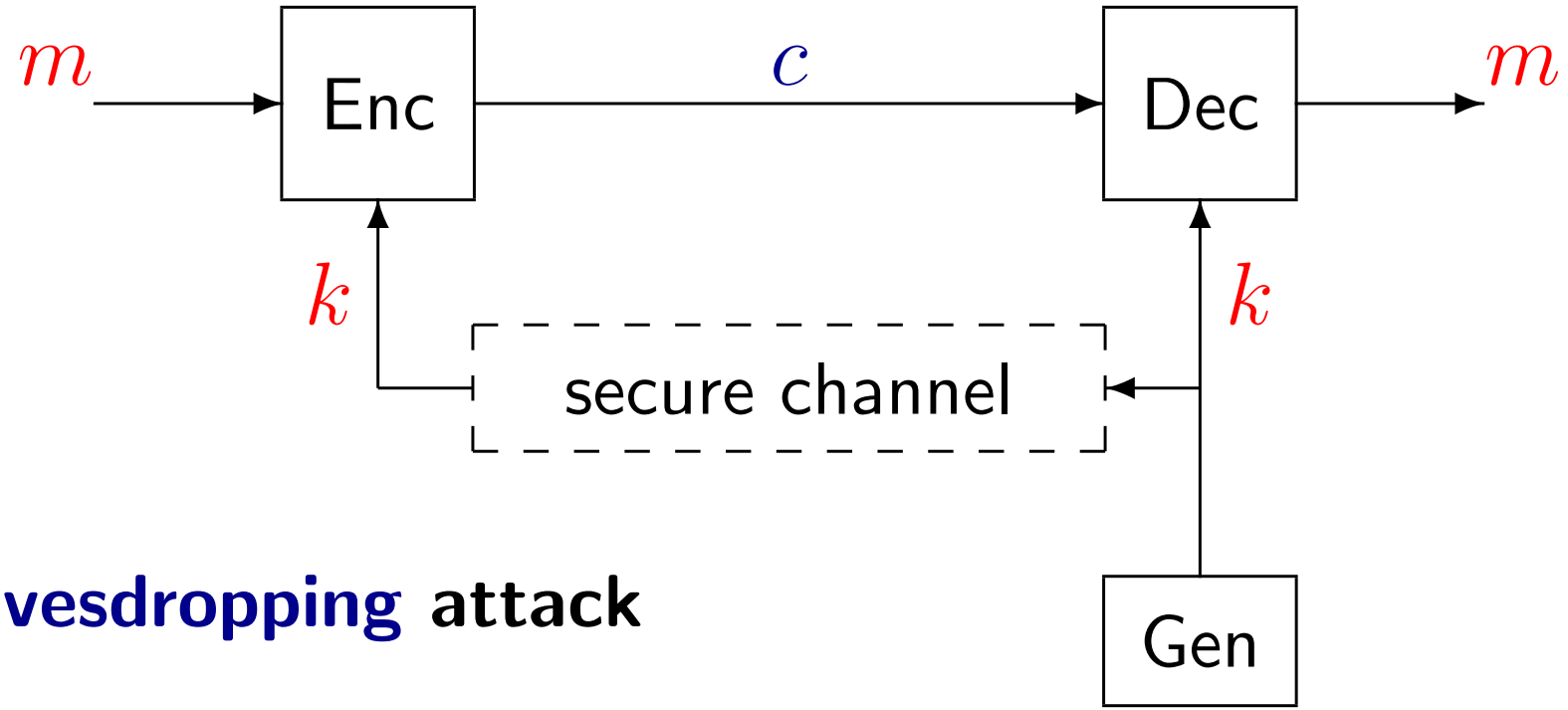
- Adversary's **goals:**
  - Find the key?
  - Recover the plaintext
  - *Guess* a single letter of the plaintext
  - Obtain *any* information about the plaintext

- Adversary's **power:**
  - Sees ciphertexts (one/many)
  - Has seen plaintext/ciphertext pairs
  - Has chosen the plaintexts
    . . . and can ask for *decryption*

CCA-security

Alice

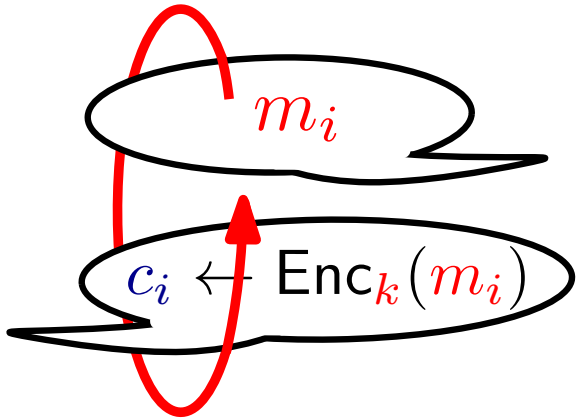
Bob



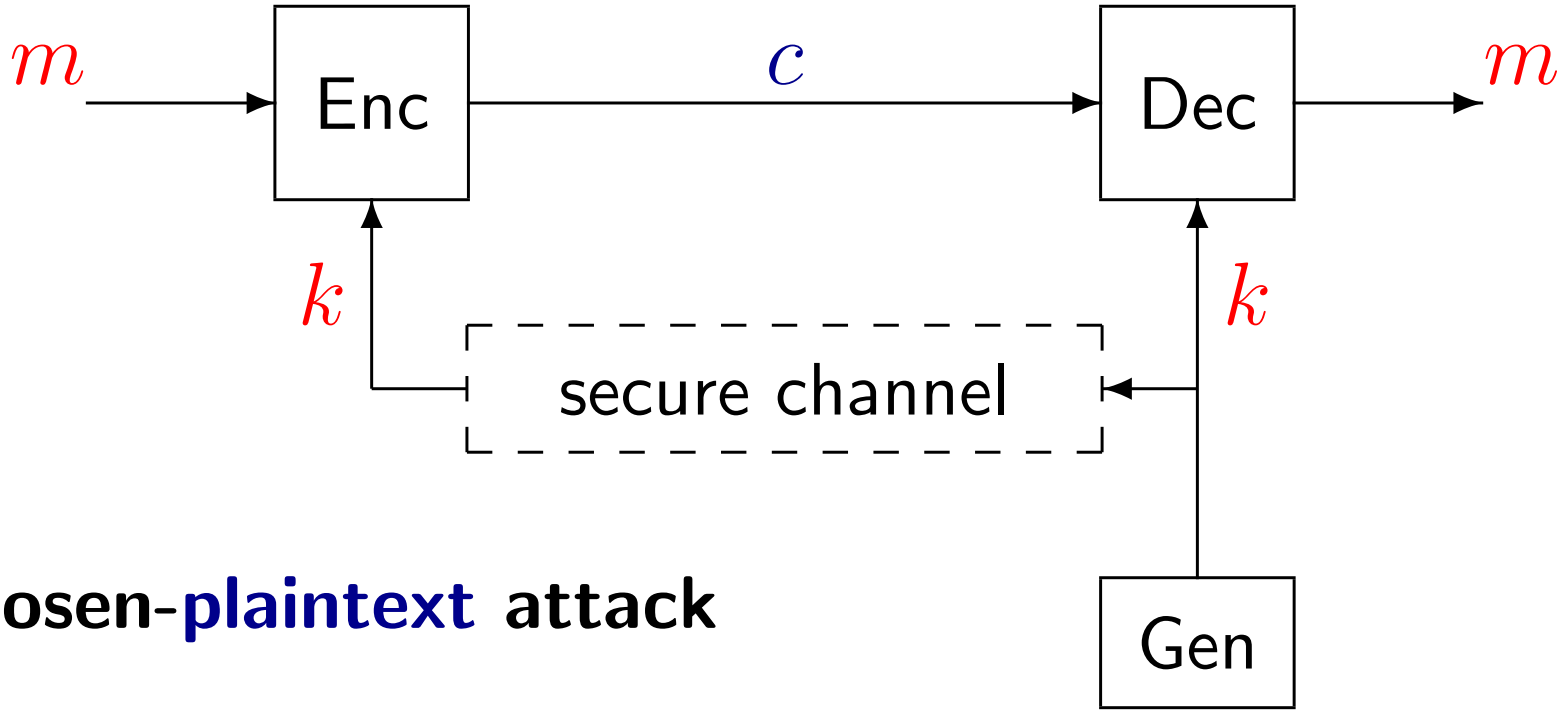
Eavesdropping attack

CCA-security

Alice



Bob



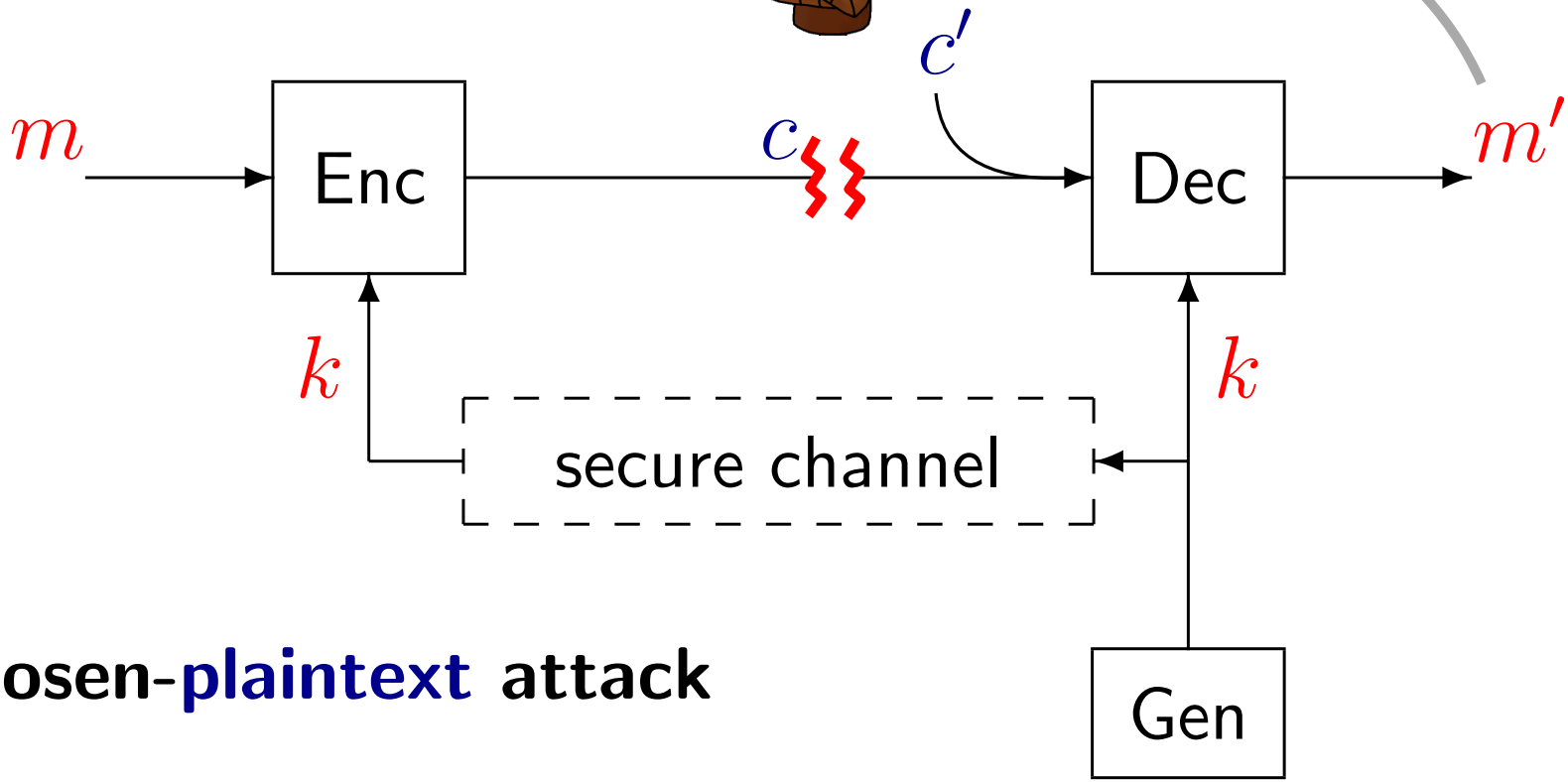
Chosen-plaintext attack

CCA-security

Active adversaries. . .

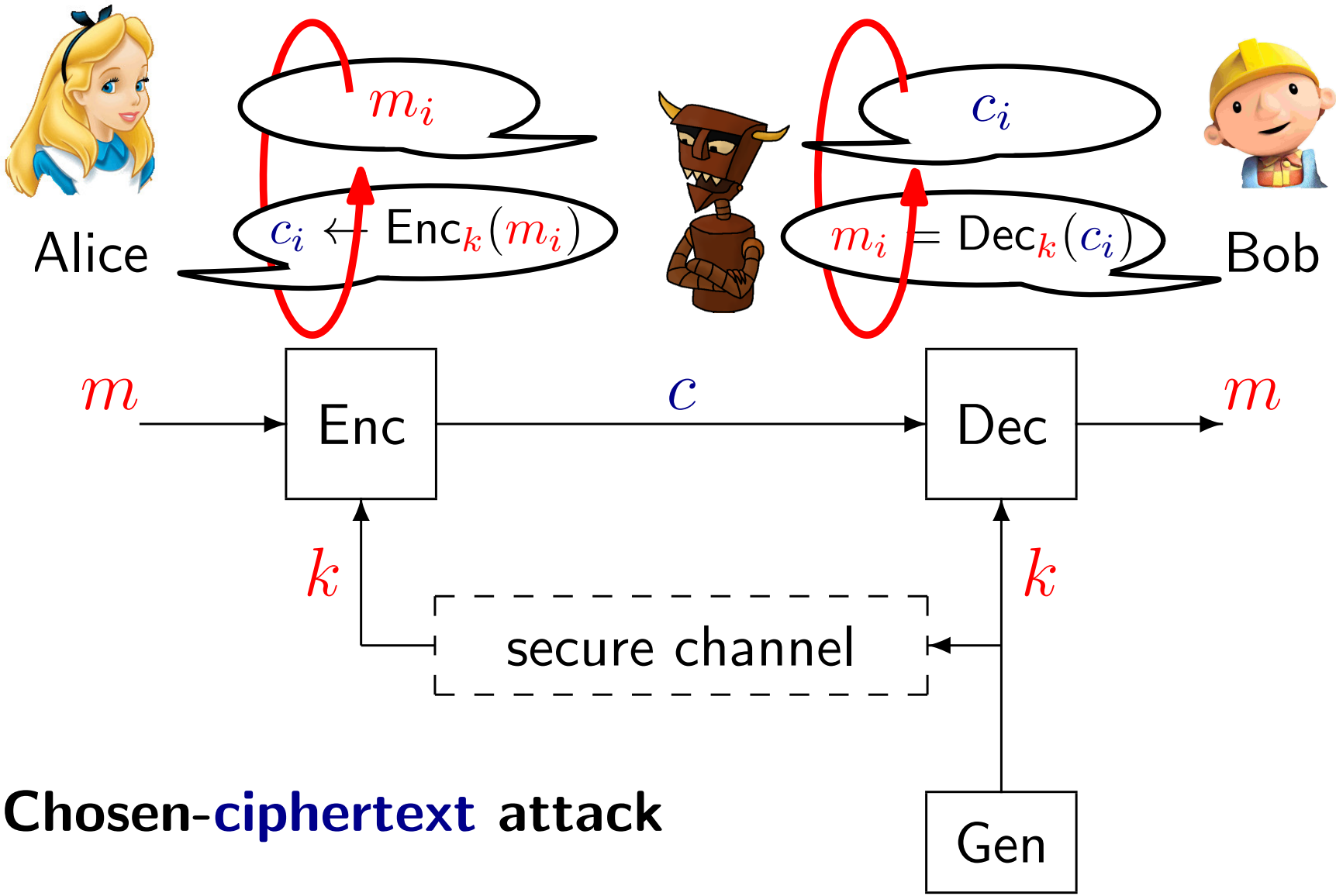
Alice

Bob



Chosen-plaintext attack

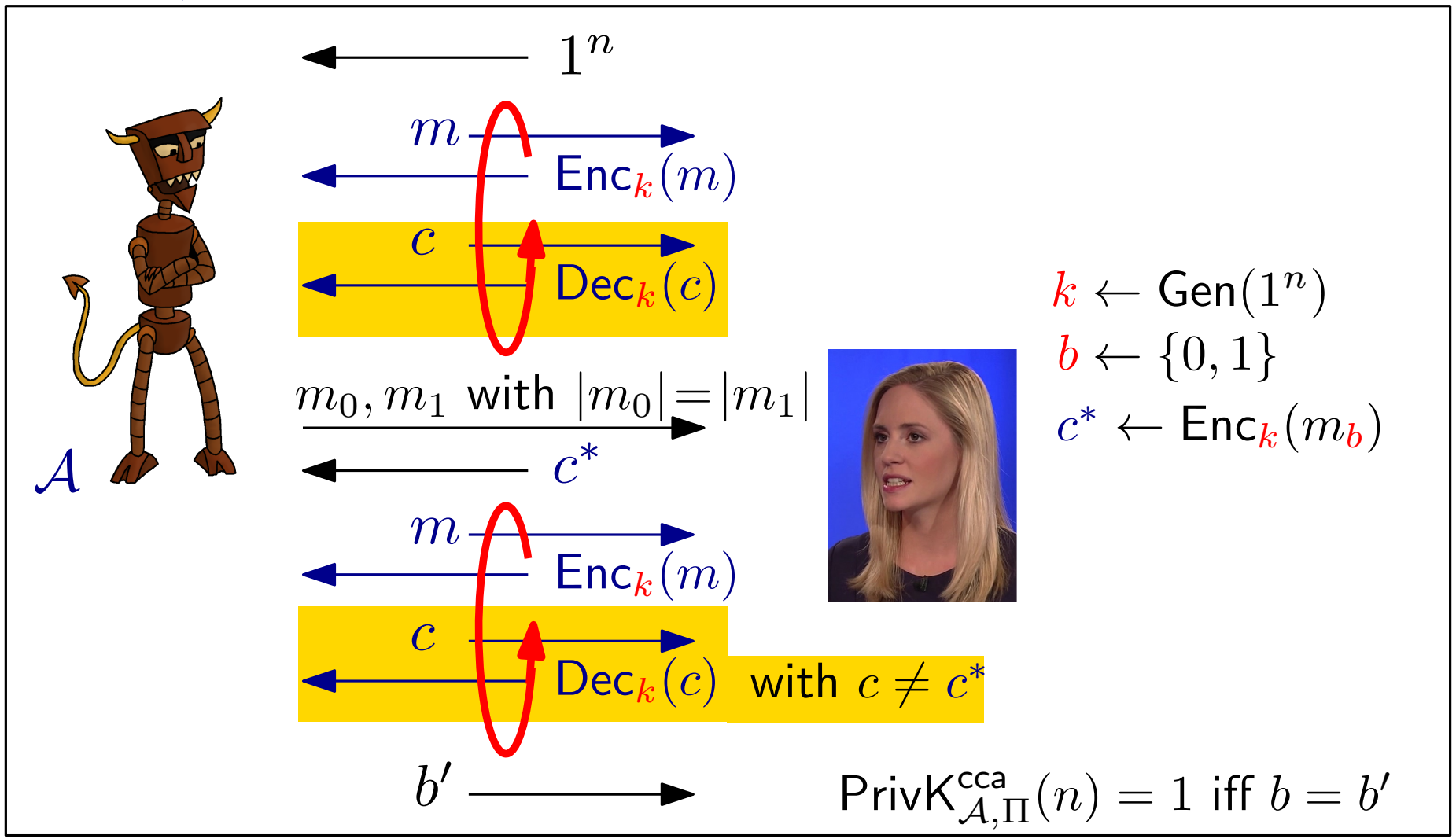
CCA-security



Chosen-ciphertext attack

CCA-security

$\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$ for $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$



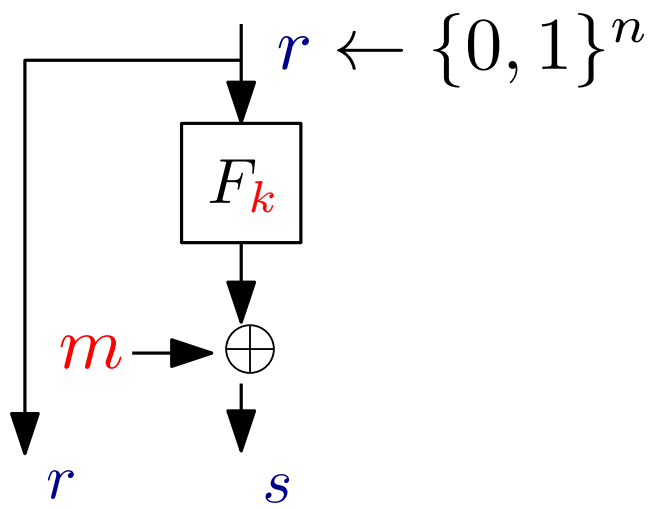
Definition 5.1. Π *is* **secure against chosen-ciphertext attacks** *if for every p.p.t.* $\mathcal{A}$ *there exists negligible* $\varepsilon(\cdot)$*:* $\Pr\left[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \varepsilon(n)$

CCA-security of studied schemes

Consider first CPA-scheme:

$\mathsf{Enc}_k(m) := (r, F_k(r) \oplus m)$

for $r \leftarrow \{0,1\}^n$



Attack:

- Given challenge $(r, s := F_k(r) \oplus m_b)$
- For any $\Delta \in \{0,1\}^n, \Delta \neq 0^n$: query $\mathsf{Dec}_k\big((r, s \oplus \Delta)\big)$

$$= \underbrace{F_k(r) \oplus (s}_{= m_b} \oplus \Delta)$$

- Learn $m_b \oplus \Delta$, and thus b

$\Rightarrow$ CCA-secure scheme must not be *malleable*

("c cannot be changed into a c' of *related* message")

CCA-security

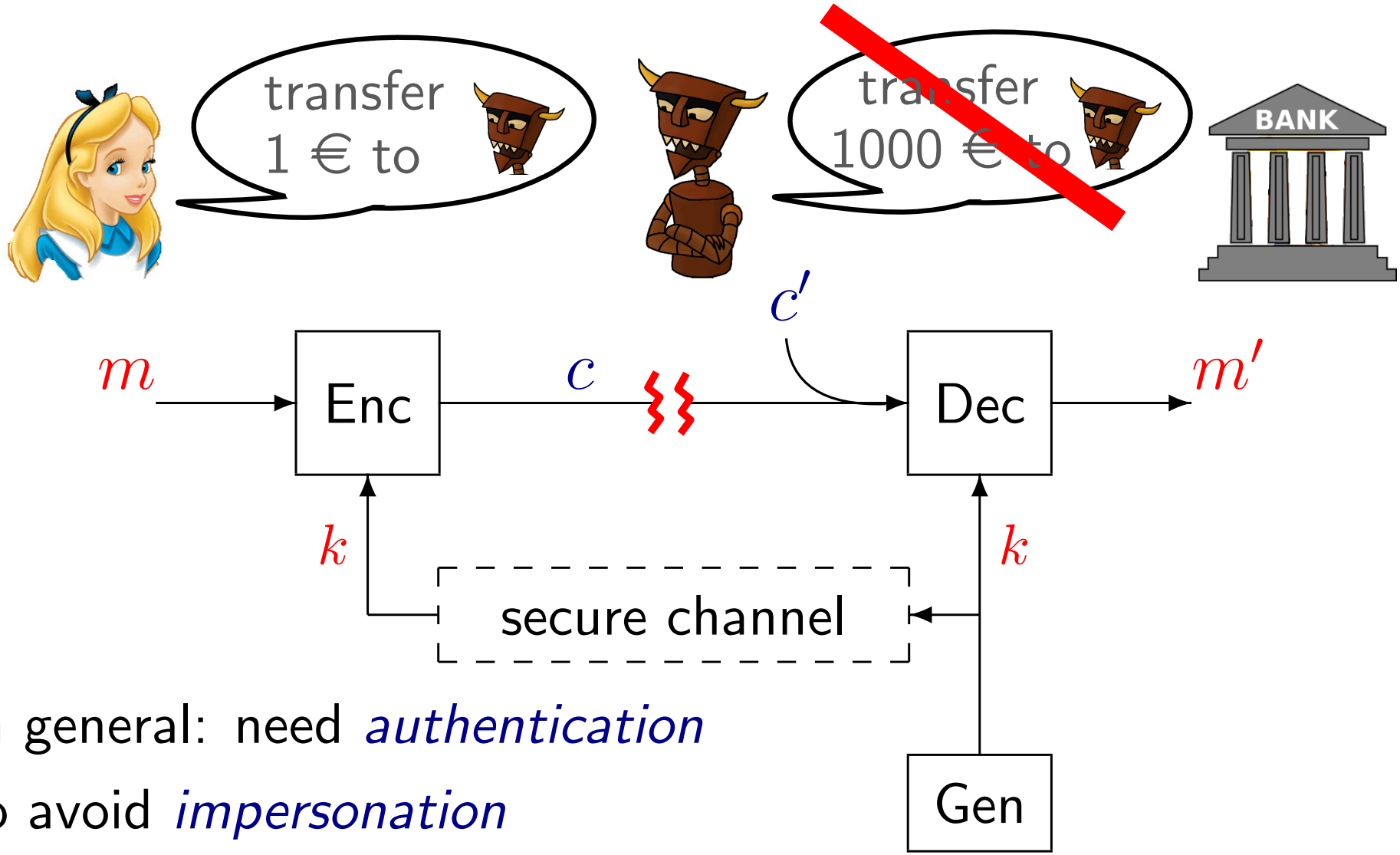
- Too paranoid?
- No! *"Padding-oracle attack"* (book, §5.1.1) against SSL
  ("PKCS7 padding" in CBC mode)

  Attacker only needs to learn *if* decryption succeeded!

- *None* of the schemes so far are CCA-secure

  (CBC, CTR, etc. are all *malleable*)

Malleability



In general: need *authentication*

to avoid *impersonation*

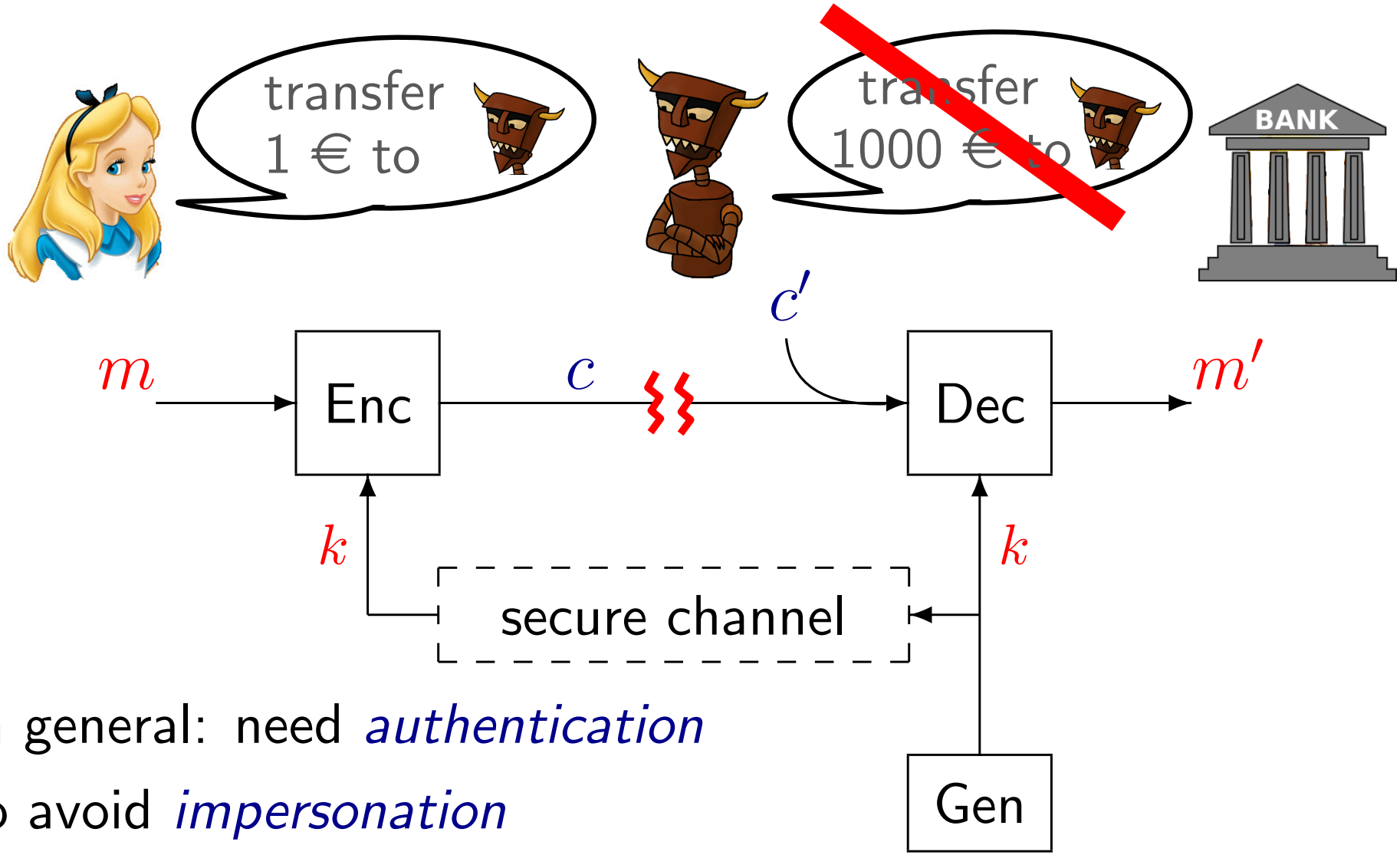
Introduction to Cryptography

(Lecture 8: authentication, AE)

Elena Andreeva

TUW 2025W

Malleability



In general: need *authentication*

to avoid *impersonation*

Message-authentication codes

§4

Secrecy vs. authenticity

General goal: Enable **secure** communication

So far: only concerned about **secrecy** of messages

(eavesdropping-, CPA-, CCA-security)

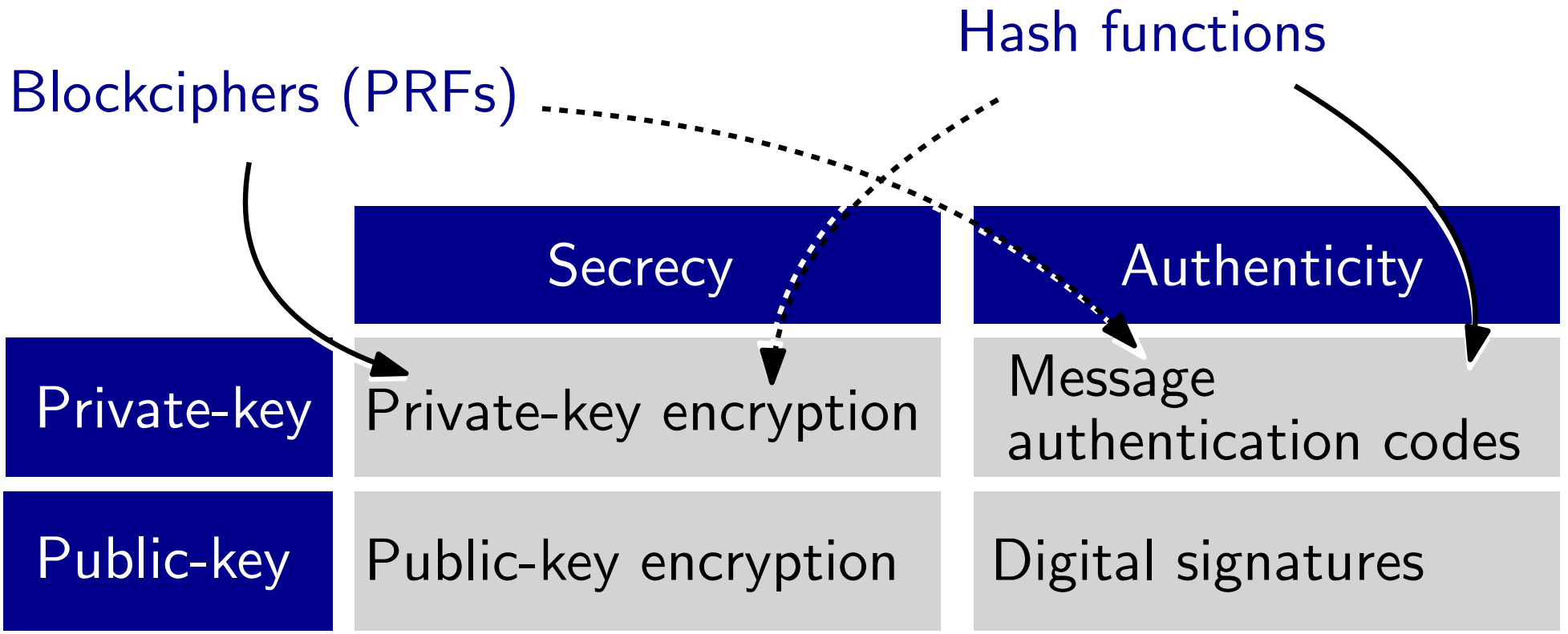
Not enough: need to ensure that messages were

- *sent by claimed sender* (authenticity)
- *not modified in transit* (integrity)

Secrecy $\not\Rightarrow$ & $\not\Leftarrow$ authenticity are **different** goals!

message-authentication codes

Topics


Hash functions
Blockciphers (PRFs)
Secrecy
Authenticity
Private-key
Private-key encryption
Message authentication codes
Public-key
Public-key encryption
Digital signatures

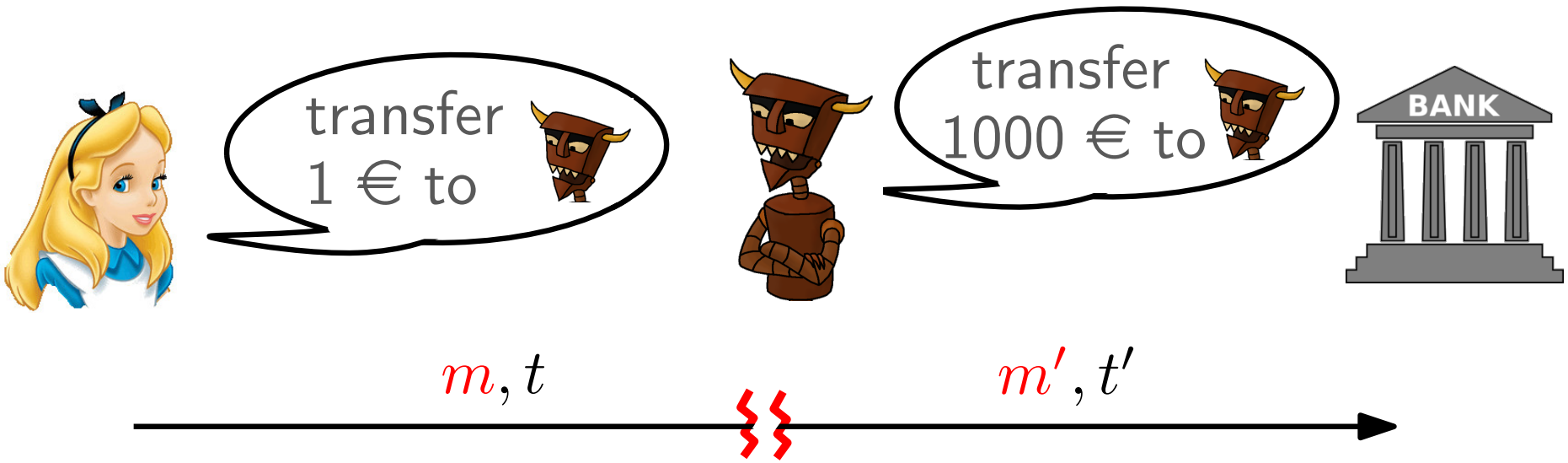
Message-authentication codes

m, t

$t = \mathsf{Mac}_k(m)$

$\mathsf{Vrfy}_k(m, t) = 1$

Message-authentication codes



$t = \mathsf{Mac}_k(m)$

$\mathsf{Vrfy}_k(m', t') = 0$

Message-authentication codes

A **message authentication code** (MAC) is defined by the following p.p.t. algorithms:

$k \leftarrow \mathsf{Gen}(1^n)$**:** given security parameter, return key k (with $|k| \geq n$)

$t \leftarrow \mathsf{Mac}_k(m)$**:** given k and message $m \in \{0,1\}^*$, return **tag** t

$b := \mathsf{Vrfy}_k(m,t)$**:** return $b = 1$ (valid) or $b = 0$ (invalid)

Correctness: For all n, all $k \leftarrow \mathsf{Gen}(1^n)$, all $m \in \{0,1\}^*$:

$$\mathsf{Vrfy}_k(m, \mathsf{Mac}_k(m)) = 1$$

Canonical verification: if Mac is deterministic, can define:

$$\mathsf{Vrfy}_k(m,t) = 1 \;\Leftrightarrow\; \mathsf{Mac}_k(m) = t$$

Security of MACs

Standard security definition:

- *Threat model:* **(adaptive) chosen-message attack**
  - Adversary can obtain tags on messages
- *Security goal:* **existential unforgeability**
  - Adversary cannot forge a tag on *any* other message

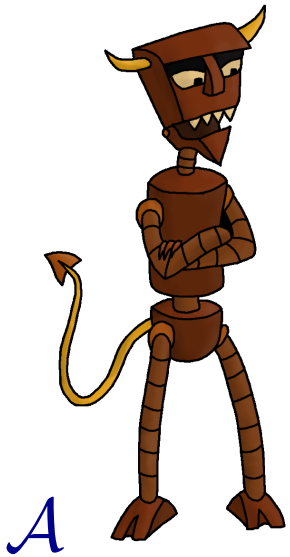
- Paranoid? Is forgery on meaningless m bad?

- We don't know how MACs will be used

  $\Rightarrow$ *strongest possible definition (cf. CCA-security)*

Security of MACs

$\mathsf{MacForge}_{\mathcal{A},\Pi}(n)$ for $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$



$\mathcal{A}$

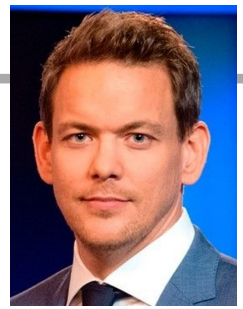
1^n ⟵

$k \leftarrow \mathsf{Gen}(1^n)$

m ⟶

$\mathsf{Mac}_k(m)$ ⟵

$\mathcal{Q} := \mathcal{Q} \cup \{m\}$



(m^*, t^*) ⟶

$\mathsf{MacForge}_{\mathcal{A},\Pi}(n) = 1$ iff: – $\mathsf{Vrfy}_k(m^*, t^*) = 1$

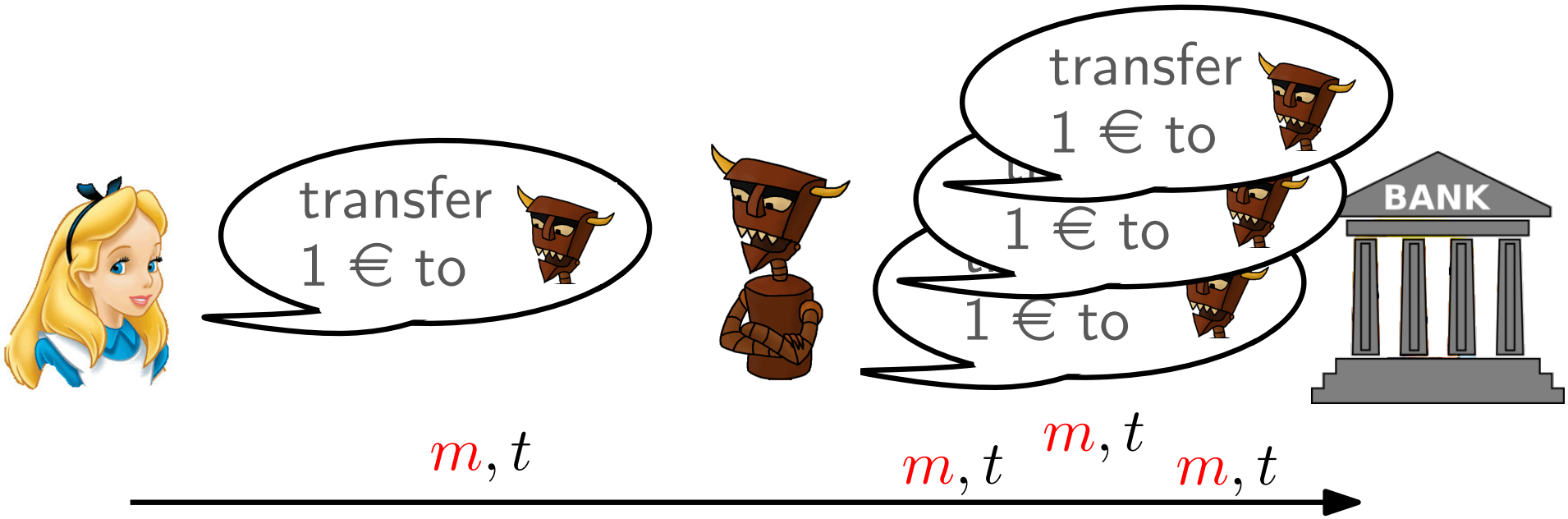
and – $m^* \notin \mathcal{Q}$

Definition 4.2. Π *is* **secure*** *if for every p.p.t.* $\mathcal{A}$ *there exists negligible* $\varepsilon(\cdot)$*:*

$$\Pr\left[\mathsf{MacForge}_{\mathcal{A},\Pi}(n) = 1\right] \leq \varepsilon(n)$$

*existentially unforgeable under adaptive chosen-message attacks

Replay attacks



$t = \mathsf{Mac}_k(m)$

Cannot be prevented by (stateless) MAC

$\Rightarrow$ higher-level measures (time stamps, . . .)

A fixed-length MAC

- Consider message space $\mathcal{M} = \{0,1\}^n$
- Goal: (deterministic) function $\mathsf{Mac}_k(\cdot)$, so that
  - given evaluations for $m_1, m_2, \ldots$
  - it is hard to predict value $\mathsf{Mac}_k(m^*)$ for any new m^*
- Idea: Use a pseudorandom function!

Construction 4.5. Let F be a PRF. Define MAC Π:

$\mathsf{Gen}(1^n)$**:** choose $k \leftarrow \{0,1\}^n$

$\mathsf{Mac}_k(m)$**:** return $F_k(m)$ for $m \in \{0,1\}^n$

$\mathsf{Vrfy}_k(m,t)$**:** return 1 iff $F_k(m) = t$

A fixed-length MAC

Theorem 4.6. Π is a secure MAC

Proof: By reduction against security of PRF F

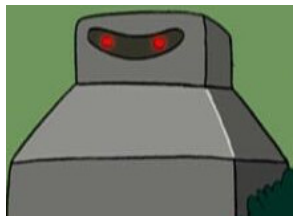
$t =$

m

$F_k(m)$

$f(m)$

Reduction $\mathcal{D}$



β

F PRF $\Rightarrow$ For any p.p.t. $\mathcal{D}$ there exists negl. $\varepsilon(\cdot)$ s.t.

$$\left|\Pr_{k\leftarrow\{0,1\}^n}\left[\mathcal{D}^{F_k(\cdot)}=1\right]-\Pr_{f\leftarrow\mathsf{Func}_n}\left[\mathcal{D}^{f(\cdot)}=1\right]\right|\leq\varepsilon(n)$$

A fixed-length MAC

Theorem 4.6. Π is a secure MAC

Proof: By reduction against security of PRF F

$t =$

m

$F_k(m)$

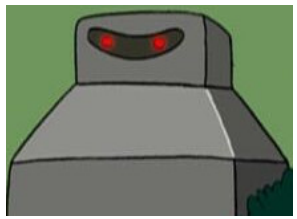
$f(m)$

m^*

t'

β

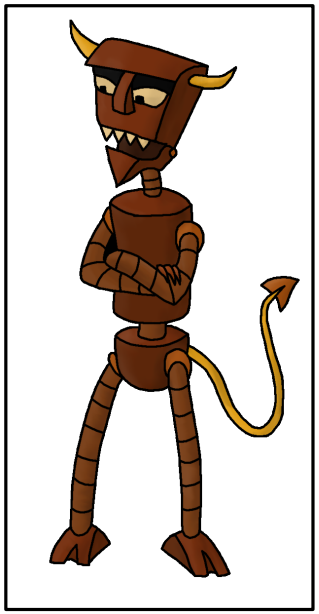
Reduction $\mathcal{D}$



m

t

m^*, t^*



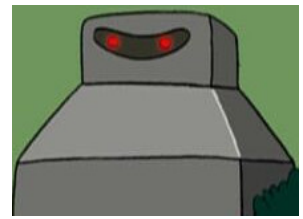
$\beta := 1$ iff $(t^* = t')$

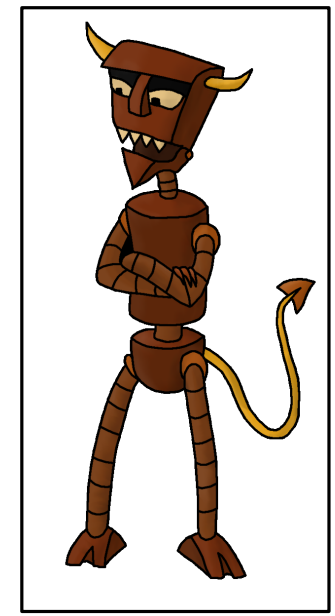
$\mathcal{A}$ p.p.t. $\Rightarrow$ $\mathcal{D}$ p.p.t.

A fixed-length MAC

Theorem 4.6. Π is a secure MAC

Proof: By reduction against security of PRF F





Reduction $\mathcal{D}$

m

$F_k(m)$

$t =$

m

t

m^*, t^*

Cases:

m^*

t'

β

$\beta := 1$ iff $(t^* = t')$

$\Leftrightarrow t^* = F_k(m^*)$

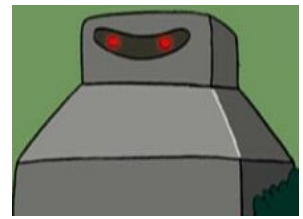
$\Leftrightarrow \mathsf{Vrfy}_k(m^*, t^*) = 1$

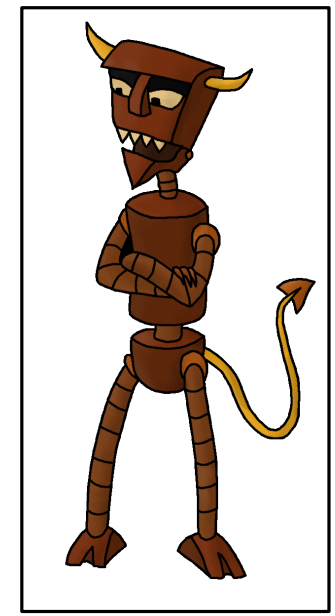
$$\Pr_k\left[\mathcal{D}^{F_k(\cdot)} = 1\right] = \Pr\left[\mathsf{MacForge}_{\mathcal{A},\Pi}(n) = 1\right]$$

A fixed-length MAC

Theorem 4.6. Π is a secure MAC

Proof: By reduction against security of PRF F





Reduction $\mathcal{D}$

m

$t = $

$f(m)$

m^*

t'

β

m

t

m^*, t^*

$\beta := 1$ iff $(t^* = t')$

Cases:

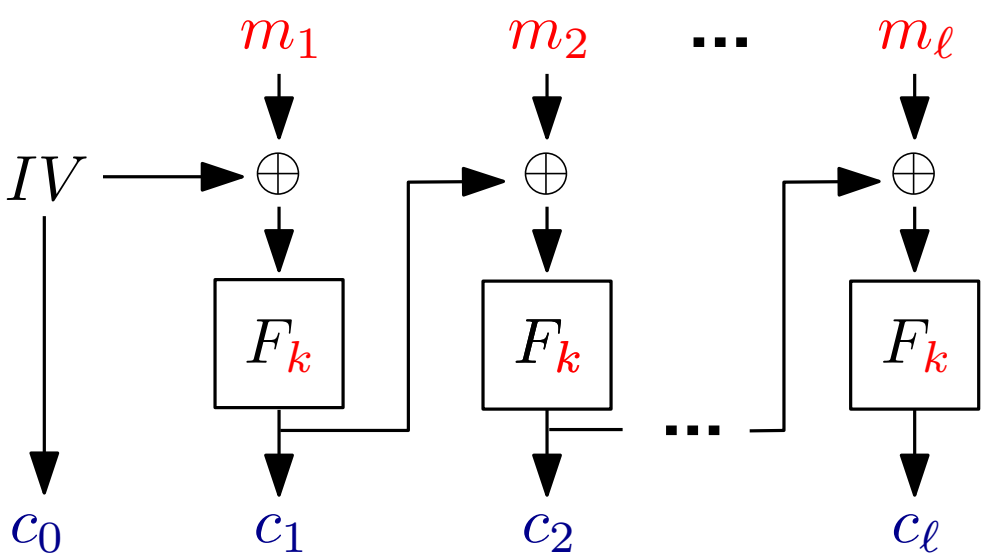
$$\Pr_f\left[\mathcal{D}^{f(\cdot)} = 1\right] = 2^{-n} \qquad (f(m^*) \text{ uniform and indep. of } t\text{'s})$$

$$\Rightarrow \Pr\left[\mathsf{MacForge}_{\mathcal{A},\Pi}(n) = 1\right] \leq 2^{-n} + \varepsilon(n) \qquad \square$$

CBC-MAC

- Until now: $\mathcal{M} = \{0,1\}^n$; e.g. AES: $n = 128$
- Goal: $\mathcal{M} = \{0,1\}^*$
- Idea: use chaining

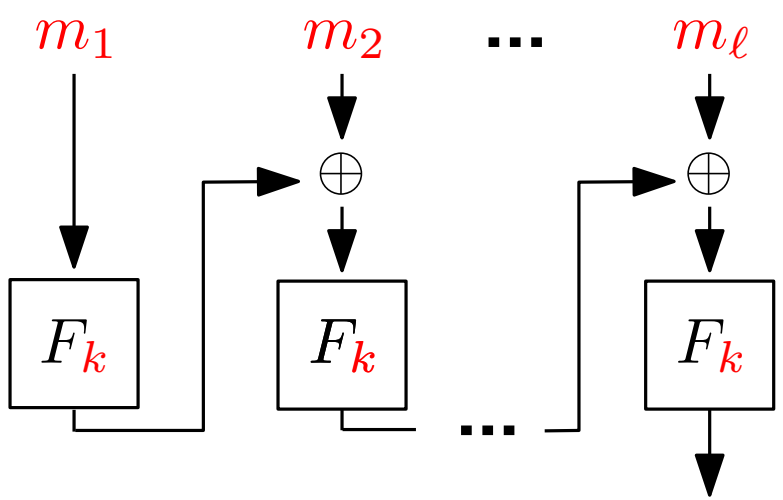
Recall CBC-mode encryption:



CBC-MAC

- Until now: $\mathcal{M} = \{0,1\}^n$; e.g. AES: $n = 128$
- Goal: $\mathcal{M} = \{0,1\}^*$

Basic **CBC-MAC:**



Verification? *Recompute!*

Differences to CBC-mode:

- no initialization vector (IV)
- only final value is output

Both changes crucial for security!

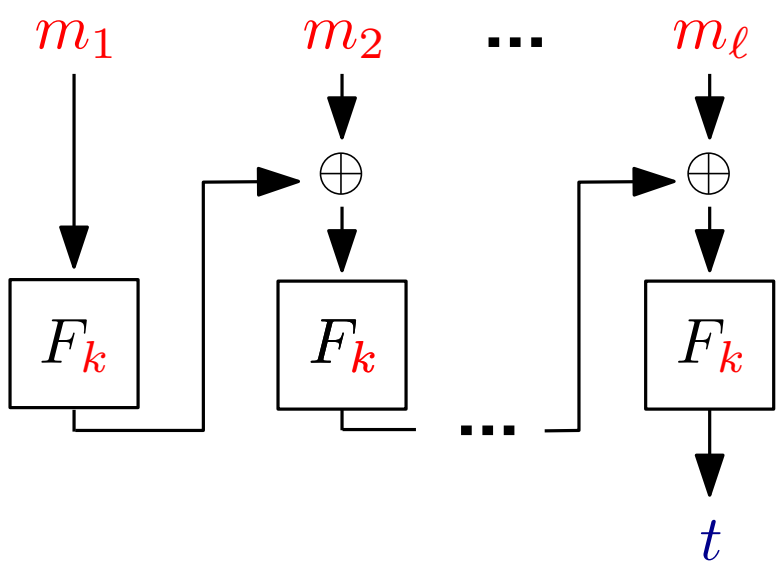
Theorem 4.10. Let ℓ be a polynomial. If F is a PRF, then the above is a secure MAC for $\mathcal{M} = \{0,1\}^{\ell(n)\cdot n}$

CBC-MAC

- Until now: $\mathcal{M} = \{0,1\}^n$; e.g. AES: $n = 128$
- Goal: $\mathcal{M} = \{0,1\}^*$

Basic **CBC-MAC:**

Not secure if we allowed variable length!



$\mathsf{Mac}_k(m) = t$

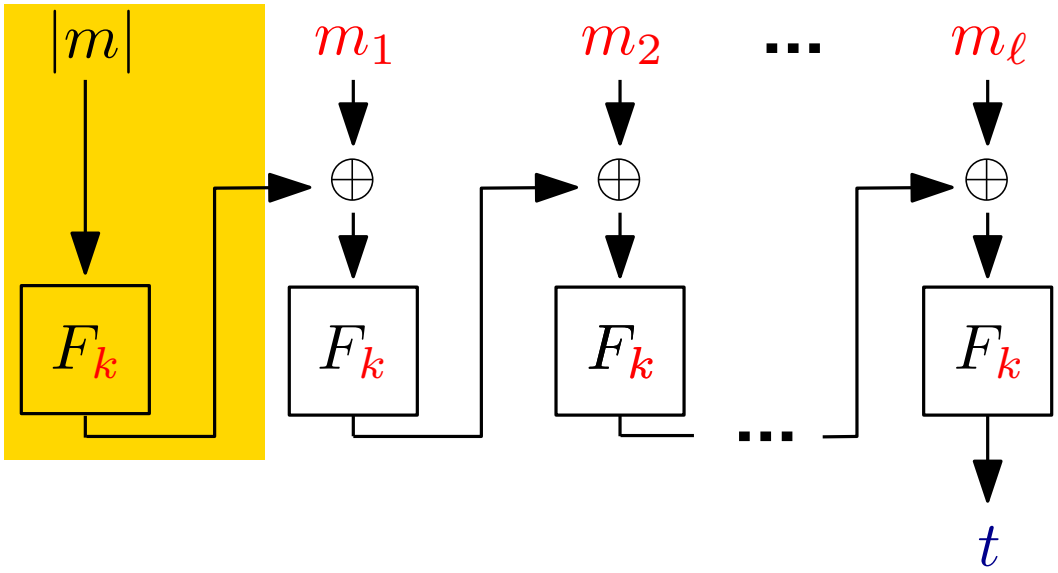
$\mathsf{Mac}_k(m \,\|\, m \oplus t) = ?$

$\Rightarrow$ Prepending message length makes it secure!

CBC-MAC

- Until now: $\mathcal{M} = \{0,1\}^n$; e.g. AES: $n = 128$
- Goal: $\mathcal{M} = \{0,1\}^*$

CBC-MAC:



Authenticated encryption

§5.2

Authenticated Encryption

Goal: **secure** communication: *secrecy* and *integrity*

- Alice and Bob share two (independent) keys k_E and k_M



c, t

$c \leftarrow \mathsf{Enc}_{k_E}(m)$
$t \leftarrow \mathsf{Mac}_{k_M}(m)$

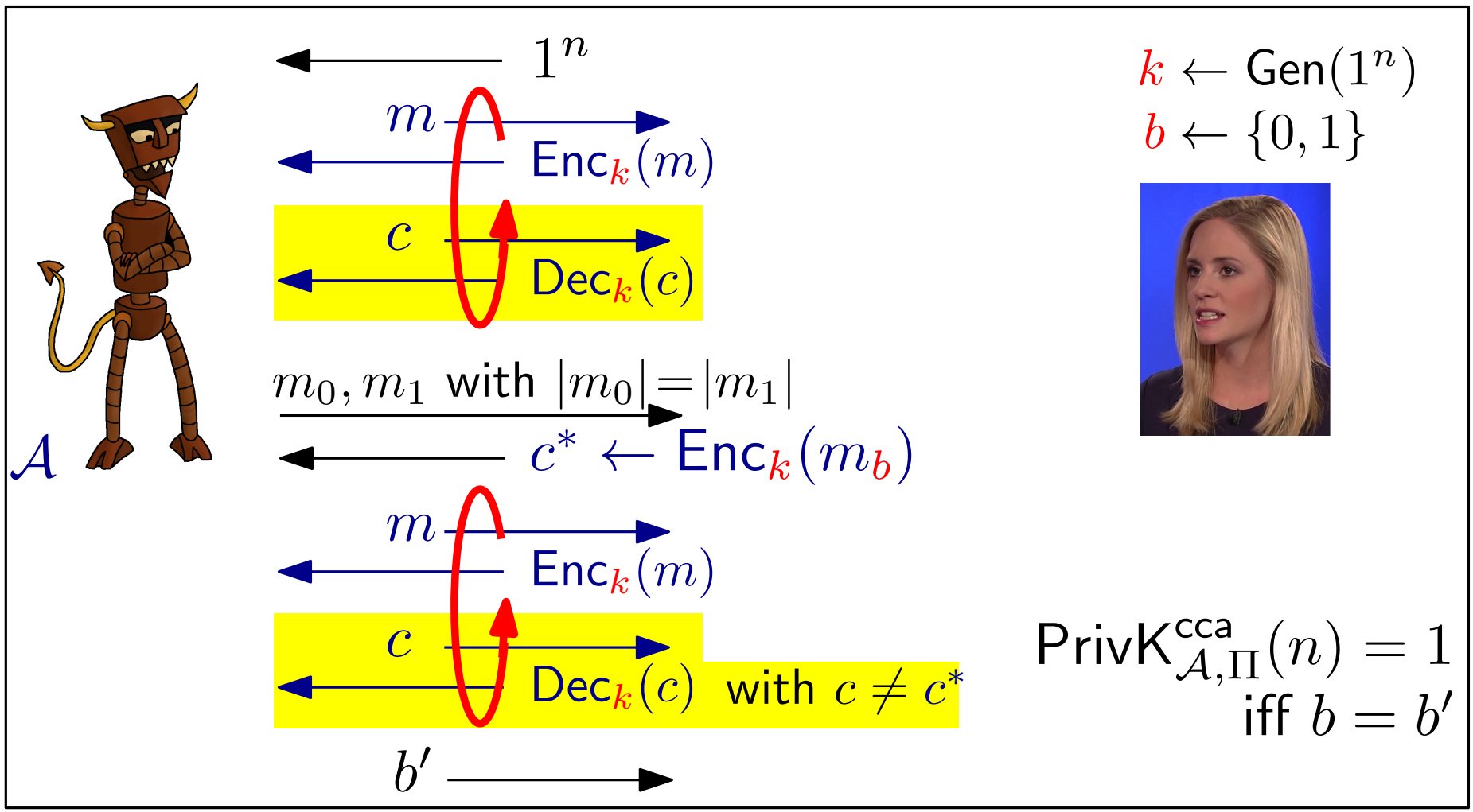
"Encrypt and authenticate"

$m = \mathsf{Dec}_{k_E}(c)$
$\mathsf{Vrfy}_{k_M}(m, t) \stackrel{?}{=} 1$

- t might reveal information about m!
- CBC-MAC $\Rightarrow$ not even CPA-secure!

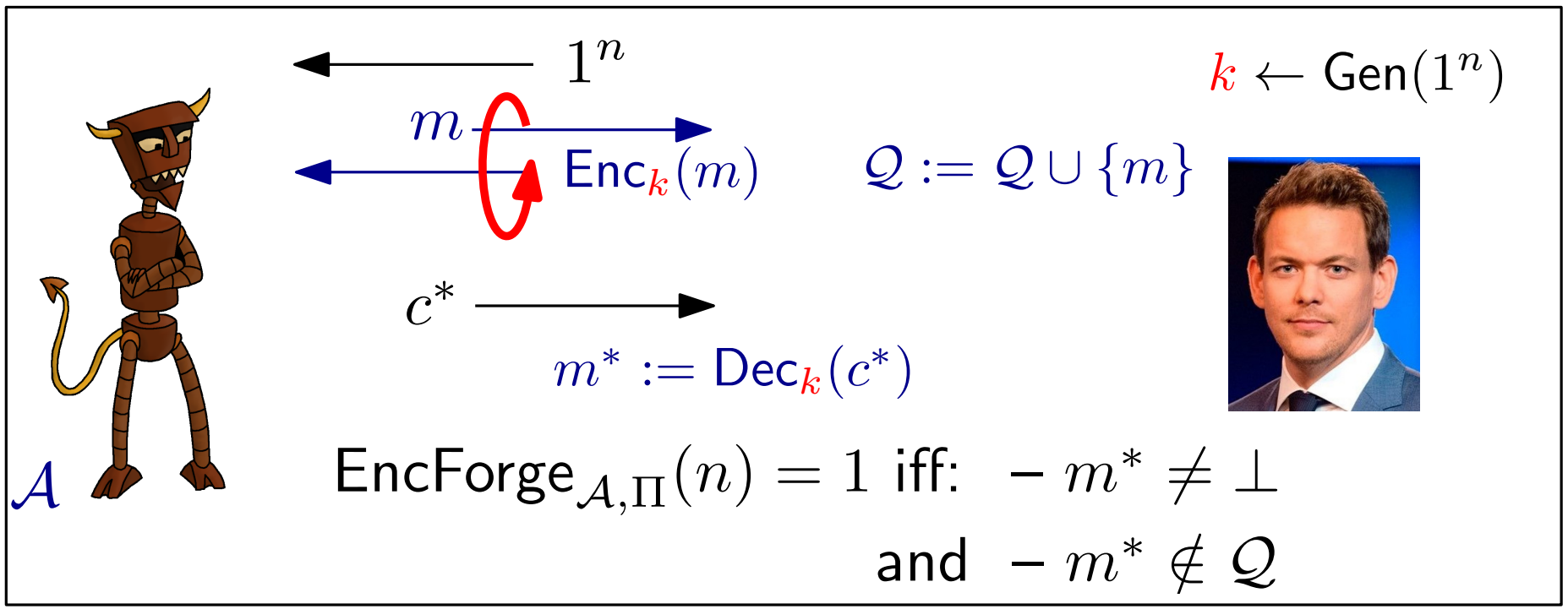
Authenticated Encryption

Definition 5.3. A private-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is an **authenticated encryption scheme** if it is CCA-secure and unforgeable



Authenticated Encryption

Definition 5.3. A private-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is an **authenticated encryption scheme** if it is CCA-secure and unforgeable



Definition 5.2. Π *is* **unforgeable** *if for every p.p.t.* $\mathcal{A}$ *there exists negligible* $\varepsilon(\cdot)$*:* $\Pr\left[\mathsf{EncForge}_{\mathcal{A},\Pi}(n) = 1\right] \leq \varepsilon(n)$

Authenticated Encryption

Goal: **secure** communication: *secrecy* and *integrity*

- Alice and Bob share two (independent) keys k_E and k_M



(c, t)

$\overline{\mathsf{Enc}}_{(k_E,k_M)}(m)$: $c \leftarrow \mathsf{Enc}_{k_E}(m)$
$t \leftarrow \mathsf{Mac}_{k_M}(c)$
return (c, t)

"Encrypt **then** authenticate"

$\overline{\mathsf{Dec}}_{(k_E,k_M)}(c, t)$:
If $\mathsf{Vrfy}_{k_M}(c, t) = 0$
return $\perp$
else $m := \mathsf{Dec}_{k_E}(c)$
return m

Authenticated Encryption

Security: If encryption scheme Π_E is CPA-secure and MAC Π_M is (deterministic and canonical and) secure, then **encrypt-then-authenticate** is:

- CPA-secure
- unforgeable

MAC security $\Rightarrow$ *adversary cannot create valid ciphertexts!*

$\Rightarrow$ decryption oracle useless for $\mathcal{A}$!

$\Rightarrow$ **We can even prove CCA-security!**

Theorem 5.7'. If Π_E is CPA-secure and Π_M is (deterministic and canonical and) secure, then **encrypt-then-authenticate is an authenticated encryption scheme**

Authenticated Encryption

Advanced AE:

- AE with *nonce*
- Single-key AE
- Lightweight AE
- Stronger security notions
  - E.g. Unverified-plaintext release

Will be covered in the Symmetric Cryptography course (192.124/2025S)

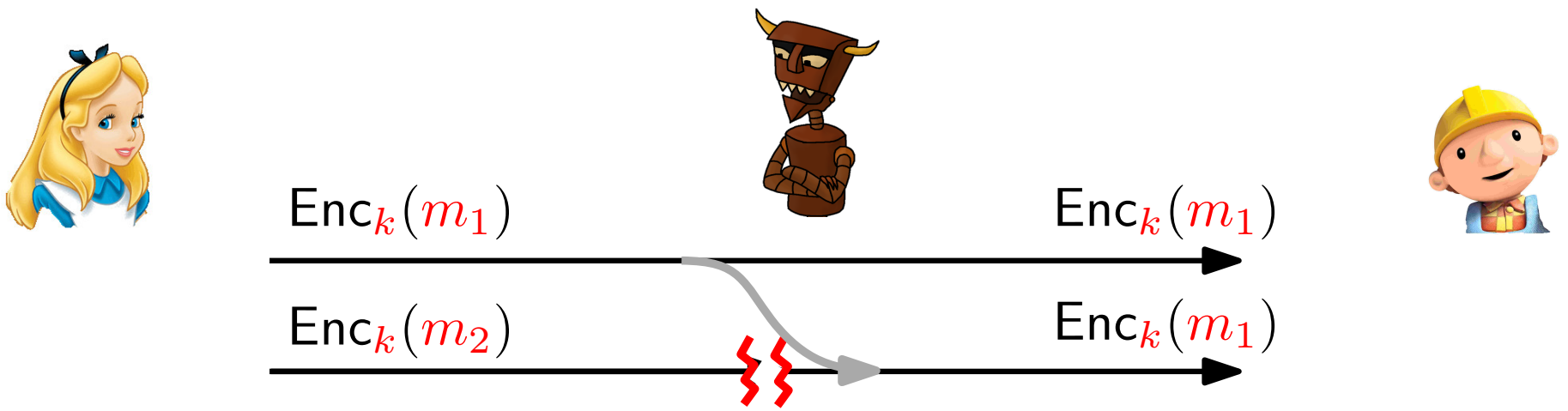
Secure communication sessions

§5.4

Secure sessions

Two parties want to communicate *securely* (secrecy and integrity) over a period of time in which they maintain *state*.

$\Rightarrow$ use authenticated encryption enough?

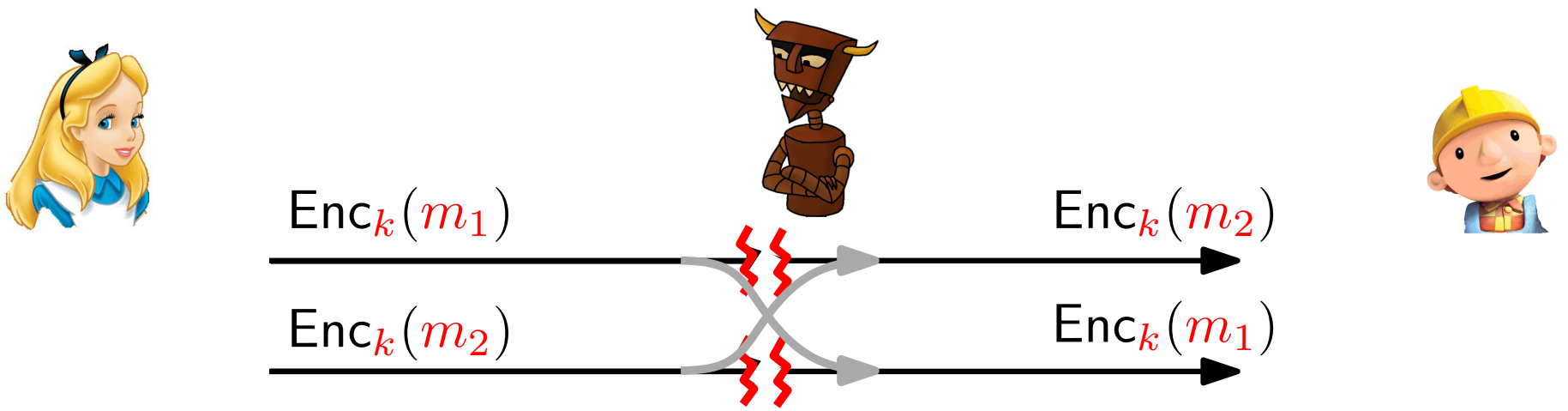


Replay attack

Secure sessions

Two parties want to communicate *securely* (secrecy and integrity) over a period of time in which they maintain *state*.

$\Rightarrow$ use authenticated encryption enough?

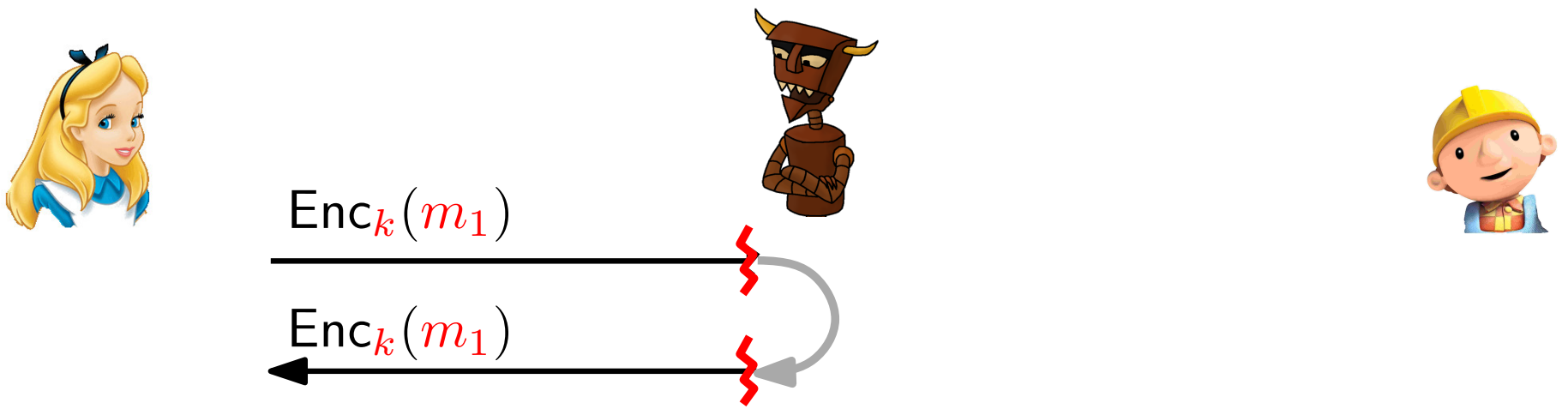


Re-ordering attack

Secure sessions

Two parties want to communicate *securely* (secrecy and integrity) over a period of time in which they maintain *state*.

$\Rightarrow$ use authenticated encryption enough?



Reflection attack

Secure sessions

Two parties want to communicate *securely* (secrecy and integrity) over a period of time in which they maintain *state*.

$\Rightarrow$ use authenticated encryption enough?

$\mathsf{Enc}_k(\text{“}A\text{”} \,\|\, m_1 \,\|\, 1)$

$\mathsf{Enc}_k(\text{“}A\text{”} \,\|\, m_2 \,\|\, 2)$

$\mathsf{Enc}_k(\text{“}B\text{”} \,\|\, m_3 \,\|\, 1)$

$\Rightarrow$ use **counters** and **identities**