Lecture 1
Introduction to Cryptography

Stefan Dziembowski
www.crypto.edu.pl/Dziembowski

University of Warsaw

5.10.16

version 1.0

Podstawowe informacje

- **Egzamin:** pisemny dwuczęściowy, kolokwium w trakcie semestru
- **Strona przedmiotu:** http://www.crypto.edu.pl/teaching/crypto-i-1
- **Podstawowy podręcznik:** Jonathan Katz and Yehuda Lindell Introduction to Modern Cryptography
- **Pozostała literatura**
  - Doug Stinson Cryptography Theory and Practice, Third Edition
  - Shafi Goldwasser and Mihir Bellare Lecture Notes on Cryptography
  - Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone Handbook of Applied Cryptography

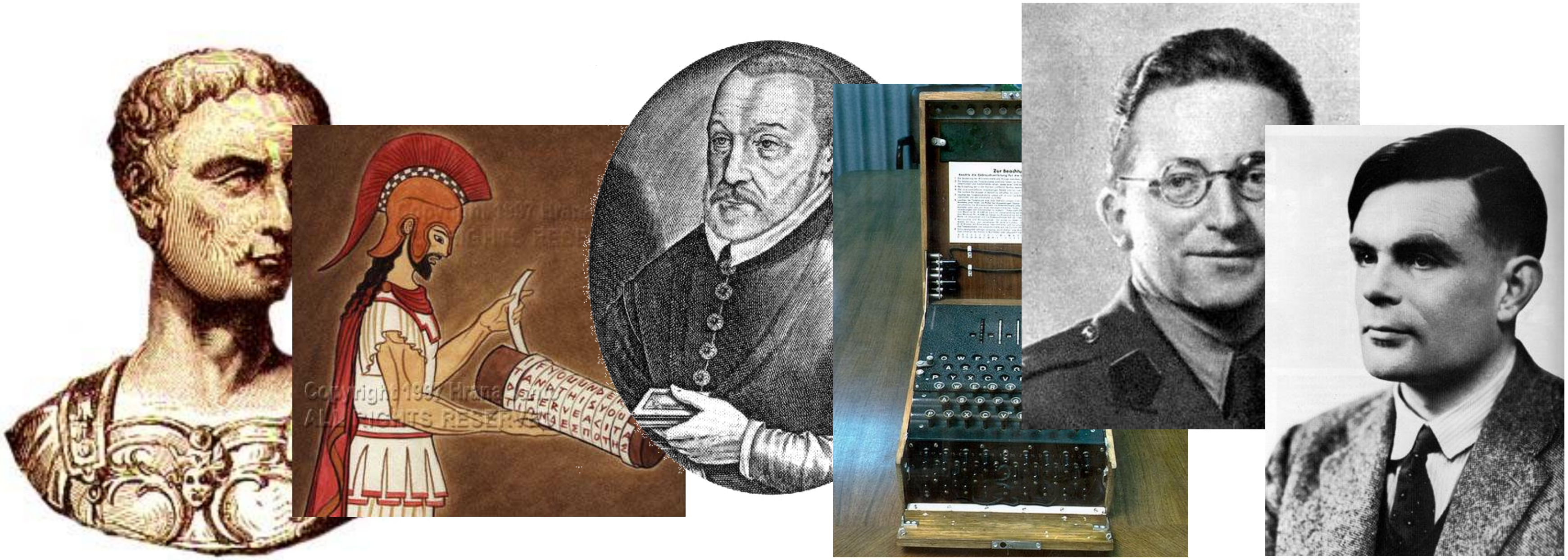
Plan

1. Introduction
2. Historical ciphers
3. Information-theoretic security
4. Computational security

Historical cryptography

cryptography ≈ encryption

main applications: **military and diplomacy**



Modern cryptography

cryptography = much more than encryption!

public-key cryptography

key agreement

signature schemes

mental poker

e-cash

zero-knowledge

electronic auctions

electronic voting

multiparty-computations

indistinguishability obfuscation

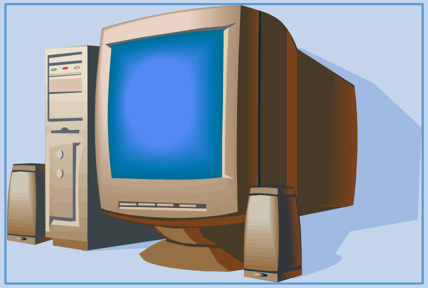
sevenites

now

What happened in the seventies?

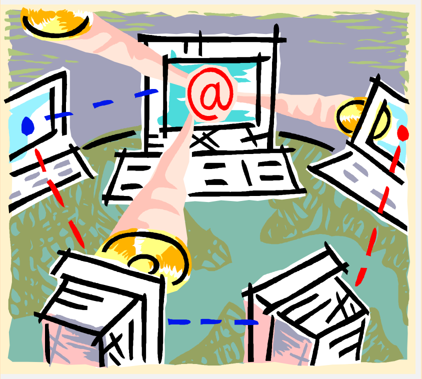
Technology

affordable hardware



Demand

companies and individuals start to do business electronically



Theory

the computational complexity theory is born

this allows researchers to reason about security in a **formal way**.

Cryptography

In the past:

the **art** of encrypting messages (mostly for the military applications).

Now:

the **science** of securing digital communication and transactions (encryption, authentication, digital signatures, e-cash, auctions, etc..)

Three components of the course

1. practical apects

2. mathematical foundations

3. new horizons

Practical aspects

- **symmetric encryption**: block ciphers (DES, AES) and tream ciphers (RC4)
- **hash functions** (MD5, SHA1,...), message authentication (CBC-MAC)
- **public-key infrastructure** (X.509, PGP, identity-based)
- **elements of number theory**
- **asymetric encrypion** (RSA, ElGamal, Rabin,...)
- **signature schemes** (RSA, ElGamal,...)

Mathematical foundations

- What makes us believe that the **protocols are secure**?
- Can we formally **define** “security”?
- Can security be **proven**?
- Do there exist “**unbreakable**” ciphers?

New horizons

Advanced cryptographic protocols, such as:

- **zero-knowledge**
- **multiparty computations**
- **private information retrieval**

This course is **not** about

- **practical data security** (firewalls, intrusion-detection, VPNs, etc.),

  (**however**, we will talk a bit about the **cryptographic protocols** used in real life)

- **history** of cryptography,

- **number theory** and **algebra**

  (we will use them **only as tools**)

- **complexity theory.**

Terminology

constructing secure systems

breaking the systems

Cryptology = cryptography + cryptanalysis

This convention is **slightly artificial** and often ignored.

Common usage:

“cryptanalysis of X” = “breaking X”

Common abbreviation: **“crypto”**

Cryptography – general picture

plan of the course:

	encryption	**authentication**
private key	**1** private key encryption	**2** private key authentication
public key	**3** public key encryption	**4** signatures

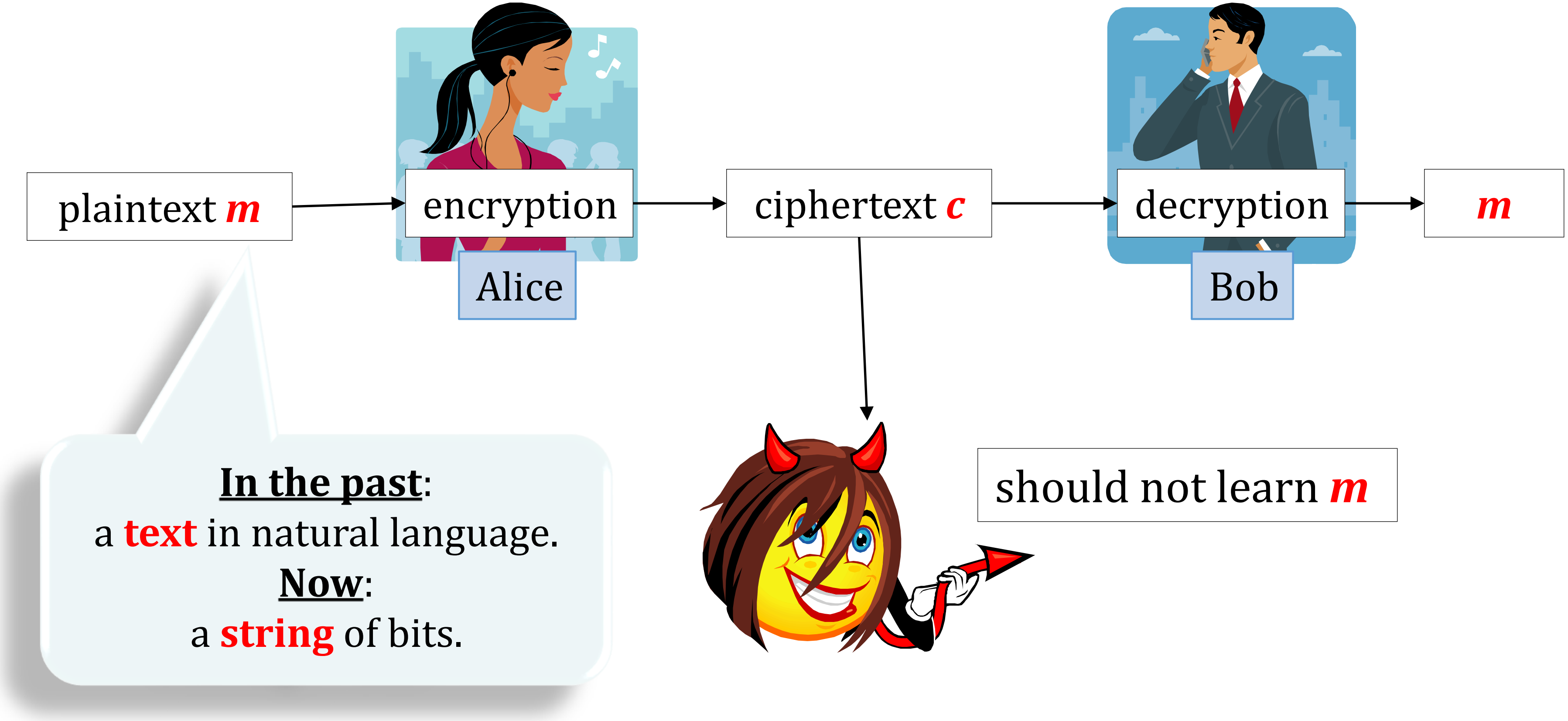
5 advanced cryptographic protocols

Preliminary plan of the lectures

1. Introduction to Cryptography
2. Symmetric Encryption I
3. Symmetric Encryption II
4. Symmetric Encryption III
5. Message Authentication and Introduction to Hash Functions
6. Key Management and Public-Key Cryptography
7. A Brush-up on Number. Theory and Algebra
8. Public-Key Encryption I
9. Public-Key Encryption II
10. Signature Schemes and Commitment Schemes
11. Commitment Schemes and Zero-Knowledge Protocols
12. Two-party and Multi-party Computation Protocols
13. Private Information Retrieval
14. Introduction to Cryptographic Currencies

Encryption schemes
(a very general picture)

Encryption scheme (cipher) = encryption & decryption



plaintext ***m***

encryption

Alice

ciphertext ***c***

decryption

Bob

m

should not learn ***m***

<u>In the past</u>:
a **text** in natural language.
<u>Now</u>:
a **string** of bits.

Art vs. science

In the past:

lack of precise definitions, ad-hoc design, usually insecure.

Nowadays:

formal definitions, systematic design, very secure constructions.

Provable security

We want to construct schemes that are **provably secure**.

But...

- **why** do we want to do it?
- **how** to define it?
- and is it **possible** to achieve it?

Provable security – the motivation

In many areas of computer science formal proofs are **not essential**.

For example, instead of proving that an algorithm is efficient, we can just simulate it on a "***typical*** input".

In **cryptography** it's **not true**, because

there cannot exist an experimental proof that a scheme is secure.

Why?

Because a notion of a "*typical* adversary" does not make sense.

Security definitions are useful also because they allow us to construct schemes in a modular way...

Kerckhoffs' principle

Auguste Kerckhoffs (1883):

The enemy knows the system

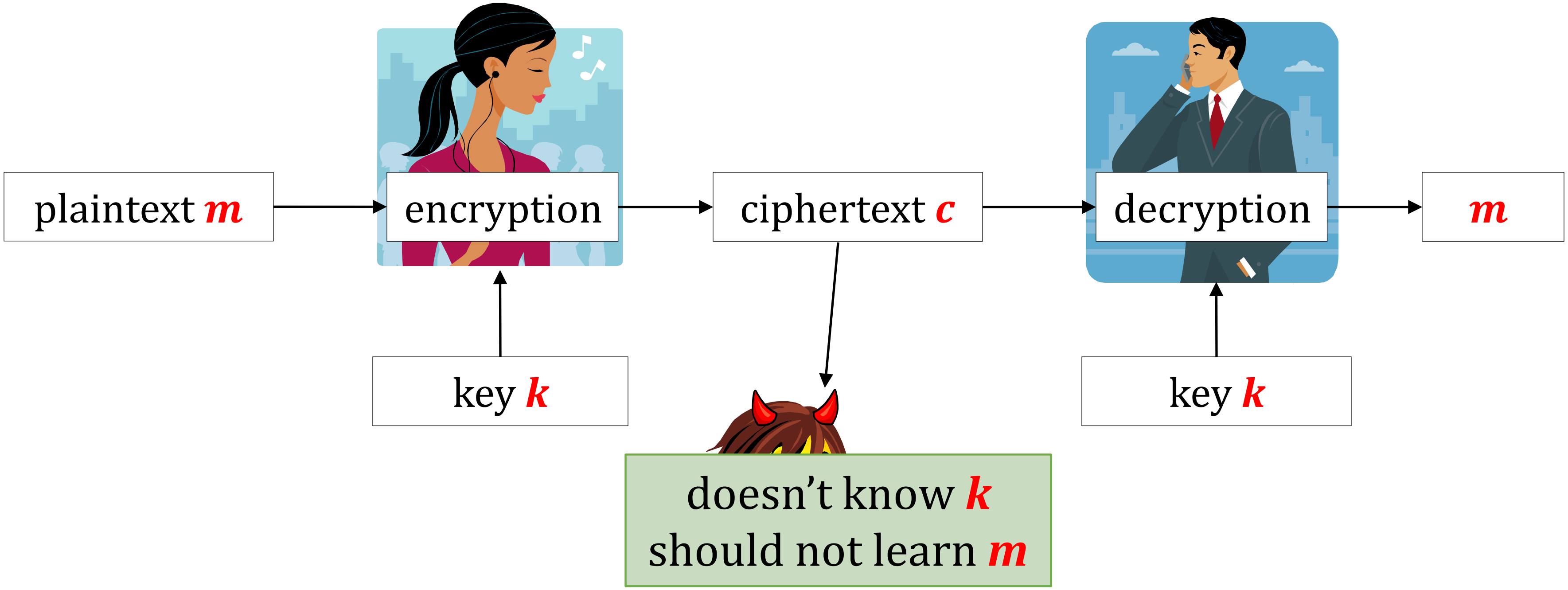
The cipher should remain secure even if **the adversary knows the specification of the cipher.**

The only thing that is **secret** is a

short key ***k***

that is **usually chosen uniformly at random**

A more refined picture



(Of course Bob can use the same method to send messages to Alice.)
(That's why it's called the **symmetric setting**)

Let us assume that k is unifromly random

Kerckhoffs' principle – the motivation

1. In commercial products it is unrealistic to assume that the design details remain secret (**reverse-engineering!**)
2. Short keys are easier to **protect**, **generate** and **replaced**.
3. The design details can be discussed and **analyzed in public**.

Not respecting this principle

=

``**security by obscurity**”.

A mathematical view

$\mathcal{K}$ – **key** space

$\mathcal{M}$ – **plaintext** space

$\mathcal{C}$ - **ciphertext** space

An **encryption scheme** is a pair **(Enc,Dec)**, where

- **Enc** : $\mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$ is an **encryption** algorithm,
- **Dec** : $\mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$ is an **decryption** algorithm.

We will sometimes write $\mathbf{Enc}_k(m)$ and $\mathbf{Dec}_k(c)$ instead of $\mathbf{Enc}(k,m)$ and $\mathbf{Dec}(k,c)$.

Correctness

for every k we should have $\mathbf{Dec}_k(\mathbf{Enc}_k(m)) = m$.

Plan

1. Introduction
2. Historical ciphers
3. Information-theoretic security
4. Computational security

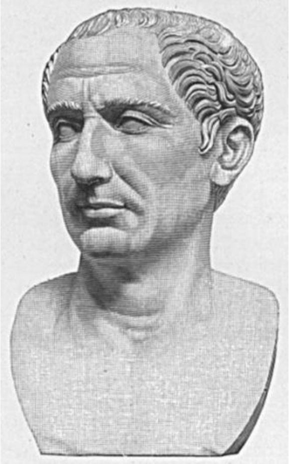
Shift cipher

$\mathcal{M}$ = **words over alphabet** $\{A,\dots,Z\} \approx \{0,\dots,25\}$

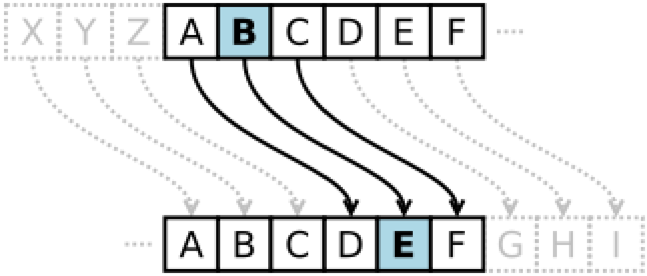
$\mathcal{K} = \{0,\dots,25\}$

$$\mathrm{Enc}_k(m_0,\dots,m_n) = (m_0 + k \bmod 26, \dots, m_n + \mathrm{k} \bmod 26)$$

$$\mathrm{Dec}_k(c_0,\dots,c_n) = (c_0 - \mathrm{k} \bmod 26, \dots, c_n - \mathrm{k} \bmod 26)$$



Cesar: $k = 3$



Security of the shift cipher

How to break the shift cipher?

Check all possible keys!

Let $\boldsymbol{c}$ be a ciphertext.

For every $\boldsymbol{k} \in \{\mathbf{0}, \ldots, \mathbf{25}\}$ check if $\mathbf{Dec}_{k}(\boldsymbol{c})$ "makes sense".

Most probably only one such $\boldsymbol{k}$ exists.

Thus $\mathbf{Dec}_{k}(\boldsymbol{c})$ is the message.

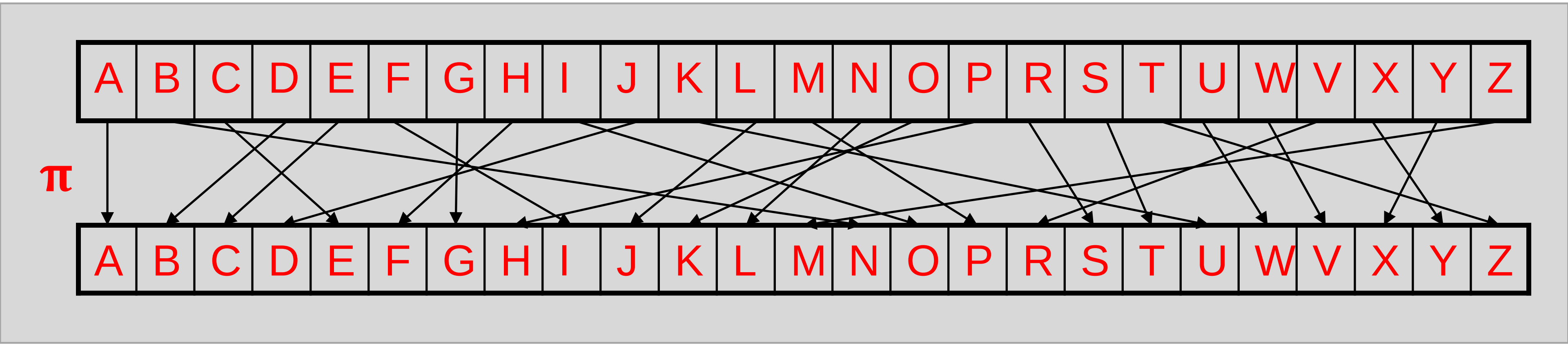
This is called a **brute force attack**.

Moral: the key space needs to be large!

Substitution cipher

$\mathcal{M}$ = **words over alphabet {A,…,Z} ≈ {0,…,25}**

$\mathcal{K}$ = **a set of permutations of {0,…,25}**


A B C D E F G H I J K L M N O P R S T U W V X Y Z
π
A B C D E F G H I J K L M N O P R S T U W V X Y Z

$$\mathbf{Enc}_{\pi}(m_0,\ldots,m_n) = (\pi(m_0),\ldots,\ \pi(m_n))$$

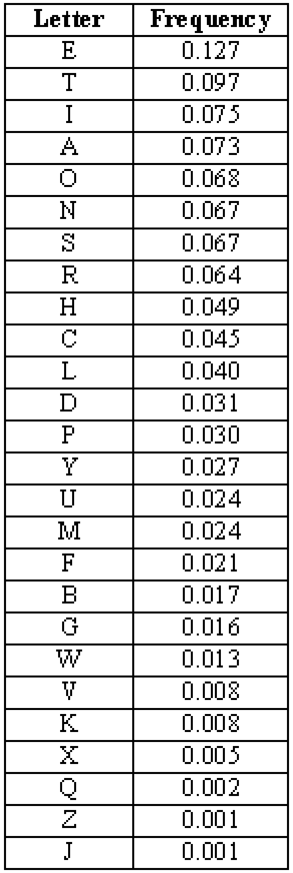
$$\mathbf{Dec}_{\pi}(c_0,\ldots,c_n) = (\pi^{-1}(c_0),\ldots,\ \pi^{-1}(c_n))$$

How to break the substitution cipher?

Use **statistical patterns** of the language.

For example: the **frequency tables**.

Texts of **50** characters can usually be broken this way.



Letter	Frequency
E	0.127
T	0.097
I	0.075
A	0.073
O	0.068
N	0.067
S	0.067
R	0.064
H	0.049
C	0.045
L	0.040
D	0.031
P	0.030
Y	0.027
U	0.024
M	0.024
F	0.021
B	0.017
G	0.016
W	0.013
V	0.008
K	0.008
X	0.005
Q	0.002
Z	0.001
J	0.001

Figure 7 - Frequency Table

Other famous historical ciphers

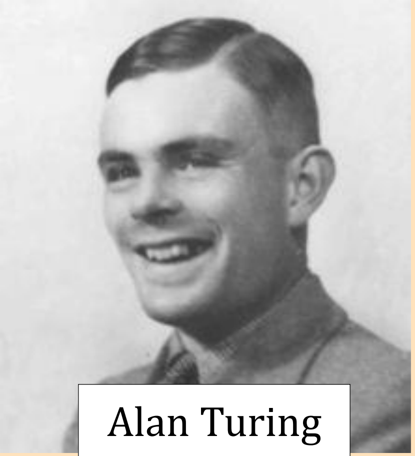
Vigenère cipher:

Blaise de Vigenère
(1523 - 1596)

Leon Battista Alberti
(1404 – 1472)

Enigma

Marian Rejewski
(1905 - 1980)



Alan Turing
(1912-1954)

In the past ciphers were designed in an ad-hoc manner

In contemporary cryptography the ciphers are designed in a **systematic way**.

Main goals:

1. define security
2. construct schemes that are "provably secure"

Plan

1. Introduction
2. Historical ciphers
3. Information-theoretic security
4. Computational security

Defining “security of an encryption scheme” is not trivial.

consider the following experiment

(m – a message)

1. the key K is chosen uniformly at random
2. $C := \mathbf{Enc}_K(m)$ is given to the adversary

how to define
security

?

Idea 1

(m – a message)

1. the key K is chosen uniformly at random
2. $C := \mathrm{Enc}_K(m)$ is given to the adversary

An idea

"The adversary should not be able to compute K."

A problem

the encryption scheme that "doesn't encrypt":

$$\mathrm{Enc}_K(m) = m$$

satisfies this definition!

Idea 2

(m – a message)

1. the key K is chosen uniformly at random
2. $C := \mathbf{Enc}_K(m)$ is given to the adversary

An idea

"The adversary should not be able to compute m."

A problem

What if the adversary can compute, e.g., the first half of m?

m_1	…	$m_{\|m\|/2}$	?	…	?

Idea 3

(*m* – a message)

1. the key *K* is chosen uniformly at random
2. $C := \mathrm{Enc}_K(m)$ is given to the adversary

An idea

"The adversary should not learn any information about *m*."

A problem

But he may already have some a priori information about *m*!

For example he may know that *m* is a sentence in English...

Idea 4

(m – a message)

1. the key K is chosen uniformly at random
2. $C := \mathrm{Enc}_K(m)$ is given to the adversary

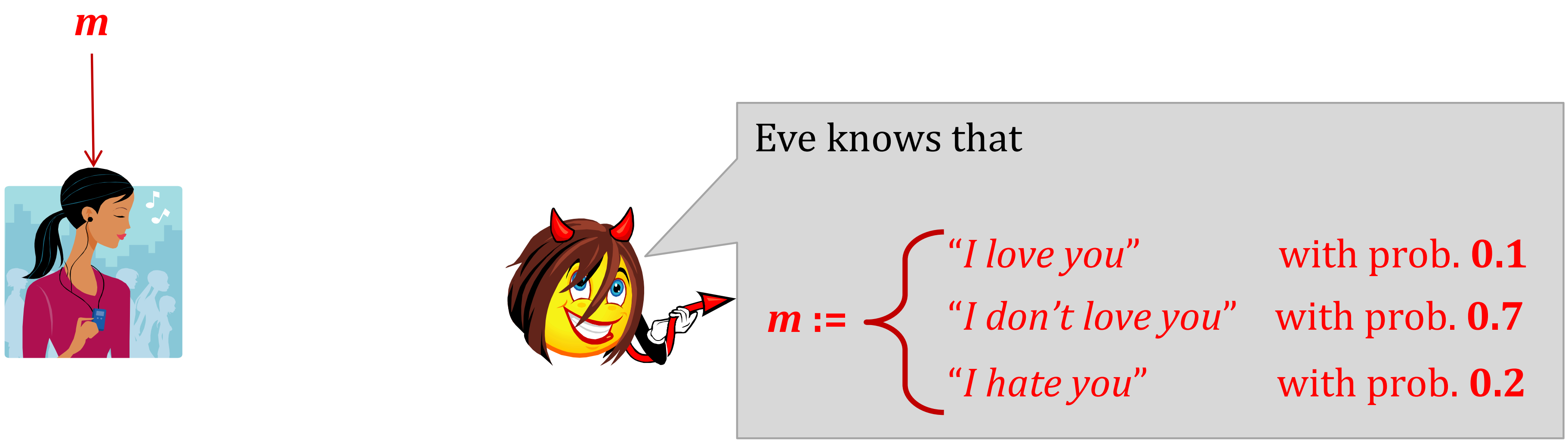
An idea

"The adversary should not learn any **additional** information about m."

This makes much more sense.

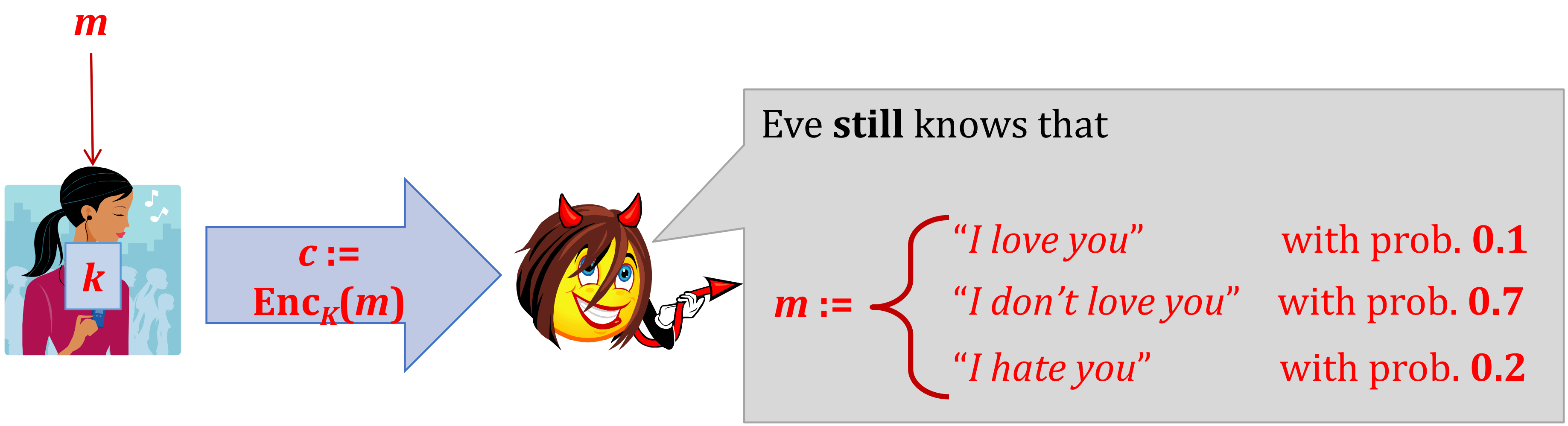
But how to formalize it?

Example



Eve knows that

$$m := \begin{cases} \textit{“I love you”} & \text{with prob. } \mathbf{0.1} \\ \textit{“I don't love you”} & \text{with prob. } \mathbf{0.7} \\ \textit{“I hate you”} & \text{with prob. } \mathbf{0.2} \end{cases}$$



Eve **still** knows that

$$m := \begin{cases} \textit{“I love you”} & \text{with prob. } \mathbf{0.1} \\ \textit{“I don't love you”} & \text{with prob. } \mathbf{0.7} \\ \textit{“I hate you”} & \text{with prob. } \mathbf{0.2} \end{cases}$$

How to formalize the "Idea 4"?

"The adversary should not learn any **additional** information about $\boldsymbol{m}$."

also called: **information-theoretically** secret

An encryption scheme is **perfectly secret** if

for every random variable $\boldsymbol{M}$

and every $\boldsymbol{m} \in \mathcal{M}$ and $\boldsymbol{c} \in \mathcal{C}$

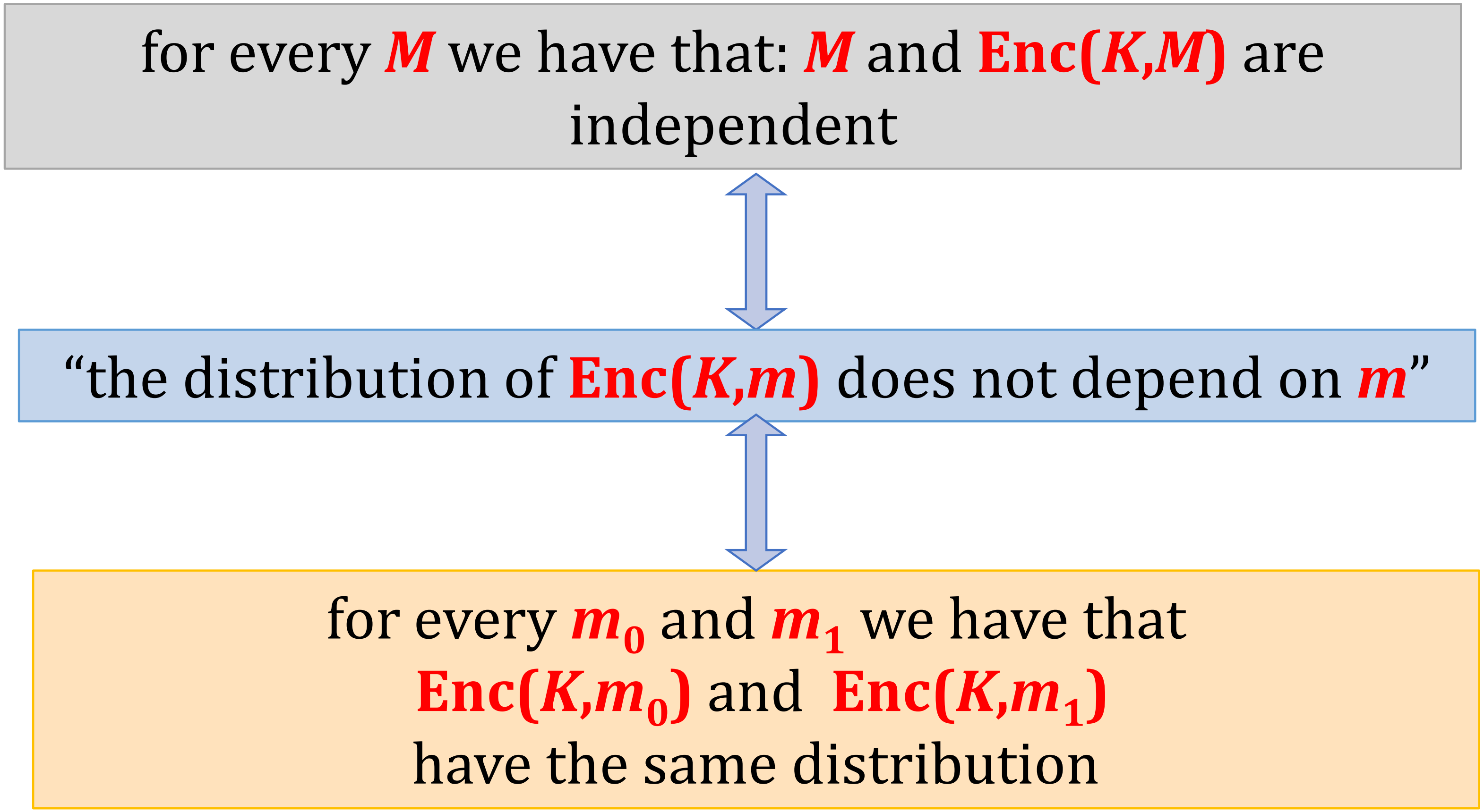
such that $\mathbf{P}(C = c) > 0$

$$\mathbf{P}(M = m) = \mathbf{P}(M = m \mid (\mathbf{Enc}(K,M)) = c)$$

⇕

equivalently: M and $\mathbf{Enc}(K,M)$ are independent

Equivalently:



for every M we have that: M and $\mathbf{Enc}(K,M)$ are independent

“the distribution of $\mathbf{Enc}(K,m)$ does not depend on m”

for every m_0 and m_1 we have that
$\mathbf{Enc}(K,m_0)$ and $\mathbf{Enc}(K,m_1)$
have the same distribution

A perfectly secret scheme: one-time pad

t – a parameter

$\mathcal{K} = \mathcal{M} = \{0,1\}^t$

component-wise **xor**

Gilbert Vernam

(1890 –1960)

Vernam's cipher:

$$\mathrm{Enc}_k(m) = k \text{ xor } m$$

$$\mathrm{Dec}_k(c) = k \text{ xor } c$$

Correctness is trivial:

$$\mathrm{Dec}_k(\mathrm{Enc}_k(m)) = k \text{ xor } (k \text{ xor } m)$$

$$m$$

Perfect secrecy of the one-time pad

Perfect secrecy of the one time pad is also trivial.

This is because for every m
the distribution of **Enc(K,m)** is uniform
(and hence does not depend on m).

for every c:
$\mathrm{P}(\mathrm{Enc}(K,m) = c) = \mathrm{P}(K = m \text{ xor } c) = 2^{-t}$

Observation

One time pad can be **generalized** as follows.

Let **$(G,+)$** be a group. Let $\mathcal{K} = \mathcal{M} = \mathcal{C} = G$.

The following is a perfectly secret encryption scheme:

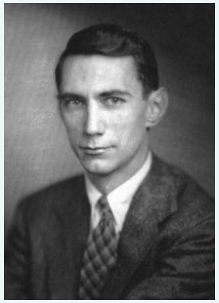
- **$\mathrm{Enc}(k,m) = m + k$**
- **$\mathrm{Dec}(k,m) = m - k$**

Why the one-time pad is not practical?

1. **The key has to be as long as the message.**
2. **The key cannot be reused**

This is because:

$$\begin{aligned}\mathbf{Enc}_k(m_0) \text{ xor } \mathbf{Enc}_k(m_1) &= (k \text{ xor } m_0) \text{ xor } (k \text{ xor } m_1) \\ &= m_0 \text{ xor } m_1\end{aligned}$$



Theorem (Shannon 1949)

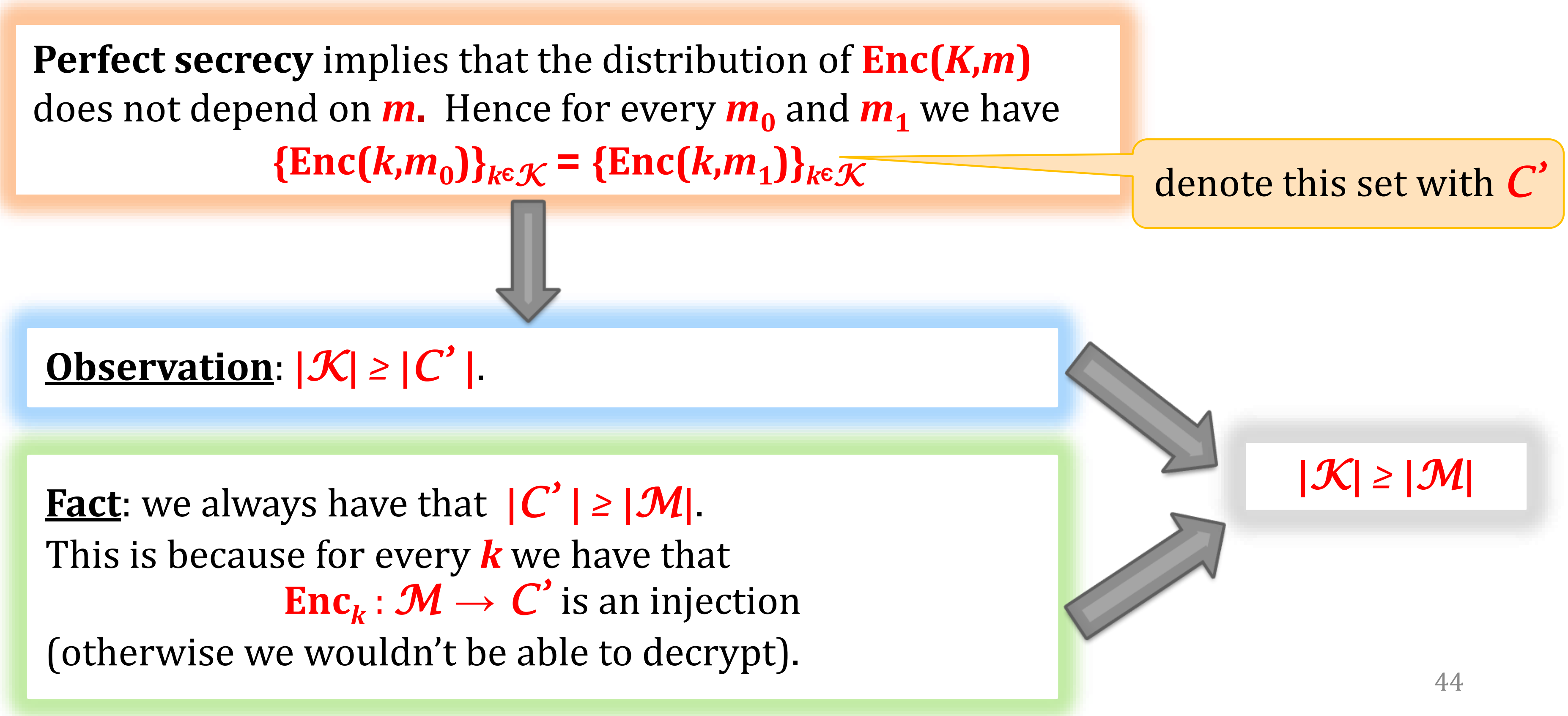
("One time-pad is optimal in the class of perfectly secret schemes")

In every perfectly secret encryption scheme

$$\mathbf{Enc} : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}, \ \mathbf{Dec} : \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$$

we have $|\mathcal{K}| \geq |\mathcal{M}|$.

Proof



Perfect secrecy implies that the distribution of **Enc(*K*,*m*)** does not depend on ***m***. Hence for every m_0 and m_1 we have

$$\{\mathbf{Enc}(k,m_0)\}_{k \in \mathcal{K}} = \{\mathbf{Enc}(k,m_1)\}_{k \in \mathcal{K}}$$

denote this set with $\mathcal{C}'$

Observation: $|\mathcal{K}| \geq |\mathcal{C}'|$.

Fact: we always have that $|\mathcal{C}'| \geq |\mathcal{M}|$.

This is because for every ***k*** we have that

$\mathbf{Enc}_k : \mathcal{M} \rightarrow \mathcal{C}'$ is an injection

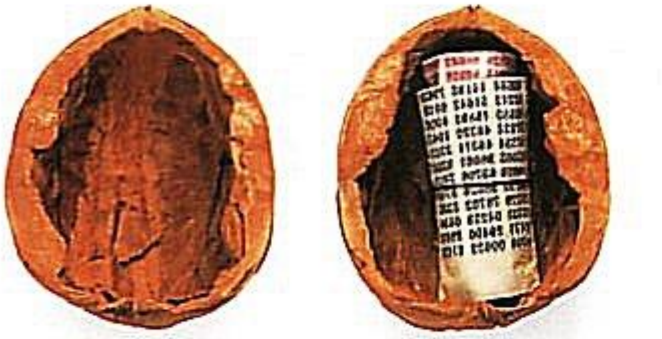
(otherwise we wouldn't be able to decrypt).

$|\mathcal{K}| \geq |\mathcal{M}|$

Practicality?

Generally, the **one-time pad** is **not very practical**, since:

- the key has to be as long as the **total** length of the encrypted messages,
- it is hard to generate truly random strings.



a **KGB** one-time pad hidden in a walnut shell

However, it is sometimes used (e.g. in the **military applications**), because of the following advantages:

- **perfect secrecy**,
- short messages can be encrypted using **pencil and paper** .

In the 1960s the Americans and the Soviets established a hotline that was encrypted using the one-time pad.(**additional advantage**: they didn't need to share their secret encryption methods)

Venona project (1946 – 1980)

Ethel and Julius Rosenberg

American **National Security Agency** decrypted **Soviet** messages that were transmitted in the 1940s.

That was possible because the Soviets reused the keys in the one-time pad scheme.

Outlook

We constructed a perfectly secret encryption scheme

Our scheme has certain drawbacks $(|\mathcal{K}| \geq |\mathcal{M}|)$.

But by Shannon's theorem this is unavoidable.

Can we go home and relax?

maybe the secrecy definition is too strong?

What to do?

Idea

use a model where the **power** of the adversary is limited.

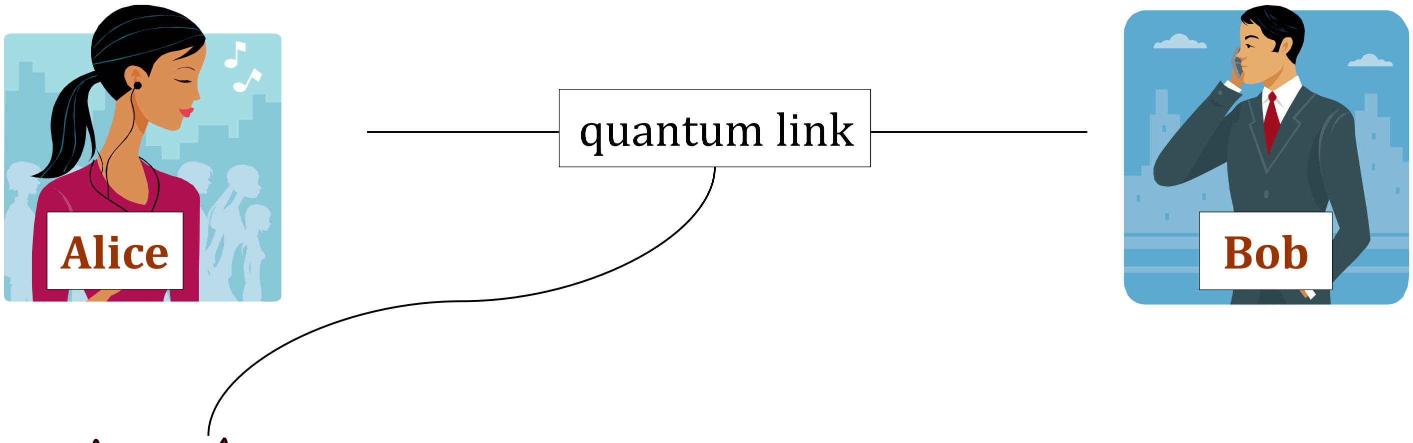
How?

Classical (computationally-secure) cryptography:
bound his computational power.

Alternative options:
quantum cryptography, bounded-storage model,...
(not too practical)

Quantum cryptography

Stephen Wiesner (1970s), Charles H. Bennett and Gilles Brassard (1984)



Alice

quantum link

Bob

Eve

Quantum indeterminacy: quantum states **cannot** be measured without disturbing the original state.

Hence **Eve** cannot read the bits in an unnoticeable way.

Quantum cryptography

Advantage:

security is based on the laws of quantum physics

Disadvantage:

needs a dedicated equipment.

Practicality?

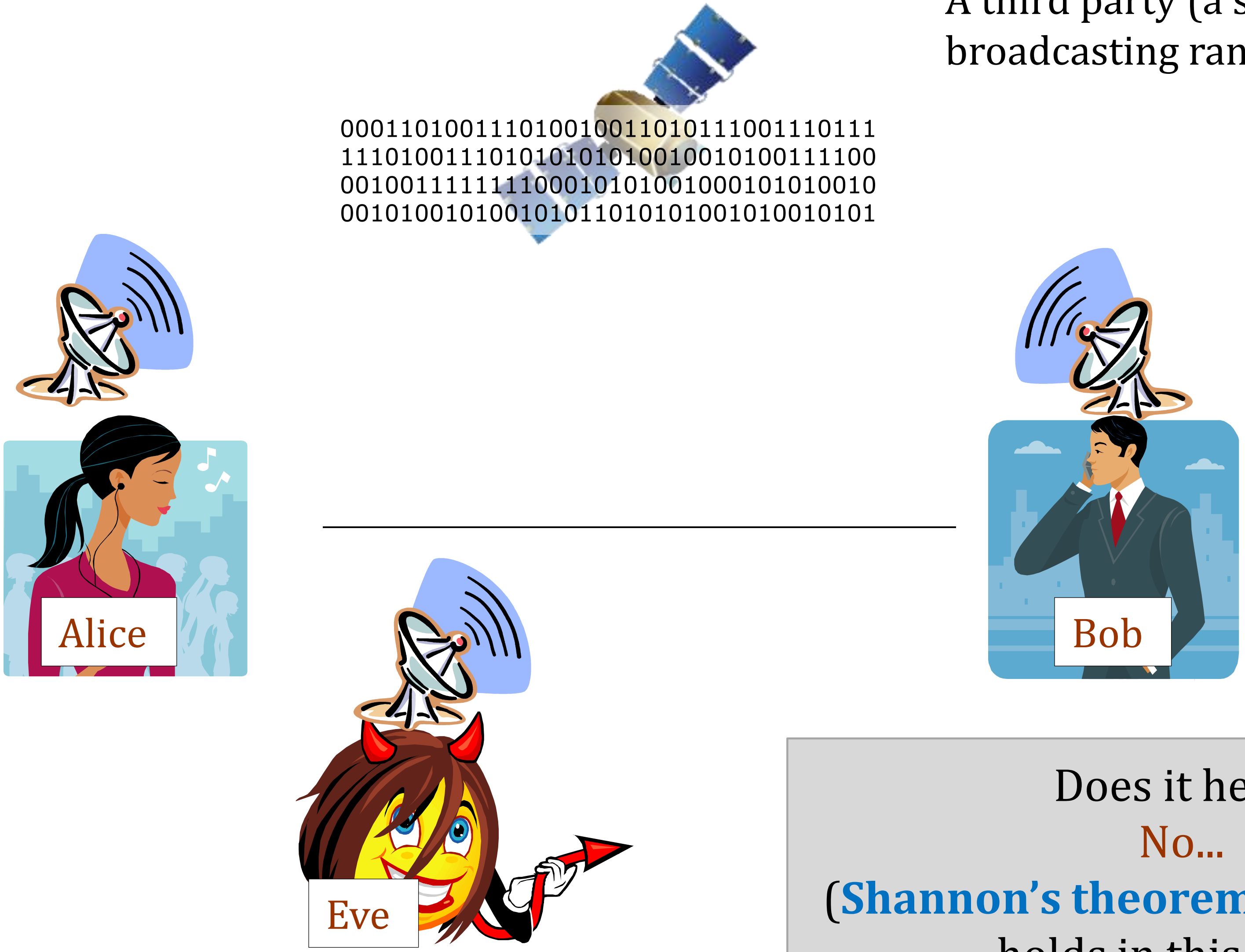
Currently: successful transmissions for distances of length around **150 km**.

Commercial products are available.

Warning:
Quantum **cryptography** should not be confused with quantum **computing**.

A satellite scenario

A third party (a satellite) is broadcasting random bits.



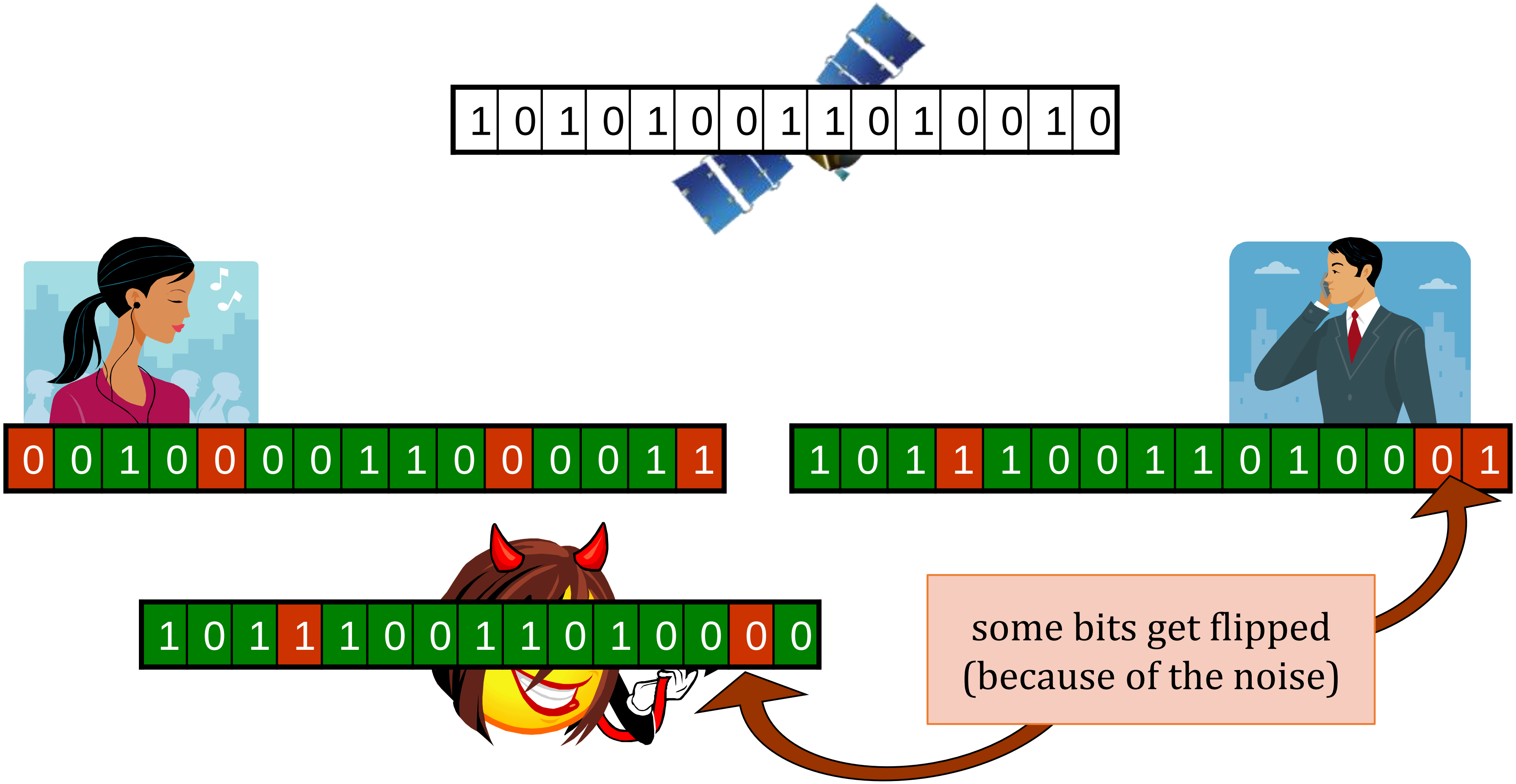
000110100111010010011010111001110111
111010011101010101010010010100111100
001001111111100010101001000101010010
001010010100101011010101001010010101

Does it help?

No...

(**Shannon's theorem** of course also holds in this case.)

Ueli Maurer (1993): noisy channel.



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

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

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

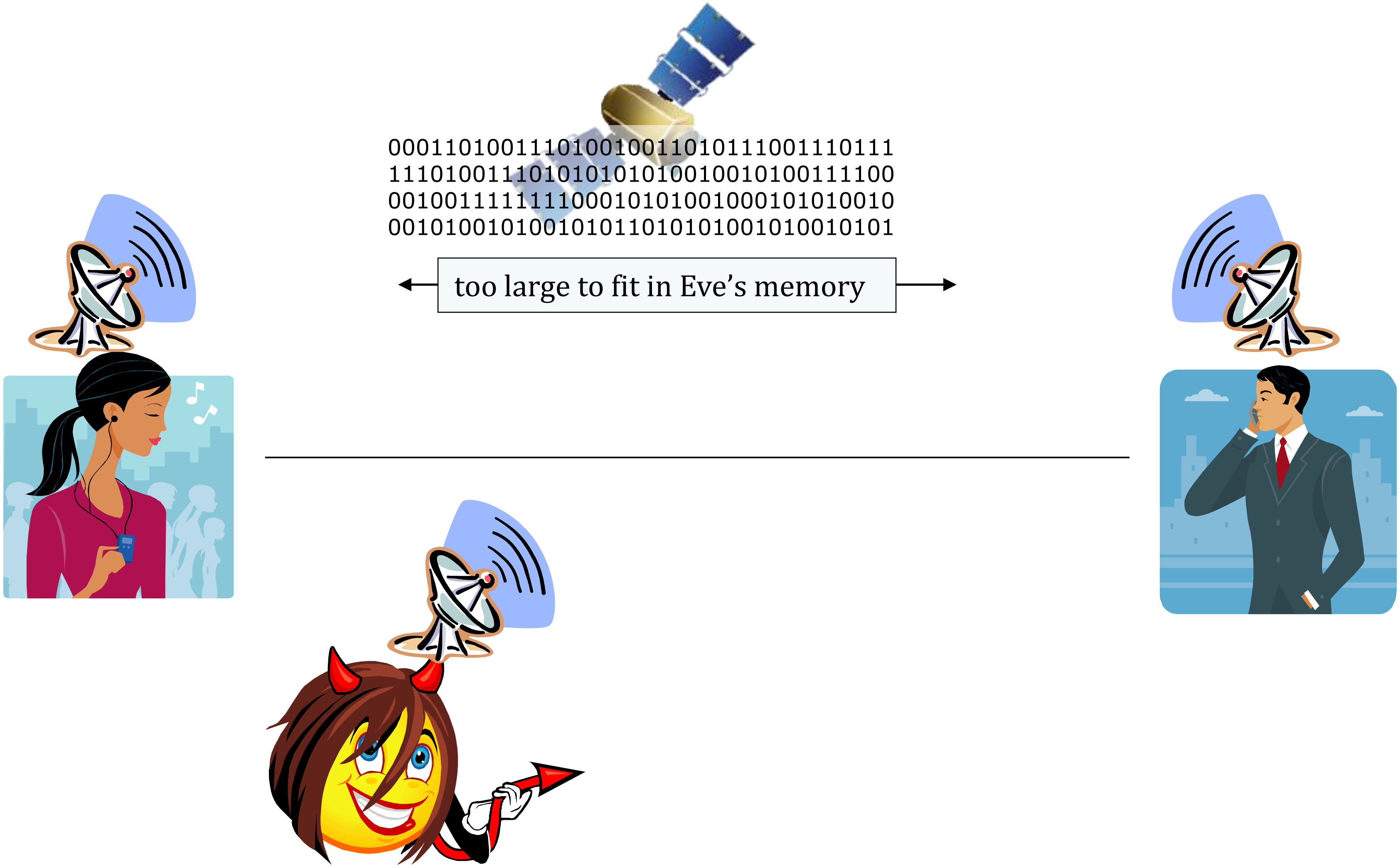
1 0 1 1 1 0 0 1 1 0 1 0 0 0 0

some bits get flipped
(because of the noise)

Assumption: the data that the adversary receives is noisy.
(The data that Alice and Bob receive may be even more noisy.)

Bounded-Storage Model

Another idea: bound the size of adversary’s memory



000110100111010010011010111001110111
111010011101010101010010010100111100
001001111111100010101001000101010010
001010010100101011010101001010010101

too large to fit in Eve’s memory

Plan

1. Introduction
2. Historical ciphers
3. Information-theoretic security
4. Computational security

How to reason about the bounded computing power?

perfect secrecy:
M and $\mathbf{Enc}_{K}(\boldsymbol{M})$
are independent

It is enough to require that

M and $\mathbf{Enc}_{K}(\boldsymbol{M})$

are independent
"***from the point of view of a computationally-limited adversary***".

How can this be formalized?

We will use the **complexity theory**!

Real cryptography starts here:

Eve is computationally-bounded

We will construct schemes that in **principle can be broken** if the adversary has a huge computing power.

For example, the adversary will be able to break the scheme by enumerating all possible secret keys.

(this is called a “**brute force attack**”)

Computationally-bounded adversary

Eve is computationally-bounded

But what does it mean?

Ideas:

1. "She has can use at most **1000**
   Intel Core 2 Extreme X6800 Dual Core Processors
   for at most **100** years..."

2. "She can buy equipment worth **1 million euro** and use it for **30** years..".

it's hard to reason
formally about it

A better idea

”The adversary has access to a **Turing Machine** that can make at most **10^{30}** steps.”

More generally, we could have definitions of a type:

“a system **X is (t,ε)-secure** if every **Turing Machine**

that operates in time **t**

can break it with probability at most **ε**.”

This would be quite precise, **but...**

We would need to specify exactly what we mean by a “**Turing Machine**”:

- **how many tapes does it have?**
- how does it access these tapes (maybe a “**random access memory**” is a more realistic model..)
- ...

*Moreover, this approach often leads to **ugly formulas**...*

What to do?

Idea

:

t steps of a Turing Machine → "**efficient computation**"

ε → a value "**very close to zero**".

How to formalize it?

Use the **asymptotics**!

Efficiently computable?

“efficiently computable” = “polynomial-time computable on a **Probabilistic Turing Machine**”

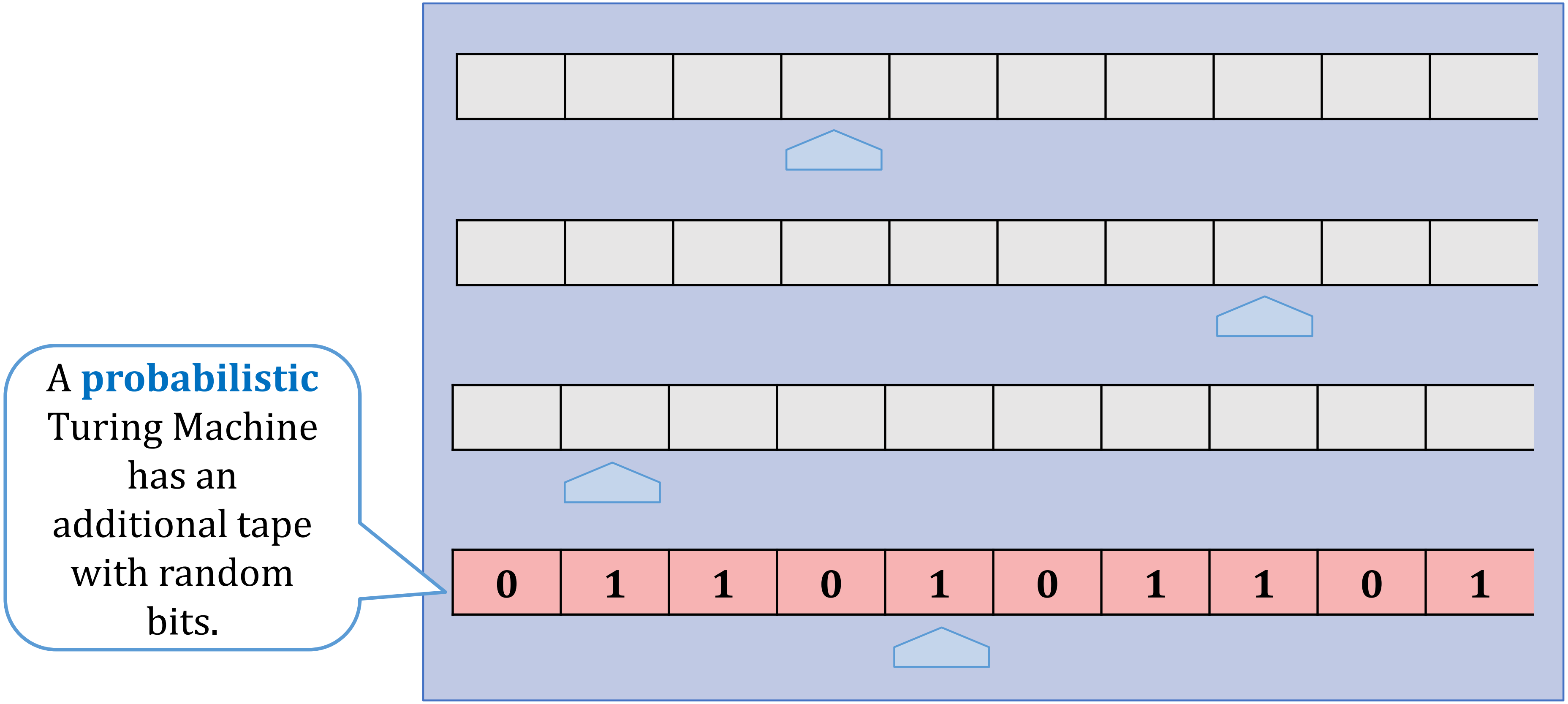
that is: running in time $O(n^c)$ (for some c)

Here we assume that the **poly-time Turing Machines** are the right model for the real-life computation.

Not true if a **quantum computer** is built...

Probabilistic Turing Machines

A standard Turing Machine has some number of tapes:



A **probabilistic** Turing Machine has an additional tape with random bits.

Some notation

If $\boldsymbol{M}$ is a Turing Machine then

$$\boldsymbol{M(X)}$$

is a **random variable** denoting the **output** of $\boldsymbol{M}$
assuming that
the contents of the random tape was chosen
uniformly at random.

More notation

$$\boldsymbol{Y \leftarrow M(X)}$$

means that the variable $\boldsymbol{Y}$ takes the value that $\boldsymbol{M}$ outputs on input $\boldsymbol{X}$ (assuming the random input is chosen uniformly).

If $\mathcal{A}$ is a set then

$$\boldsymbol{Y \leftarrow \mathcal{A}}$$

means that $\boldsymbol{Y}$ is chosen uniformly at random from the set $\mathcal{A}$.

Very small?

“very small”

=

“negligible”

=

approaches 0 faster than the inverse of any polynomial

<u>Formally</u>

A function $\boldsymbol{\mu} : \mathbf{N} \rightarrow \mathbf{R}$ is negligible if for every positive integer $\boldsymbol{c}$ there exists an integer $\mathbf{N}$ such that for all $\boldsymbol{x} > \mathbf{N}$

$$|\boldsymbol{\mu}(\boldsymbol{x})| \leq \frac{\mathbf{1}}{\boldsymbol{x}^{\boldsymbol{c}}}$$

Negligible or not?

$f(n) := \frac{1}{n^2}$	**no**
$f(n) := 2^{-n}$	**yes**
$f(n) := 2^{-\sqrt{n}}$	**yes**
$f(n) := n^{-\log n}$	**yes**
$f(n) := \frac{1}{n^{1000}}$	**no**

Nice properties of these notions

A sum of two polynomials is a polynomial:

poly + poly = poly

A product of two polynomials is a polynomial:

poly * poly = poly

A sum of two negligible functions is a negligible function:

negl + negl = negl

Moreover

A negligible function multiplied by a polynomial is negligible

negl * poly = negl

Security parameter

Typically, we will say that a **scheme X is secure** if

$\forall$ **polynomial-time** Turing Machine ***M*** **P (*M* breaks the scheme X) is negligible**

The terms "**negligible**" and "**polynomial**" make sense only if **X** and the **adversary** take an additional input $\mathbf{1^n}$ called

a **security parameter.**

In other words: we consider an infinite sequence **X(1),X(2),...** of schemes.

Example

security parameter $\boldsymbol{n}$ = the length of the secret key $\boldsymbol{k}$

in other words: $\boldsymbol{k}$ is always a random element of $\mathbf{\{0,1\}}^{n}$

The adversary can always **guess** $\boldsymbol{k}$ with probability $\mathbf{2}^{-n}$.

This probability **is negligible**.

He can also **enumerate all possible keys** $\boldsymbol{k}$ in time $\mathbf{2}^{n}$.
(the “brute force” attack)

This time **is exponential**.

Is this the right approach?

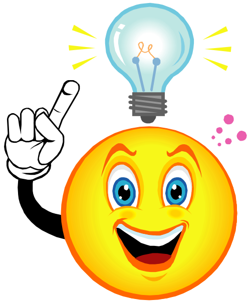
Advantages

1. All types of **Turing Machines** are “equivalent” up to a “**polynomial reduction**”.
   Therefore we do need to specify the details of the model.

2. The formulas get much simpler.

Disadvantage

Asymptotic results don’t tell us anything about security of the **concrete systems**.



However

Usually one can prove **formally** an asymptotic result and then argue **informally** that “the constants are reasonable”

(and can be calculated if one really wants).

How to change the security definition?

we will require that m_0,m_1 are chosen by a **poly-time adversary**

An encryption scheme is **perfectly secret** if for every m_0,$m_1 \in \mathcal{M}$

$\mathbf{Enc}(K, m_0)$ and $\mathbf{Enc}(K, m_1)$ are identically distributed

we will require that no **poly-time adversary** can distinguish
$\mathbf{Enc}(K, m_0)$ from $\mathbf{Enc}(K, m_1)$

A game

(Enc,Dec) – an encryption scheme

security parameter
1^n

adversary
(polynomial-time probabilistic Turing machine)

oracle

chooses $\boldsymbol{m_0,m_1}$ such that
$\boldsymbol{|m_0|=|m_1|}$

$\xrightarrow{\boldsymbol{m_0,m_1}}$

1. selects $\boldsymbol{k}$ randomly from $\boldsymbol{\{0,1\}^n}$
2. chooses a random $\boldsymbol{b = 0,1}$
3. calculates
   $\boldsymbol{c := \mathrm{Enc}(k,m_b)}$

$\xleftarrow{\boldsymbol{c}}$

has to guess $\boldsymbol{b}$

Alternative name: **has indistinguishable encryptions**

Security definition:
We say that **(Enc,Dec)** is **semantically-secure** if any **polynomial time** adversary guesses $\boldsymbol{b}$ correctly with probability at most $\frac{1}{2} + \varepsilon(n)$, where $\boldsymbol{\varepsilon}$ is negligible.

Testing the definition

Suppose the adversary can compute ***k*** from **Enc(*k*,*m*)**. Can he win the game?

YES!

Suppose the adversary can compute **some bit of *m*** from **Enc(*k*,*m*)**.Can he win the game?

YES!

Multiple messages

In real-life applications we need to encrypt **multiple messages with one key**.

The adversary may learn something about the key by looking at

ciphertexts $c_1,\ldots,c_t$ of

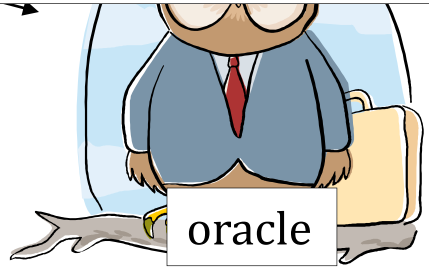
some messages $m_1,\ldots,m_t$.

How are these messages chosen?
let's say: the adversary can choose them!

(good tradition: be as pessimistic as possible)

A chosen-plaintext attack (CPA)

security parameter
1^n



1. chooses random $\boldsymbol{k} \leftarrow \{0,1\}^n$
2. chooses random $\boldsymbol{b} \leftarrow \{0,1\}$

oracle

chooses m'_1

m'_1

c_1 = Enc(k,m'_1)

⋮

chooses m'_t

m'_t

c_t = Enc(m'_t)

challenge phase:

chooses m_0,m_1

m_0,m_1

c = Enc(k,m_b)

the interaction continues . . .

has to guess b

CPA-security

Alternative name: CPA-secure

Security definition

We say that **(Enc,Dec)** **has indistinguishable encryptions under a chosen-plaintext attack (CPA)** if

every **randomized polynomial time** adversary
guesses $\boldsymbol{b}$ correctly
with probability at most $\frac{1}{2} + \boldsymbol{\varepsilon}(\boldsymbol{n})$, where $\boldsymbol{\varepsilon}$ is negligible.

Observation

A **CPA-secure** encryption scheme cannot be deterministic.

Typical options:

- **Enc** has a “state” (e.g. a counter)
- **Enc** is randomized, i.e., it takes as additional input:
  - some perfect randomness ***R***, or
  - takes as an **nonce** ***R***

weaker requirement

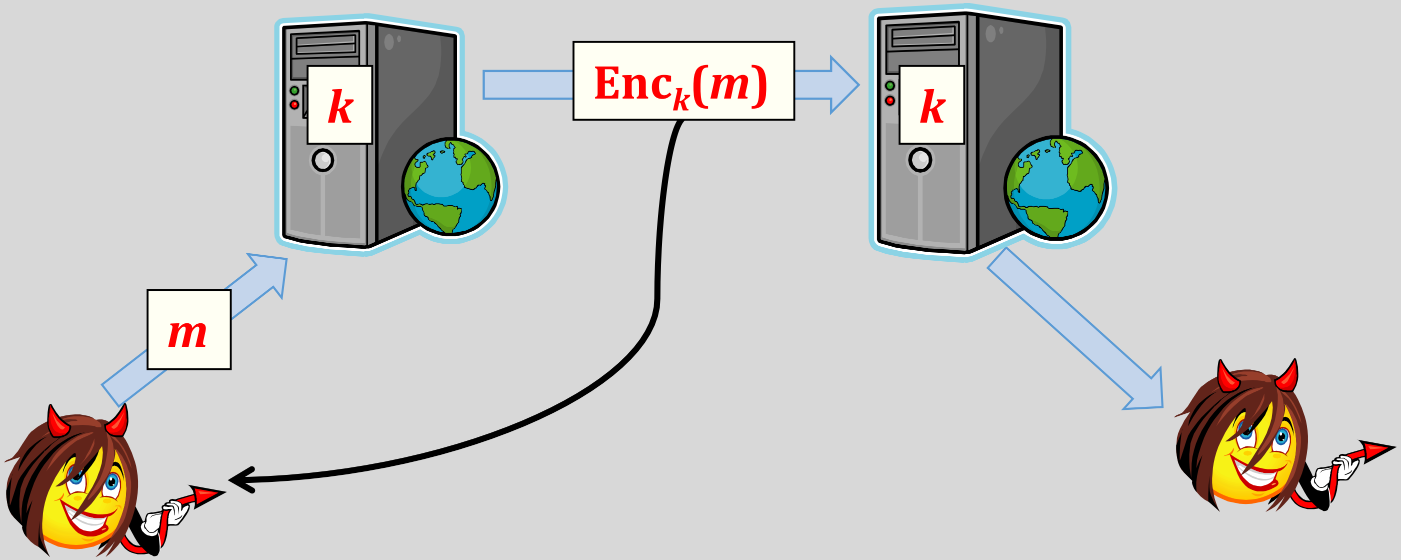
nonce = “**n**umber used **once**”

CPA in real-life

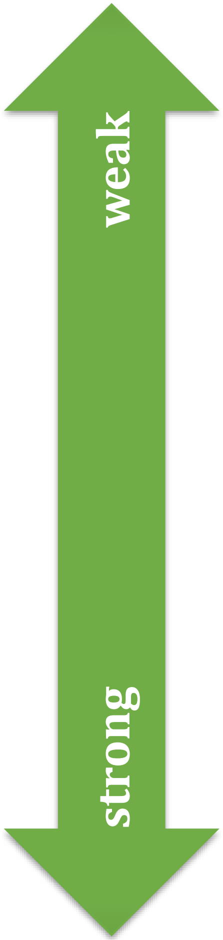
Q: Aren’t we too pessimistic?

A: No! **CPA** can be implemented in practice.

Example: routing



Other attacks known in the literature



- **ciphertext-only attack** – the adversary has no information about the plaintext

- **known plaintext attack** – the plaintext are drawn from some distribution that the adversary does not control

- **batch chosen-plaintext attack** – like the **CPA** attack, but the adversary has to choose $\boldsymbol{m_1,\ldots,m_t}$ at once.

(“our” **CPA**-attack is also called the “**adaptive CPA-attack**”)

- **chosen ciphertext attack** – we will discuss it later…