Mid-term 25th October

Everyone in CM3

2 Groups – 2 exams

10:15-11:30	From Abbey to Jalal
11:45-13:00	From Jeanmonod to Zrouga

You need to be seated at that point

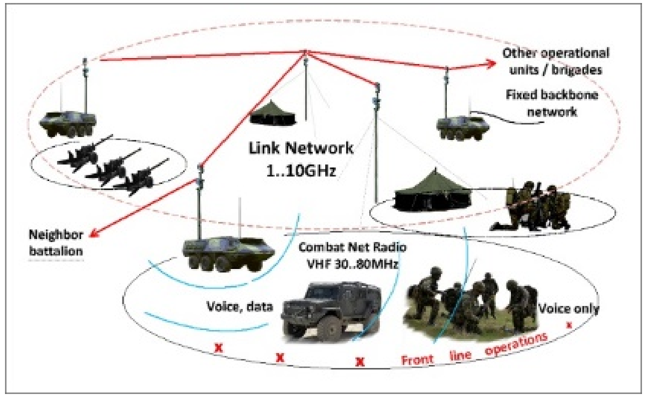
1h15 for the exam

1st group: if you enter the room you **CANNOT** leave before 11:30 and you **CANNOT** use your phone/computer

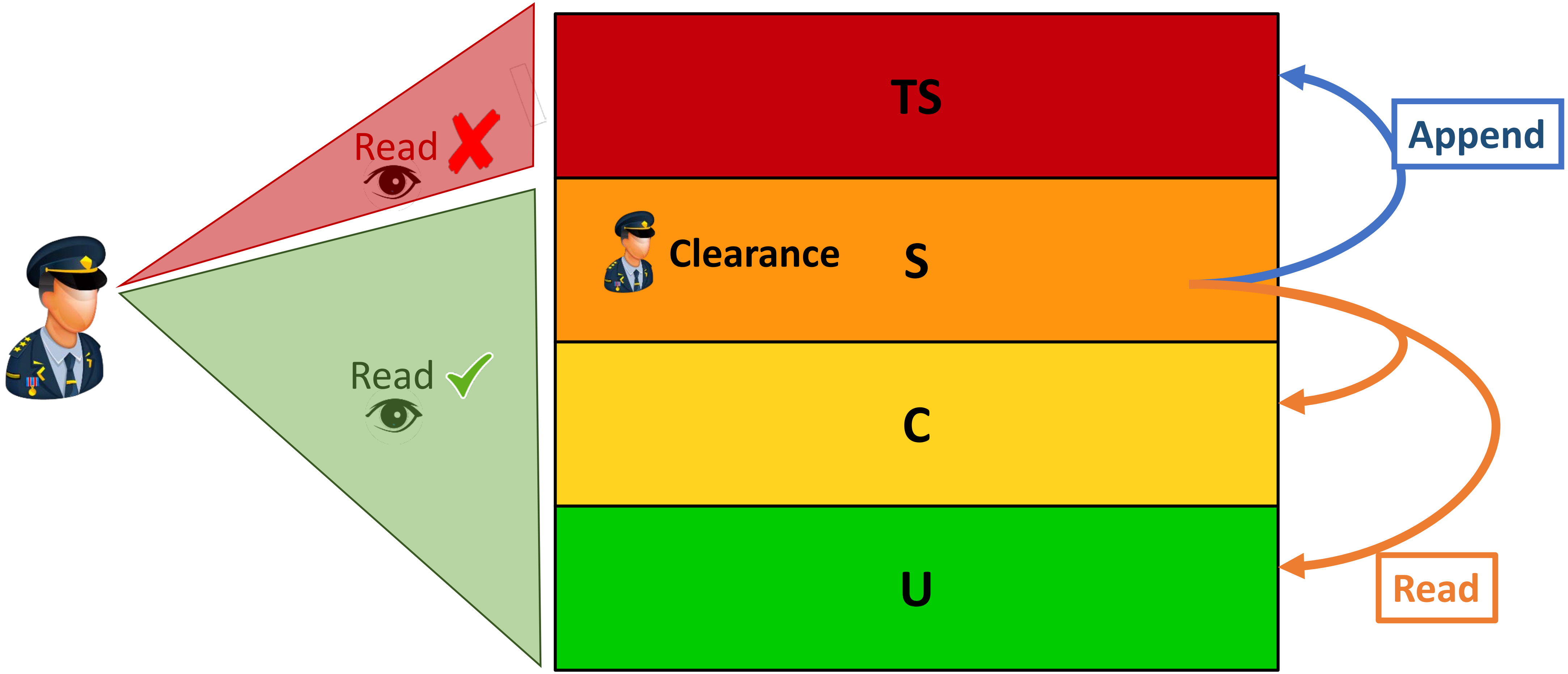
Last week – Mandatory Access Control

Central security policy *assigns permissions*

🏥



Last week – Bell LaPadula model - **Confidentiality**



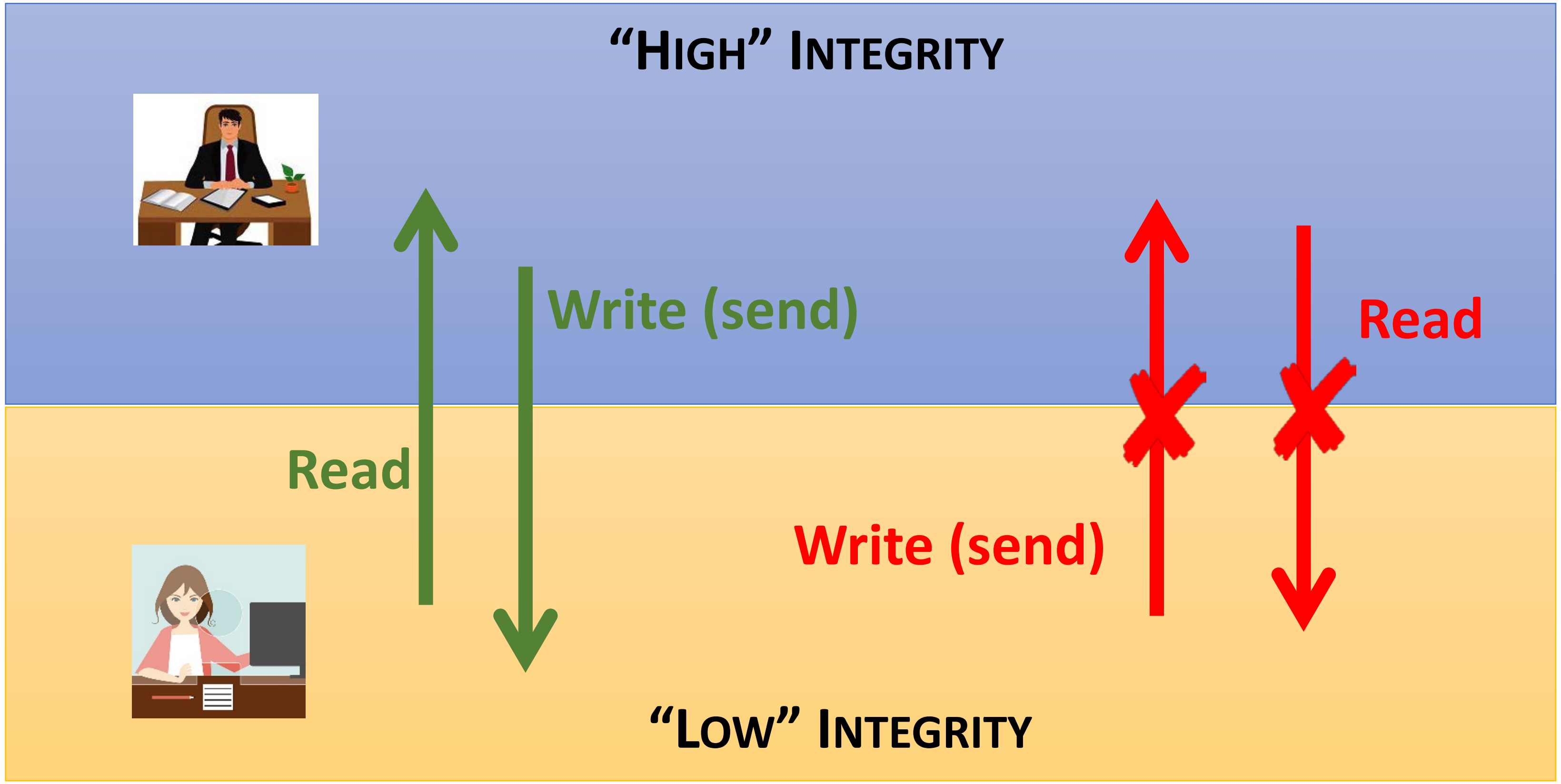
No Read Up (NRU)

ss-property

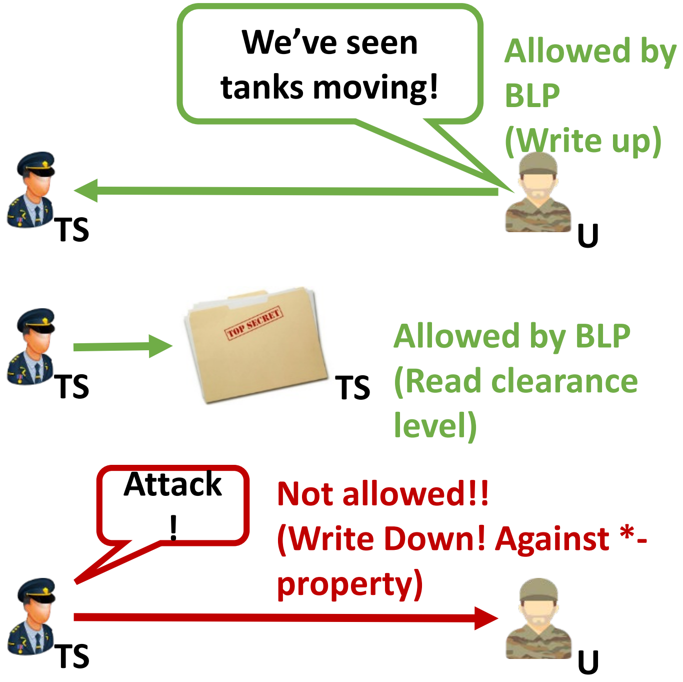
No Write Down (NWD)

***-property**

Last week – Biba - **Integrity**



Declassification


We've seen tanks moving!

Allowed by BLP (Write up)

TS

U

TS

TS

Allowed by BLP (Read clearance level)

Attack !

Not allowed!! (Write Down! Against *-property)

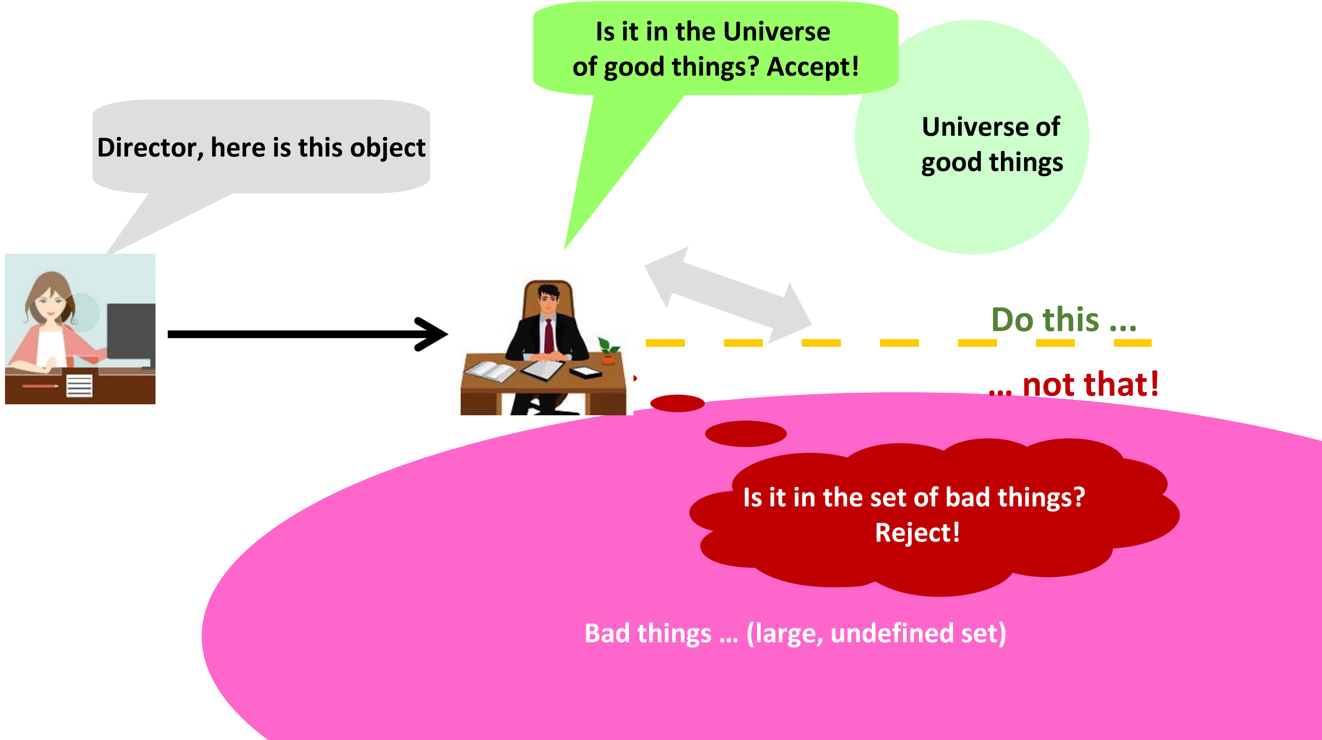
TS

U

Beware of these flows!

Sanitization

Covert channels difficult to catch!


Director, here is this object

Is it in the Universe of good things? Accept!

Universe of good things

Do this …

… not that!

Is it in the set of bad things? Reject!

Bad things … (large, undefined set)

Combining security properties

Secure composition of mechanisms is hard!

Composing confidentiality and integrity

- BLP: confidentiality, no integrity
- BIBA: integrity, no confidentiality
- Example we study: Chinese Wall Model

From multi-level to multi-lateral security

- “Different” entities seek different properties.
- These properties may be opposed to each other.

Chinese Wall model

Inspiration: UK rules about handling “conflicts of interest” in the financial sector.

- A separation must exist at all times, even within the same firm, between people engaging in activities that conflict with each other.

- Cost of failure: large fines and reputation

Consultancy services for different clients

Financial advice and auditing of same client

David FC. Brewer and Michael J. Nash. “The Chinese Wall Security Policy.” in IEEE SSP 1989

Chinese Wall model: Entities and Basic Concepts

All objects are associated with a label denoting their origin

“Pepsi Ltd.”, “Coca-Cola Co.”, “Microsoft Audit”, “Microsoft Investments”

The originators define “conflict sets” of labels

{“Pepsi Ltd.”, “Coca-Cola Co.”}, {“Microsoft Audit”, “Microsoft Investments”}

Pepsi

Coca-cola

Subjects are associated with a <u>history</u> of their accesses to objects, and in particular their labels.

Audit

Invest

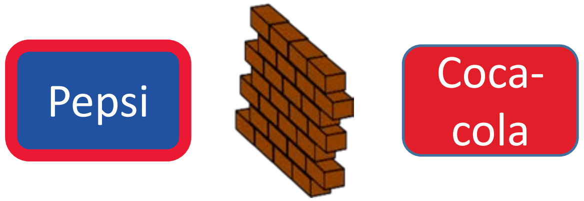
Chinese Wall model: Entities and Basic Concepts

All objects are associated with a label denoting their origin

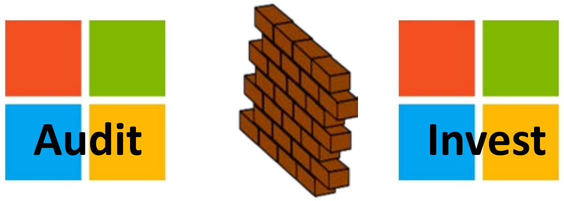
“Pepsi Ltd.”, “Coca-Cola Co.”, “Microsoft Audit”, “Microsoft Investments”

The originators define “conflict sets” of labels

{“Pepsi Ltd.”, “Coca-Cola Co.”}, {“Microsoft Audit”, “Microsoft Investments”}



Subjects are associated with a <u>history</u> of their accesses to objects, and in particular their labels.



Chinese Wall model: Entities and Basic Concepts

All objects are associated with a label denoting their origin

“Pepsi Ltd.”, “Coca-Cola Co.”, “Microsoft Audit”, “Microsoft Investments”

The originators define “conflict sets” of labels

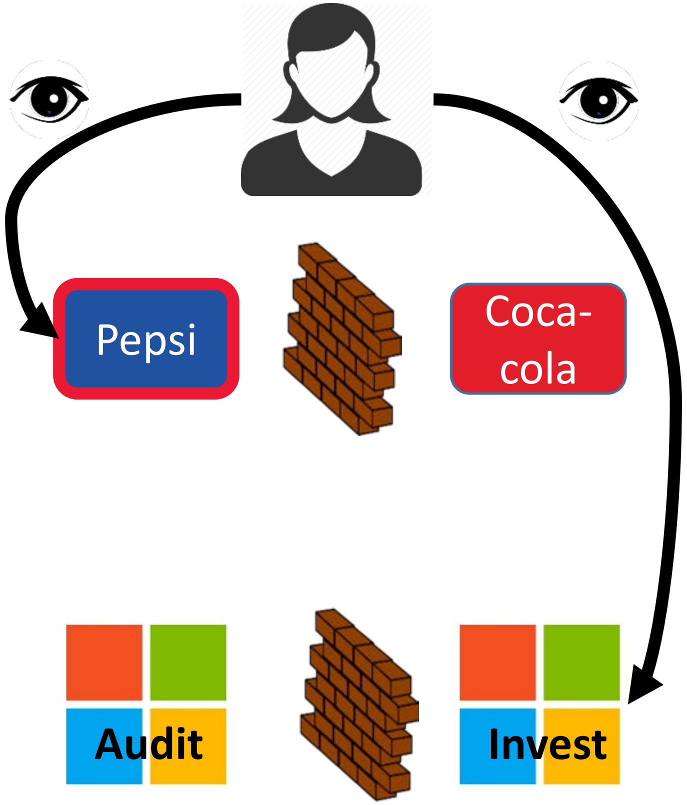
{“Pepsi Ltd.”, “Coca-Cola Co.”}, {“Microsoft Audit”, “Microsoft Investments”}

Subjects are associated with a history of their accesses to objects, and in particular their labels.

Alice

1. Pepsi

2. Microsoft Invest



Chinese Wall model: Access rules

A subject can read an object (for either read or write) if the access ***does not allow an information flow*** between items with labels in the same conflict set

Alice starts her first day at work

1) She accesses files of “Pepsi Ltd” (OK)

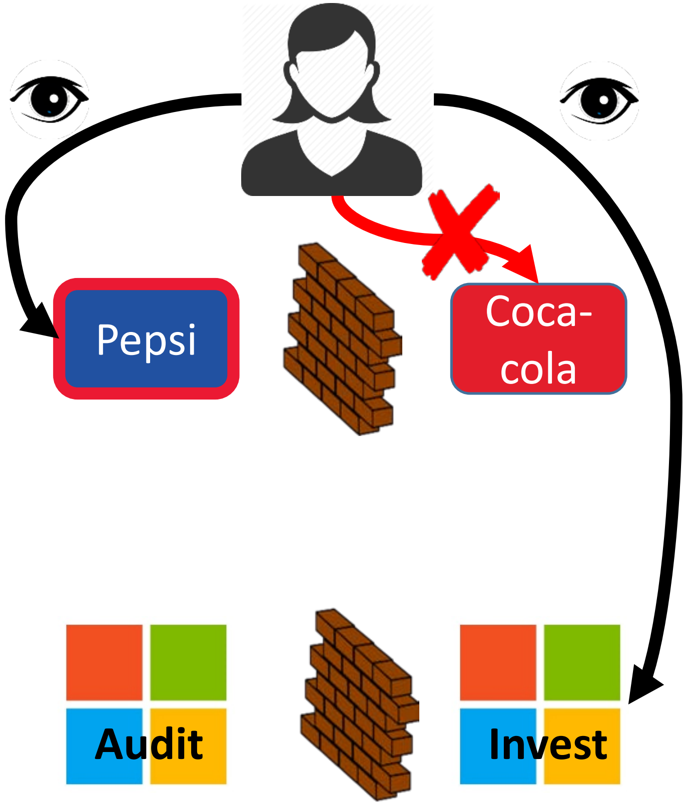
2) She accesses files of “Microsoft invest” (OK)

3) She tries to access files of “Coca-cola Co.” (access denied!)

ALICE

1. Pepsi

2. Microsoft Invest



Why?

Chinese Wall model: Access rules

A subject can read an object (for either read or write) if the access ***does not allow an information flow*** between items with labels in the same conflict set

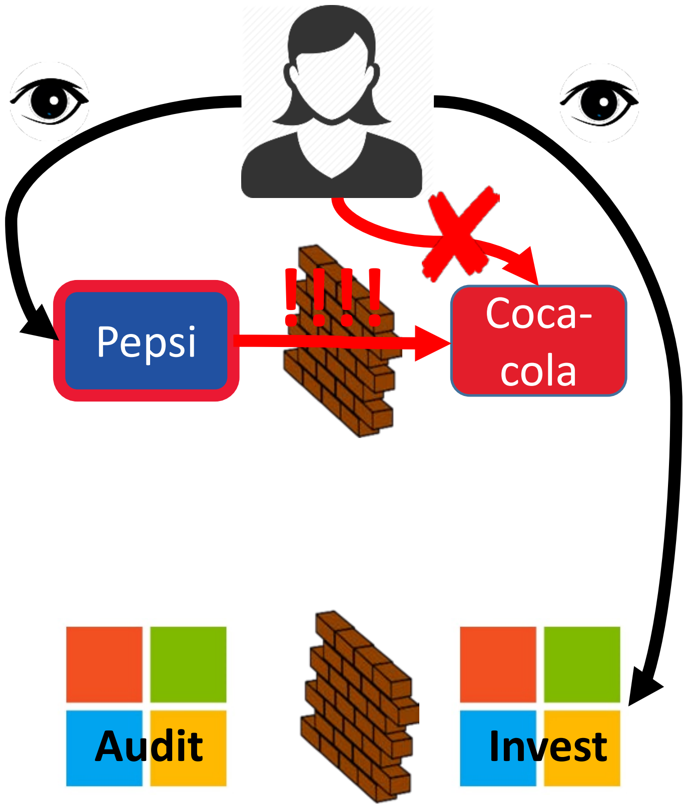
Alice starts her first day at work

Alice
1. Pepsi
2. Microsoft Invest

1) She accesses files of "Pepsi Ltd" (OK)

2) She accesses files of "Microsoft invest" (OK)

3) She tries to access files of "Coca-cola Co." (access denied!)



Why? She has already accessed files from "Coca-cola Co." thus an <u>information flow</u> between those and "Pepsi Ltd" might happen

Chinese Wall model: Indirect flows

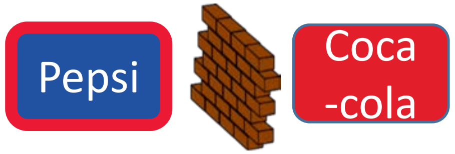
Direct flow within a conflict set is easy to detect! What about indirect?

Alice and Bob start together

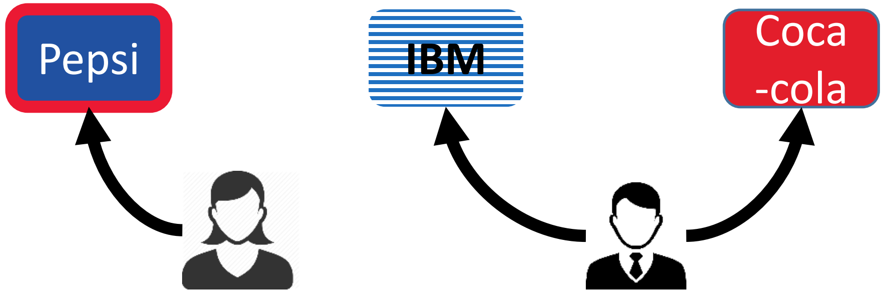
1) Alice is assigned to “Pepsi Ltd” (OK)

2) Bob is assigned to “Coca-cola Co.” and “IBM Co.” (OK)

3) Alice tries to access files of “IBM Co.” (access denied!)



Why?



Chinese Wall model: Indirect flows

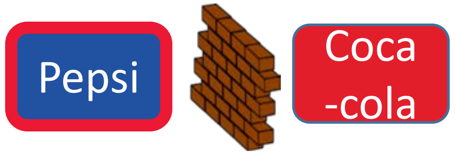
Direct flow within a conflict set is easy to detect! What about indirect?

Alice and Bob start together

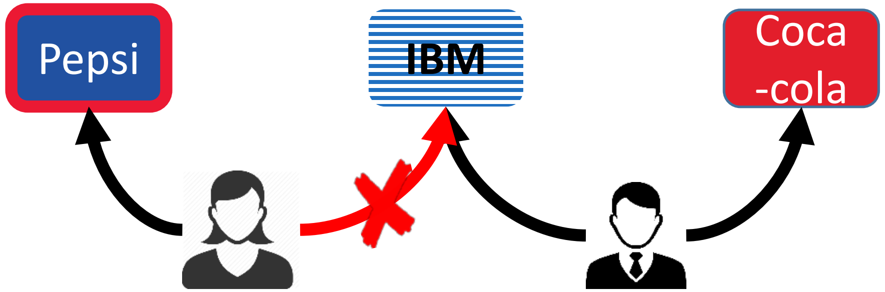
1) Alice is assigned to “Pepsi Ltd” (OK)

2) Bob is assigned to “Coca-cola Co.” and “IBM Co.” (OK)

3) Alice tries to access files of “IBM Co.” (access denied!)



Why?



Chinese Wall model: Indirect flows

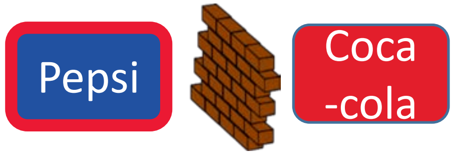
Direct flow within a conflict set is easy to detect! What about indirect?

Alice and Bob start together

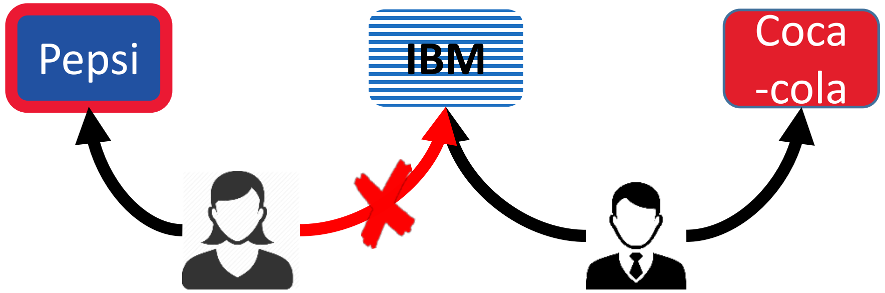
1) Alice is assigned to “Pepsi Ltd” (OK)

2) Bob is assigned to “Coca-cola Co.” and “IBM Co.” (OK)

3) Alice tries to access files of “IBM Co.” (access denied!)



Why? If she writes in IBM with her knowledge of Pepsi, then the information ***may*** flow to Coca-cola.



SANITIZATION is necessary for business

“Un-label” some items as long as the information cannot lead to any conflict of interest, e.g., extract some “general market information”

Multilateral security

At least two principals require two different, “incompatible” or even conflicting security properties

- *Both Alice and Bob want financial privacy, but they wish to compute who is richest*

- *Alice wants privacy of her location, but her insurer Bob want to make sure she is not driving much on rural roads (integrity)*

- *Users of an on-line service want anonymity, but the service wants to identify who is committing fraud*

- *Elections: need to both provide privacy and integrity (even if you lose)*

- *In telephony: users want privacy, the network wants to be paid for communications*

Who secures the TCB?

How to deal with multilateral security

1. Use a trusted third party

- They need to be trusted by all parties to enforce the security properties all parties care about

- **Single point of failure** (5Cs: Cost, Compulsion, Collusion, Corruption, Carelessness)

2. Use some form of secure hardware

- One party provides the hardware, that is used by the other

SIM card in your mobile phone - Keeps the authentication keys away from YOU

Digital Rights Management controls in DVD players - Keeps you away from some functions of the equipment

- Single point of failure - manufacturer?

3. Modern Cryptography.

- Can enforce any multilateral security property

- At what cost? Advanced Privacy Enhancing Technologies master course!

Computer Security (COM-301)
Applied cryptography

Carmela Troncoso

SPRING Lab

carmela.troncoso@epfl.ch

Some slides/ideas adapted from: George Danezis, Yoshi Kohno

Textbooks

Ross Anderson & Dieter Gollmann: Chapters on Cryptography

Handbook of Applied Cryptography by A. Menezes, P. van Oorschot and S. Vanstone

Dan Boneh - https://www.coursera.org/learn/crypto

Jonathan Kazt - https://www.coursera.org/learn/cryptography

More advanced than this lecture

Warning: Don't try this at home!

2-hour introduction to applied cryptography does not qualify you to design cryptographic primitives or protocols!

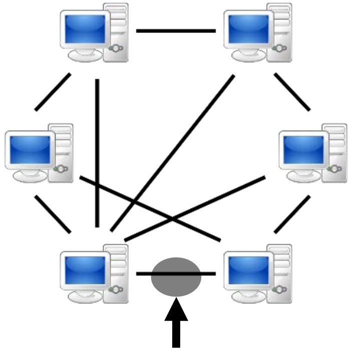
This course: What you can expect from cryptographic algorithms and how to use them in a security system

What is missing?

- Cryptanalysis
- How to prove formally that a scheme is secure
- How to securely implement cryptographic schemes

To do these you need a real cryptographer!

Why cryptography matters?



Data in transit

Data at rest

What is the TCB?

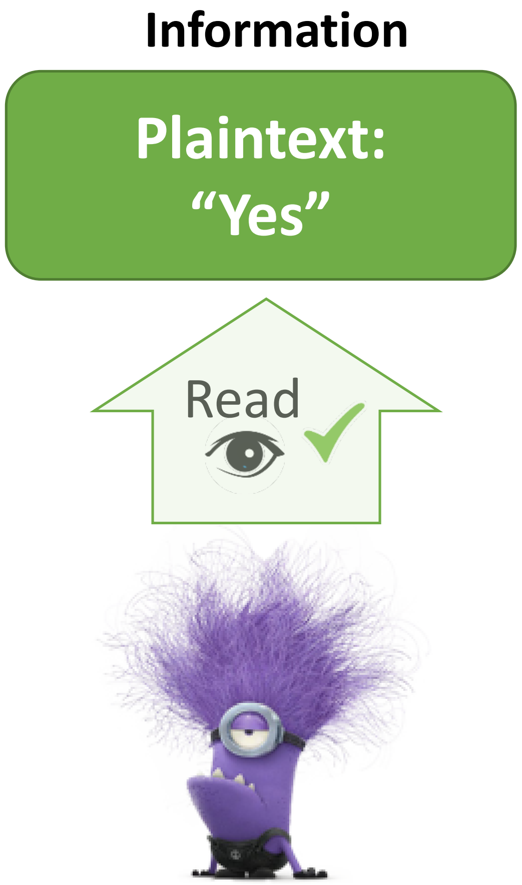
CRYPTOGRAPHY

Frees you from physical security

Reduces TCB to the confidentiality or integrity of keys

Glossary

Confidentiality: information cannot be accessed by unauthorized parties

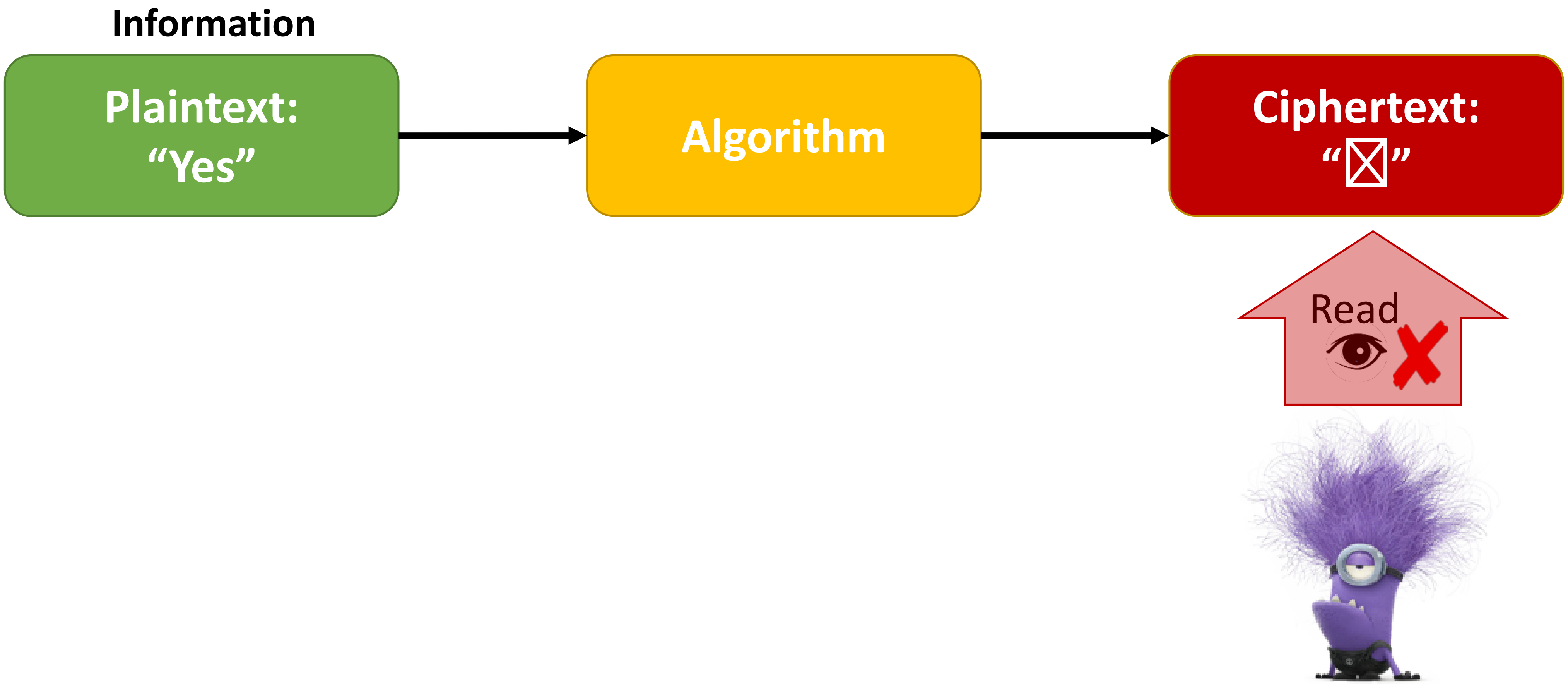


Information

Plaintext: “Yes”

Glossary

Confidentiality: information cannot be accessed by unauthorized parties



Information

Plaintext: "Yes"

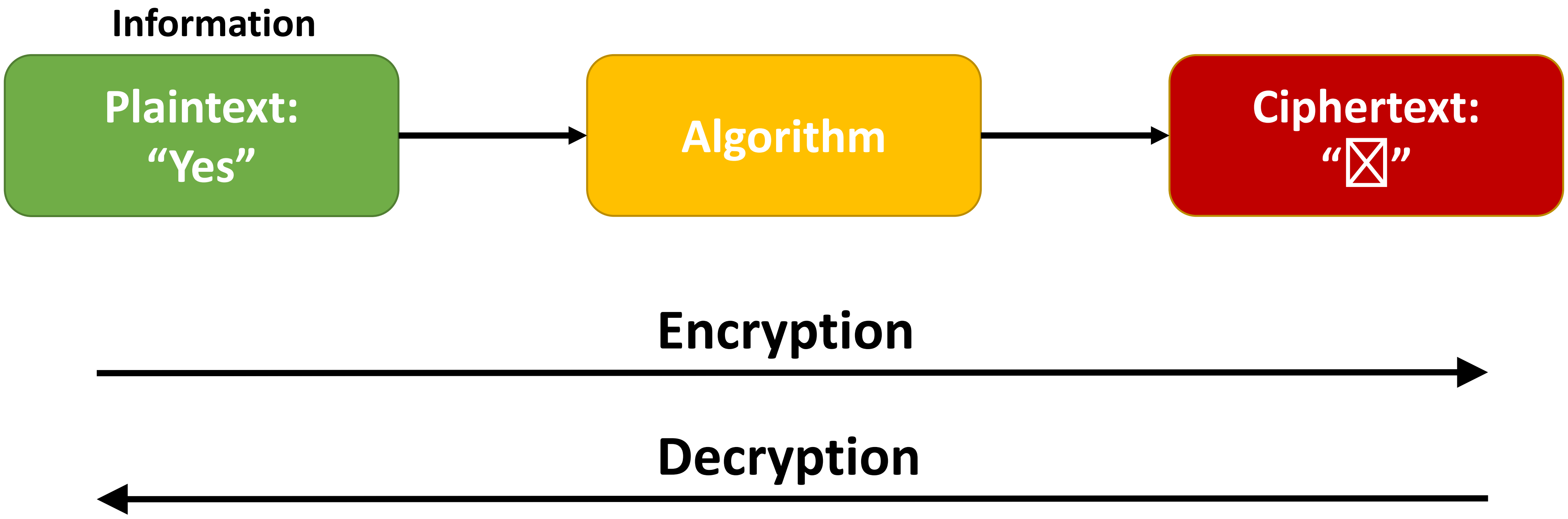
Algorithm

Ciphertext: "⊠"

Read

Glossary

Confidentiality: information cannot be accessed by unauthorized parties



Information

Plaintext:
"Yes"

Algorithm

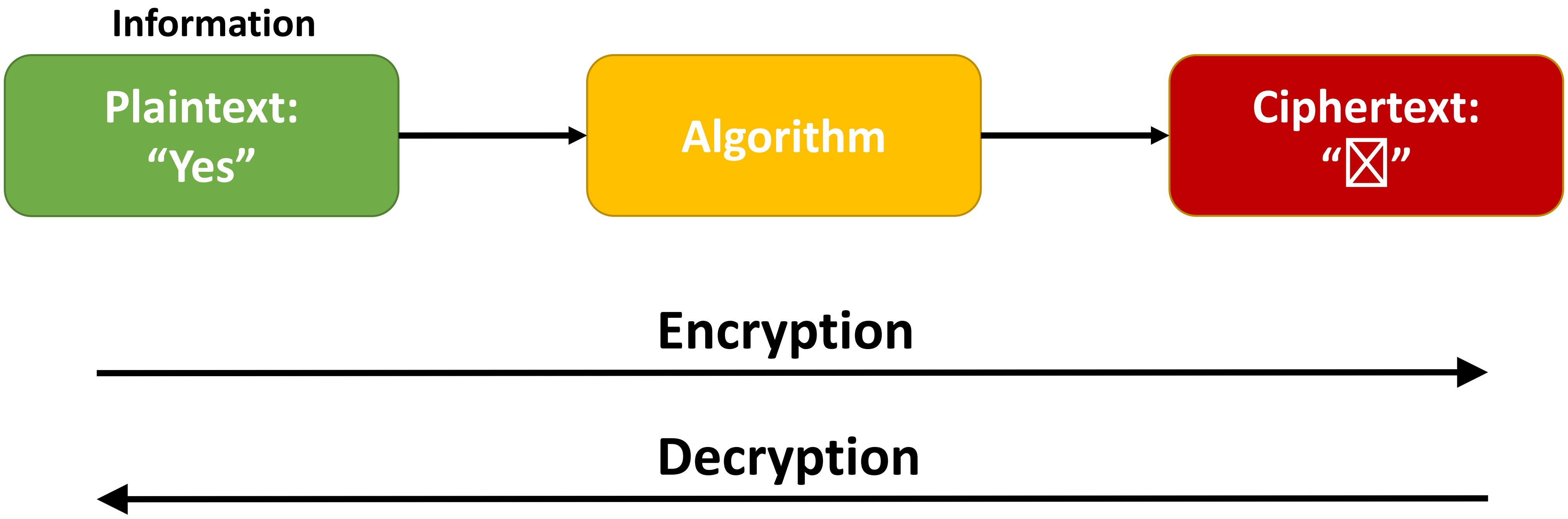
Ciphertext:
"⊠"

Encryption

Decryption

Glossary

Confidentiality: information cannot be accessed by unauthorized parties



Information

Plaintext: "Yes"

Algorithm

Ciphertext: "☒"

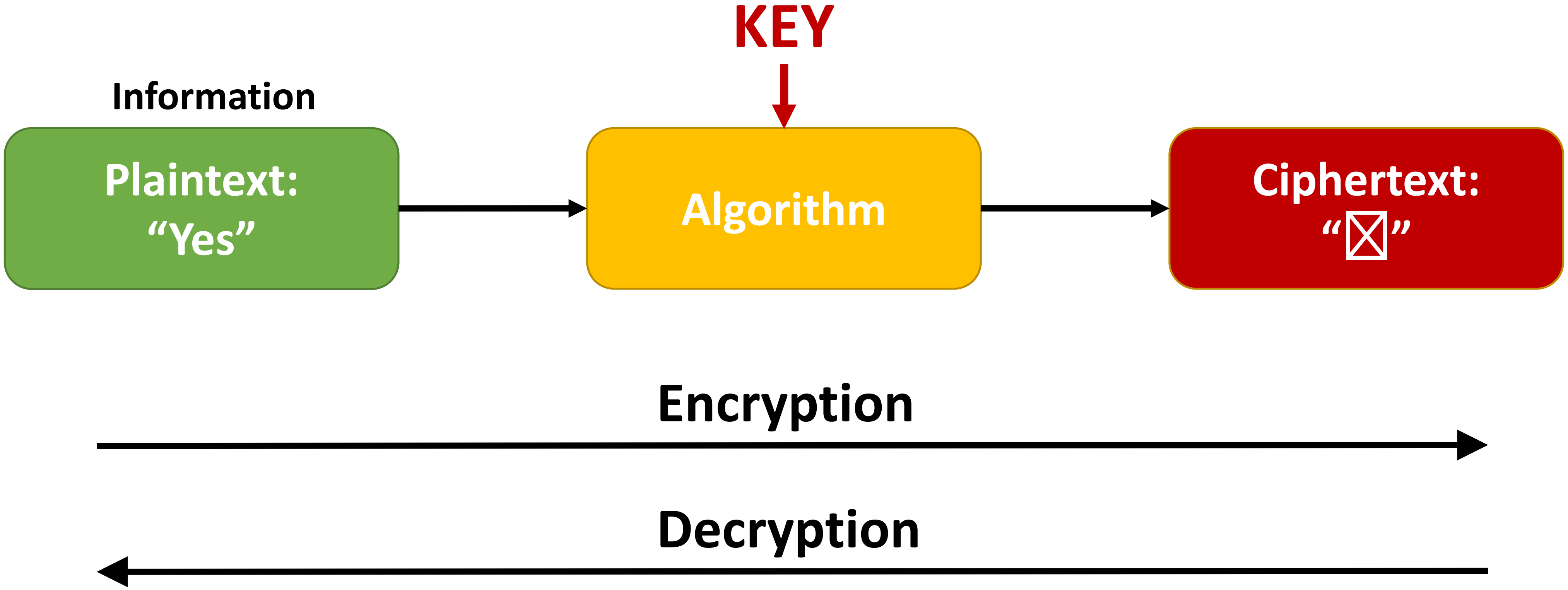
Encryption

Decryption

🤔 This sounds like encoding…

Glossary

Confidentiality: information cannot be accessed by unauthorized parties



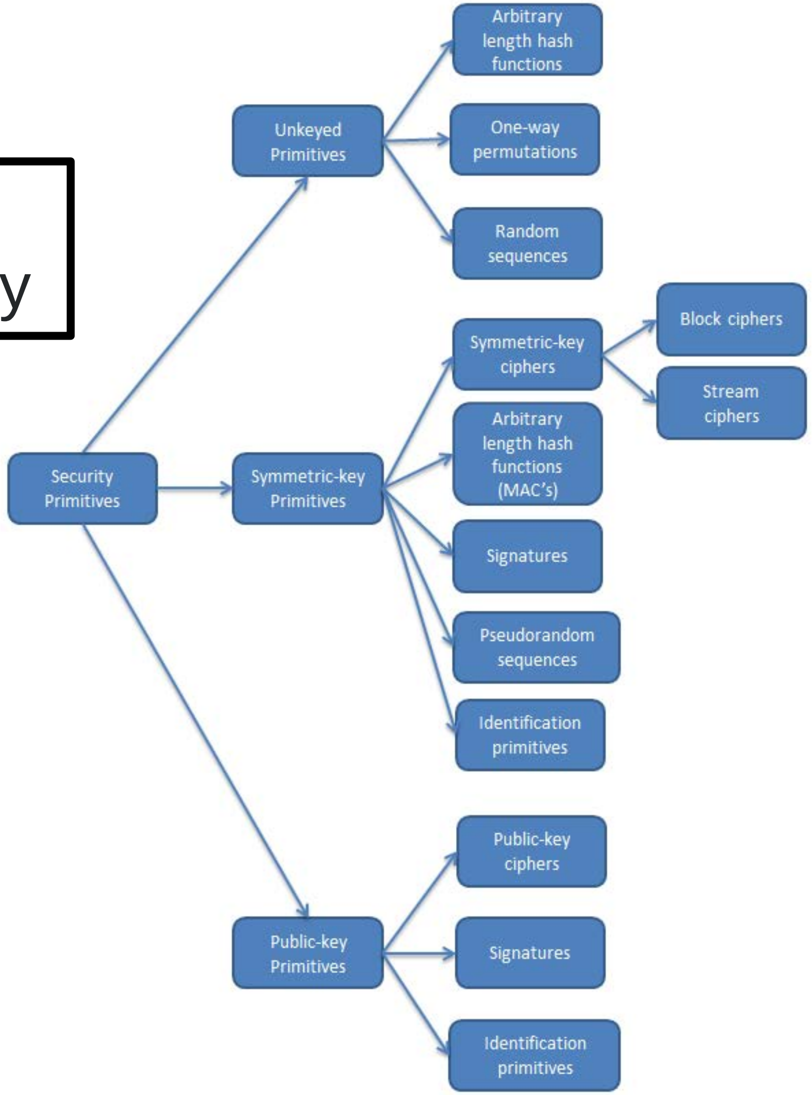
As opposed to encoding, encryption cannot be reversed without a KEY

Glossary

Cryptographic primitives
universal, exchangeable building blocks in cryptography

What exactly a primitive is depends on the level of abstraction

The most basic primitives are those, where a function is considered secure, but either you can't break it down any further or there is no security argument for its individual parts



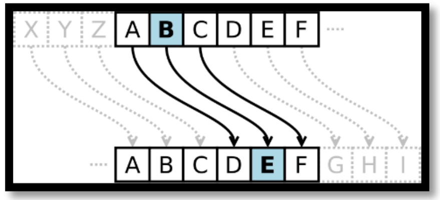
https://crypto.stackexchange.com/questions/39735/whats-a-cryptographic-primitive-really

https://i.stack.imgur.com/2yBJf.png

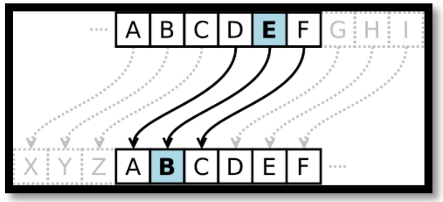
Some history – the quest for confidentiality

Caesar's cipher (50 BC)

Choose a shift (3 for Julius Caesar) and rotate the alphabet



Encrypt



Decrypt

hello world → khoor zruog

Kamasutra cipher (400 AD)

Choose a permutation of the alphabet

Key: HOWBUGIACRYEVZXPJQMSNTFDKL

```
HOWBUGIACRYEV
ZXPJQMSNTFDKL
```

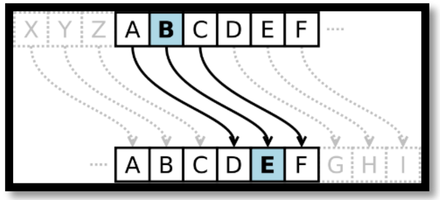
Encrypt / Decrypt: substitute by opposite letter

hello world → zkvvx pxfvy

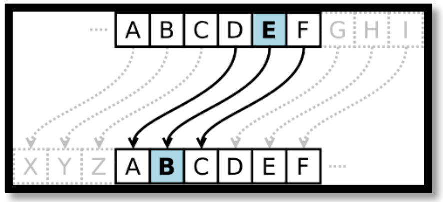
Some history – the quest for confidentiality

Caesar's cipher (50 BC)

Choose a shift (3 for Julius Caesar) and rotate the alphabet



Encrypt



Decrypt

hello world → khoor zruog

Kamasutra cipher (400 AD)

Choose a permutation of the alphabet

Key: `HOWBUGIACRYEVZXPJQMSNTFDKL`

```
HOWBUGIACRYEV
ZXPJQMSNTFDKL
```

Encrypt / Decrypt: substitute by opposite letter

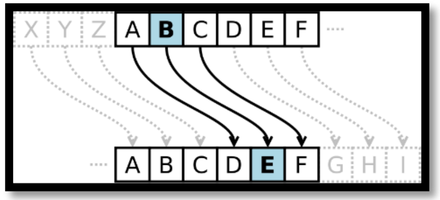
hello world → zkvvx pxfvy

Problem??

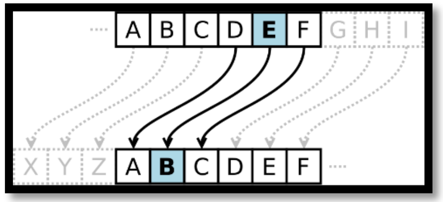
Some history – the quest for confidentiality

Caesar's cipher (50 BC)

Choose a shift (3 for Julius Caesar) and rotate the alphabet



Encrypt



Decrypt

hello world → khoor zruog

Kamasutra cipher (400 AD)

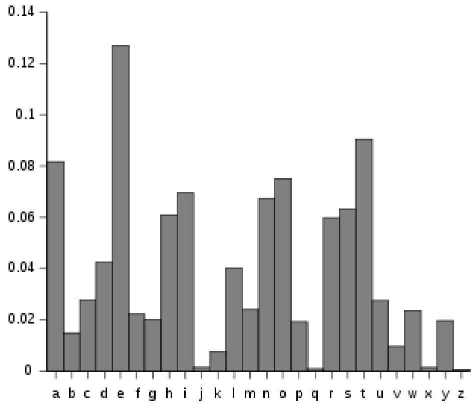
Choose a permutation of the alphabet

Key: HOWBUGIACRYEVZXPJQMSNTFDKL

HOWBUGIACRYEV
ZXPJQMSNTFDKL

Encrypt / Decrypt: substitute by opposite letter

hello world → zkvvx pxfvy

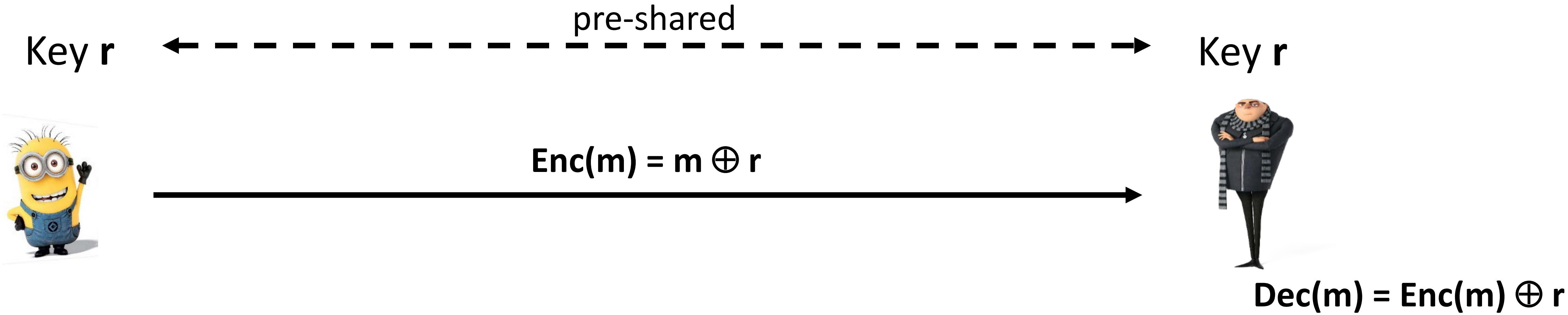


Problem??

Frequency analysis!

The perfect cipher: One Time Pad

Key = string of **random** bits as long as the message



pre-shared

Key **r**

Key **r**

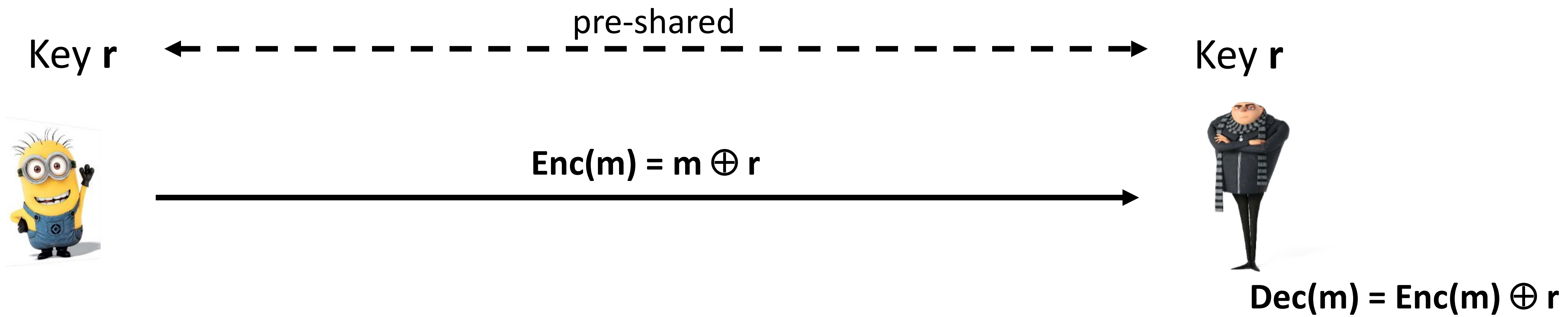
Enc(m) = m ⊕ r

Dec(m) = Enc(m) ⊕ r

Message	YEAH
Binary (ASCII)	01111001011001010110000101101000
Pad	01110101000111010100101001001010
Encryption	00001100011110000010101100100010

The perfect cipher: One Time Pad

Key = string of **random** bits as long as the message



Key **r**

pre-shared

Key **r**

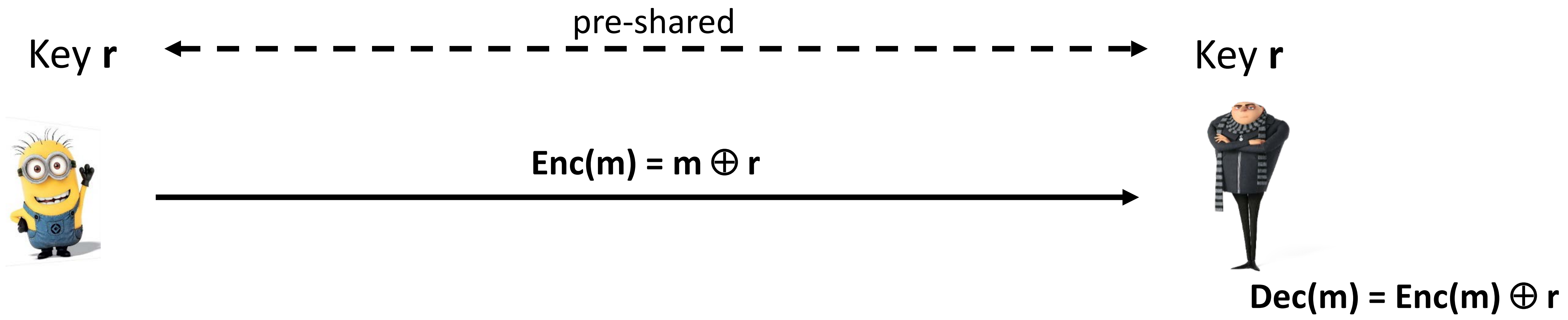
Enc(m) = m ⊕ r

Dec(m) = Enc(m) ⊕ r

Message	YEAH		N0PE
Binary (ASCII)	01111001011001010110000101101000		01101110011011110111000001100101
Pad	01110101000111010100101001001010	**same** →	01110101000111010100101001001010
Encryption	00001100011110000010101100100010		00011011011100100011101000101111

The perfect cipher: One Time Pad

Key = string of **random** bits as long as the message



pre-shared

Key **r**

Key **r**

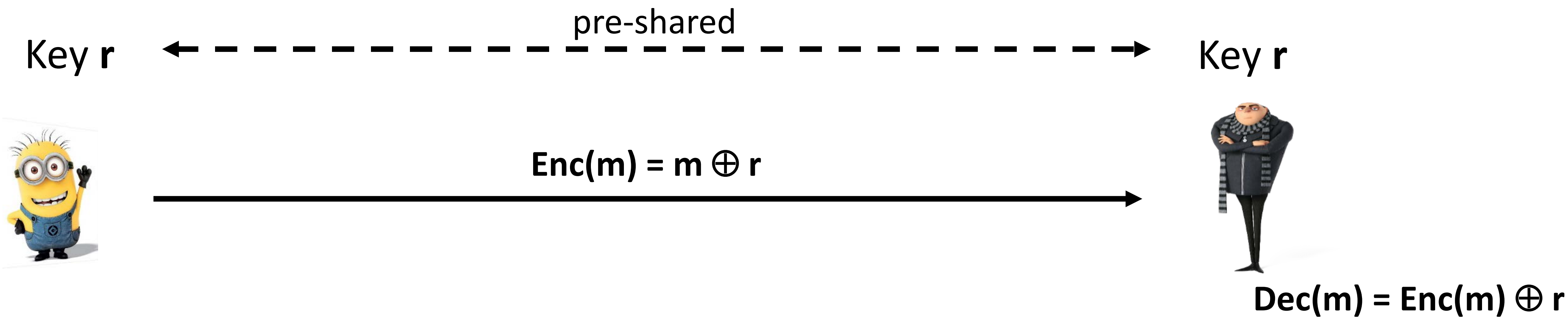
Enc(m) = m ⊕ r

Dec(m) = Enc(m) ⊕ r

Message	YEAH		YEAH
Binary (ASCII)	01111001011001010110000101101000	**same** →	01111001011001010110000101101000
Pad	01110101000111010100101001001010		10101001010100101010001001010010
Encryption	00001100011110000010101100100010		11010000001101111100001100111010

The perfect cipher: One Time Pad

Key = string of **random** bits as long as the message



Key **r**

pre-shared

Key **r**

Enc(m) = m ⊕ r

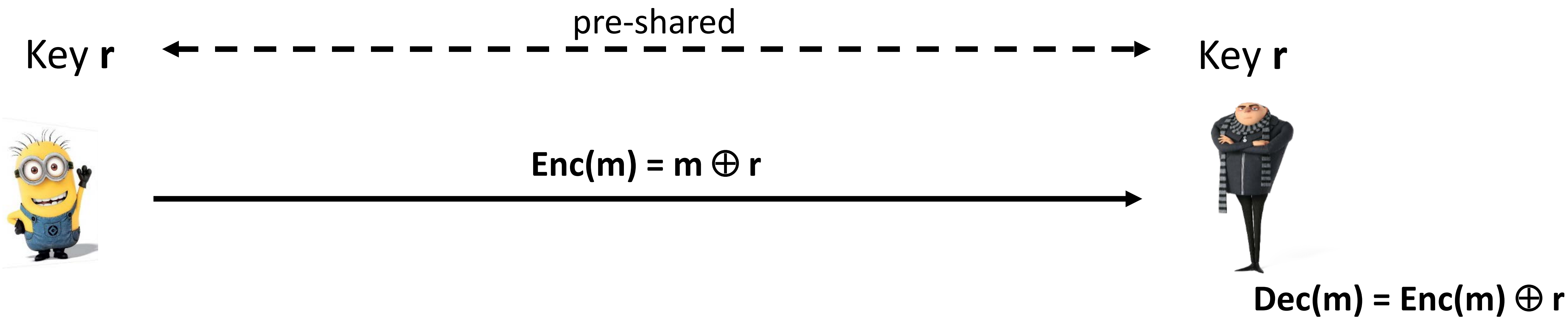
Dec(m) = Enc(m) ⊕ r

Message	YEAH		YEAH
Binary (ASCII)	01111001011001010110000101101000	**same** →	01111001011001010110000101101000
Pad	01110101000111010100101001001010		10101001010100101010001001010010
Encryption	00001100011110000010101100100010		11010000001101111100001100111010

Delete “r” – must never be reused!

The perfect cipher: One Time Pad

Key = string of **random** bits as long as the message



Message	YEAH		YEAH
Binary (ASCII)	01111001011001010110000101101000	**same** →	01111001011001010110000101101000
Pad	01110101000111010100101001001010		10101001010100101010001001010010
Encryption	00001100011110000010101100100010		11010000011011111000011001111010

Delete "r" – must never be reused!

(msg1 ⊕ pad) ⊕ (msg2 ⊕ pad) → (msg1 ⊕ msg2)

Reveals where msg differ

Frequency analysis works

ASCII patterns (space or letter)

00- **01-**

The perfect cipher: One Time Pad

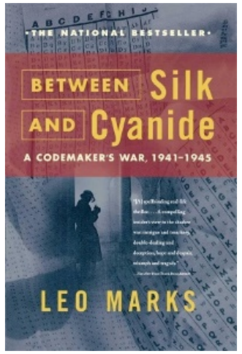
What are the downsides of the one-time-pad?

Key as long as the message (nowadays USBs contain several GB) and pre-shared! ← Moscow–Washington hotline

”Each country delivered keying tapes used to encode its messages via its embassy abroad”

https://en.wikipedia.org/wiki/Moscow%E2%80%93Washington_hotline

Key **cannot** be reused

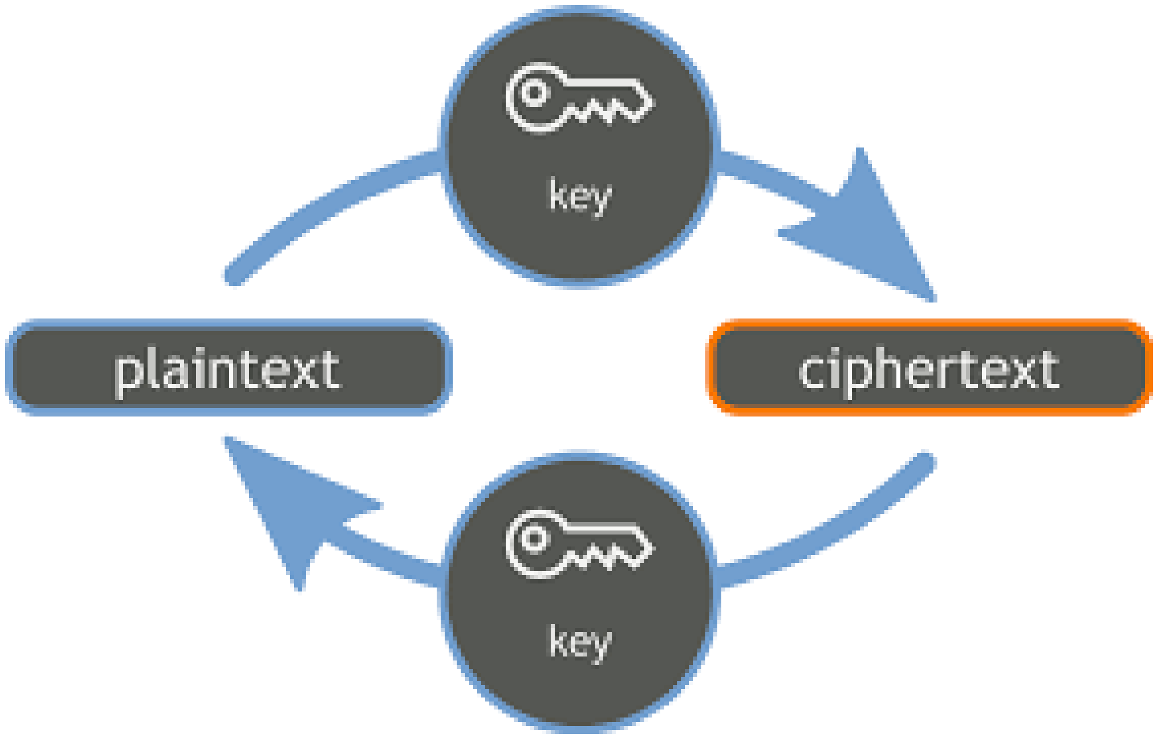


Key **must** be random!

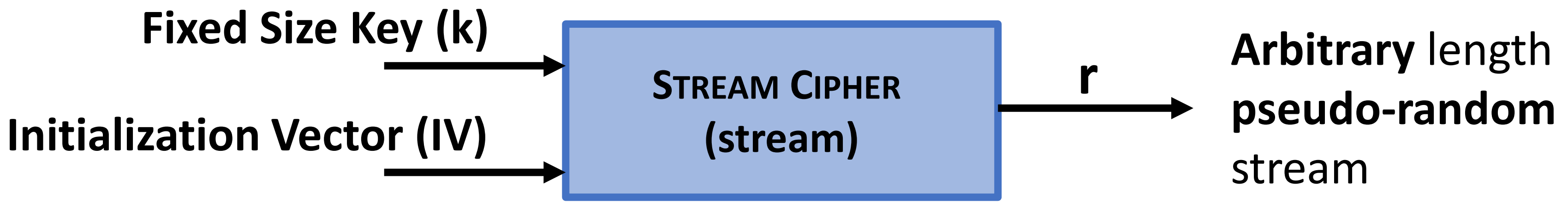
No integrity

Symmetric encryption

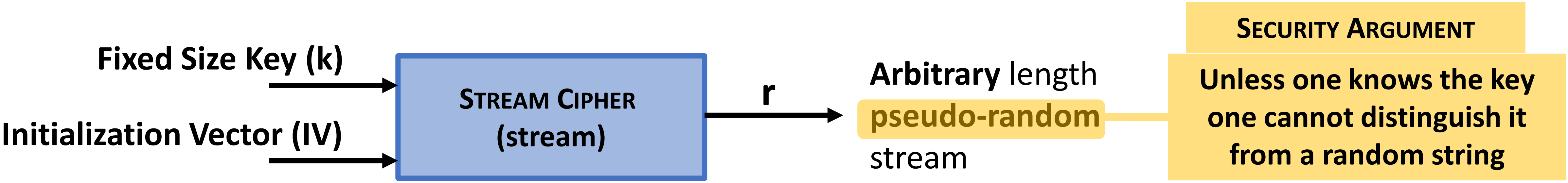
Encryption of plaintext and decryption of ciphertext are done using **THE SAME KEY**



Stream Ciphers: a cheap infinite OTP



Stream Ciphers: a cheap infinite OTP



Fixed Size Key (k)

Initialization Vector (IV)

STREAM CIPHER (stream)

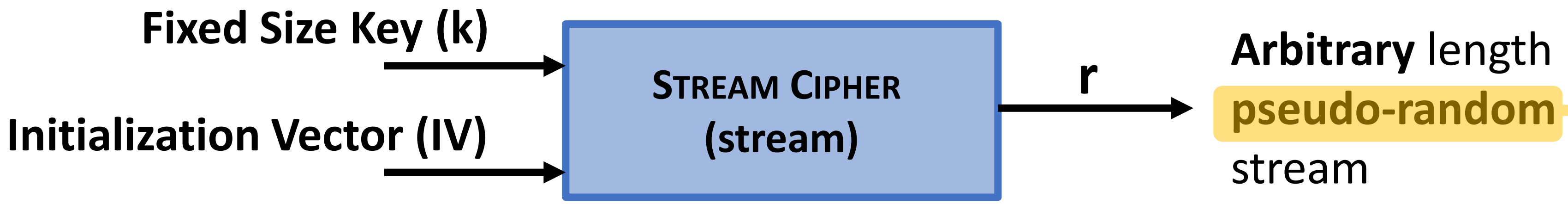
r

Arbitrary length **pseudo-random** stream

SECURITY ARGUMENT

Unless one knows the key one cannot distinguish it from a random string

Stream Ciphers: a cheap infinite OTP



SECURITY ARGUMENT

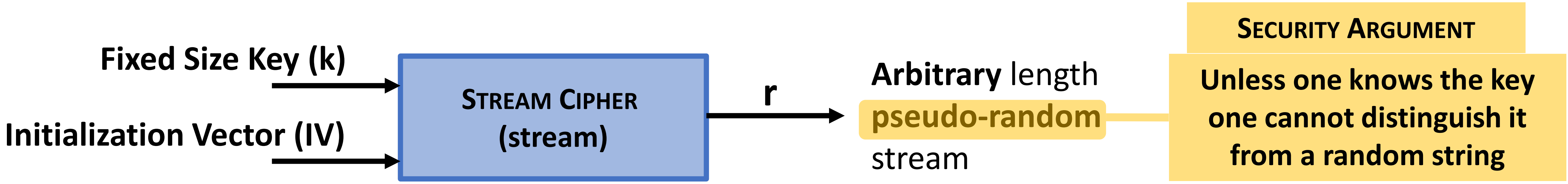
Unless one knows the key one cannot distinguish it from a random string

Key **k** ← pre-shared → Key **k**

↑

Fresh IV (**public**)

Stream Ciphers: a cheap infinite OTP



Fixed Size Key (k)

Initialization Vector (IV)

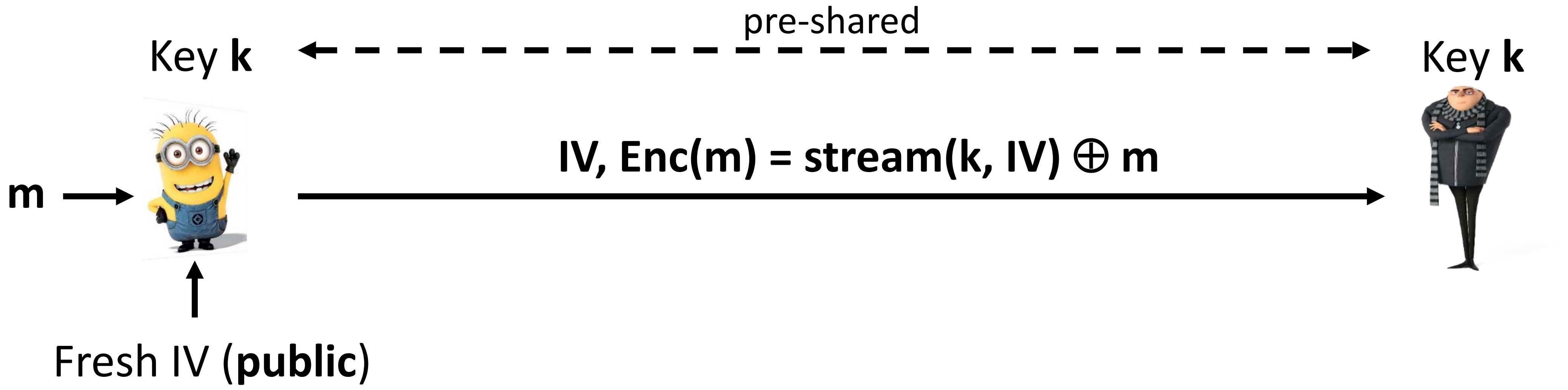
STREAM CIPHER (stream)

r

Arbitrary length **pseudo-random** stream

SECURITY ARGUMENT

Unless one knows the key one cannot distinguish it from a random string



pre-shared

Key **k**

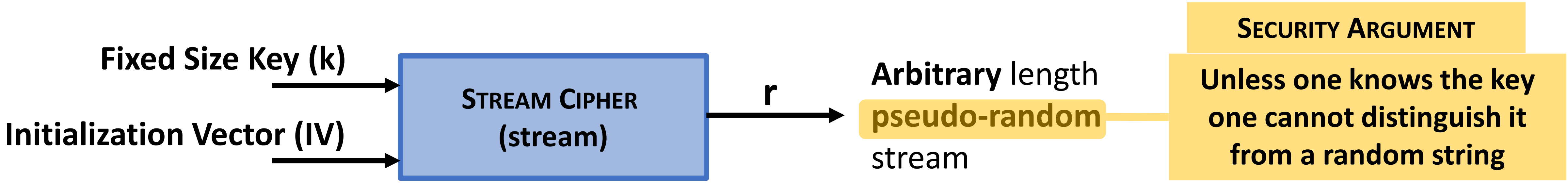
Key **k**

m

IV, Enc(m) = stream(k, IV) ⊕ m

Fresh IV (**public**)

Stream Ciphers: a cheap infinite OTP



Fixed Size Key (k)

Initialization Vector (IV)

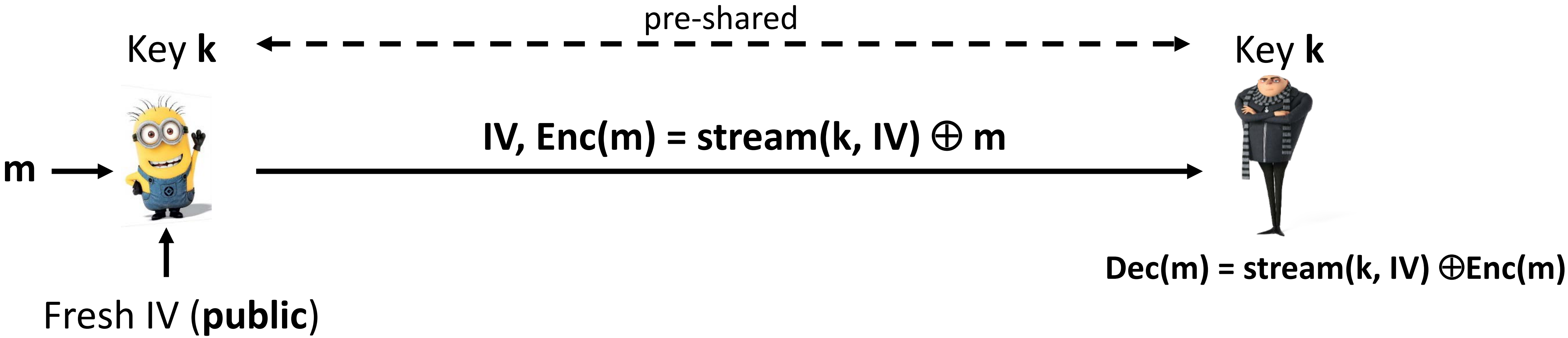
STREAM CIPHER
(stream)

r

Arbitrary length
pseudo-random
stream

SECURITY ARGUMENT

Unless one knows the key one cannot distinguish it from a random string



pre-shared

Key **k**

Key **k**

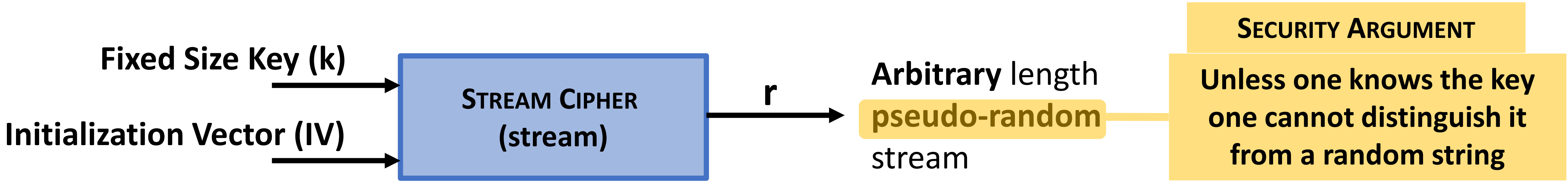
m

IV, Enc(m) = stream(k, IV) ⊕ m

Fresh IV (**public**)

Dec(m) = stream(k, IV) ⊕Enc(m)

Stream Ciphers: a cheap infinite OTP



Fixed Size Key (k)

Initialization Vector (IV)

STREAM CIPHER (stream)

r

Arbitrary length **pseudo-random** stream

SECURITY ARGUMENT

Unless one knows the key one cannot distinguish it from a random string

pre-shared

Key **k**

m

IV, Enc(m) = stream(k, IV) $\oplus$ m

Key **k**

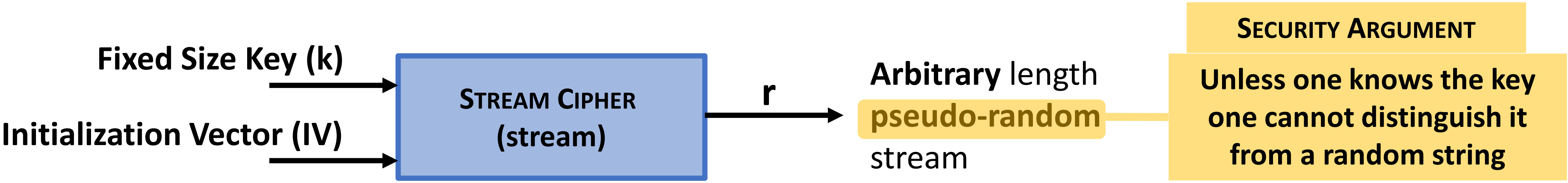
Dec(m) = stream(k, IV) $\oplus$Enc(m)

Fresh IV (**public**)

Remaining downsides?

- Key as long as the message (nowadays USBs contain several GB) and pre-shared!
- Key **cannot** be reused
- Key **must** be random!
- No integrity

Stream Ciphers: a cheap infinite OTP



Fixed Size Key (k)

Initialization Vector (IV)

STREAM CIPHER (stream)

r

Arbitrary length **pseudo-random** stream

SECURITY ARGUMENT

Unless one knows the key one cannot distinguish it from a random string

Key **k** ← pre-shared → Key **k**

m →

IV, Enc(m) = stream(k, IV) ⊕ m

Fresh IV (**public**)

Dec(m) = stream(k, IV) ⊕Enc(m)

Remaining downsides?

- ~~Key as long as the message (nowadays USBs contain several GB)~~
- and pre-shared!
- ~~Key **cannot** be reused~~
- Key **must** be random!
- No integrity

Better than before, though still necessary

What is an Initialization Vector?

- Fixed-size input to iterative cryptographic primitives to start the process

- Has to be **unique** (for a key): no IV may be reused under the same key
  Multiple messages (even same!) encrypted with the same key look different

- It **must be random**! If it is predictable, it gives advantage to the adversary

- It **does not need to be secret**! Keeping the key secret is enough

Stream ciphers

STRENGTHS

Speed of transformation: algorithms are linear in time and constant in space

Low error propagation: errors in one bit do not affect subsequent symbols

WEAKNESSES

Low diffusion: all information of a plaintext symbol is contained in one encrypted symbol

Susceptibility to insertions/ modifications: text can be inserted, difficult to detect

***Trivium* (80 bit key, < 4000 gates in HW)**

***Salsa20* (128/256 bit key, Random access)**

More stream ciphers: https://en.wikipedia.org/wiki/ESTREAM

Stream ciphers

STRENGTHS

Speed of transf ant in space

Low error prop t symbols

WEAKNESSES

Low diffusion: a one encrypted symbol

Susceptibility to fficult to detect

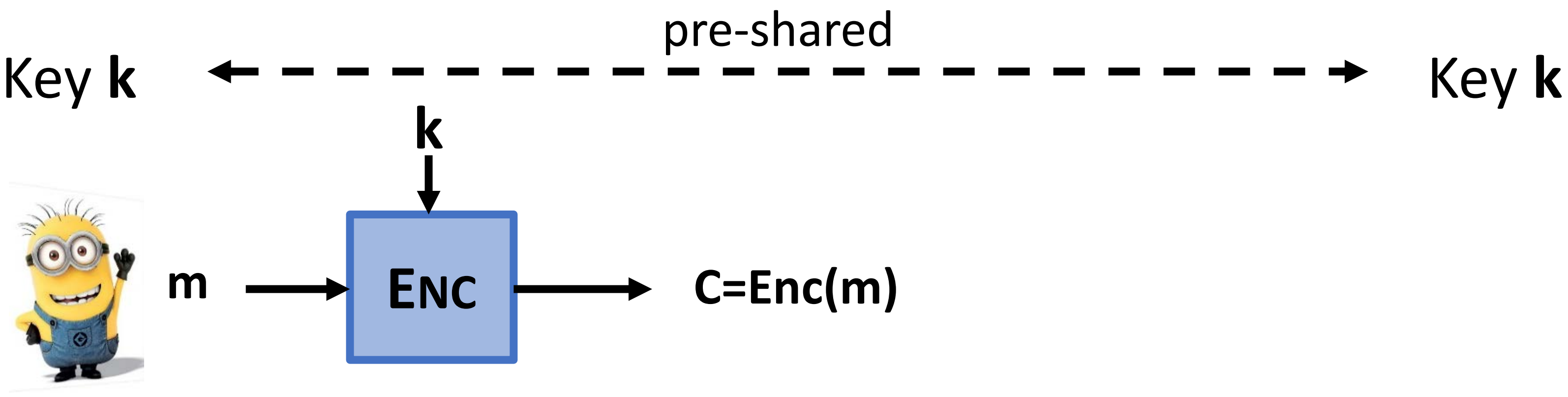
Don’t design your own

Trivium **(80 bit key, < 4000 gates in HW)**

Salsa20 **(128/256 bit key, Random access)**

More stream ciphers: https://en.wikipedia.org/wiki/ESTREAM

Block Ciphers

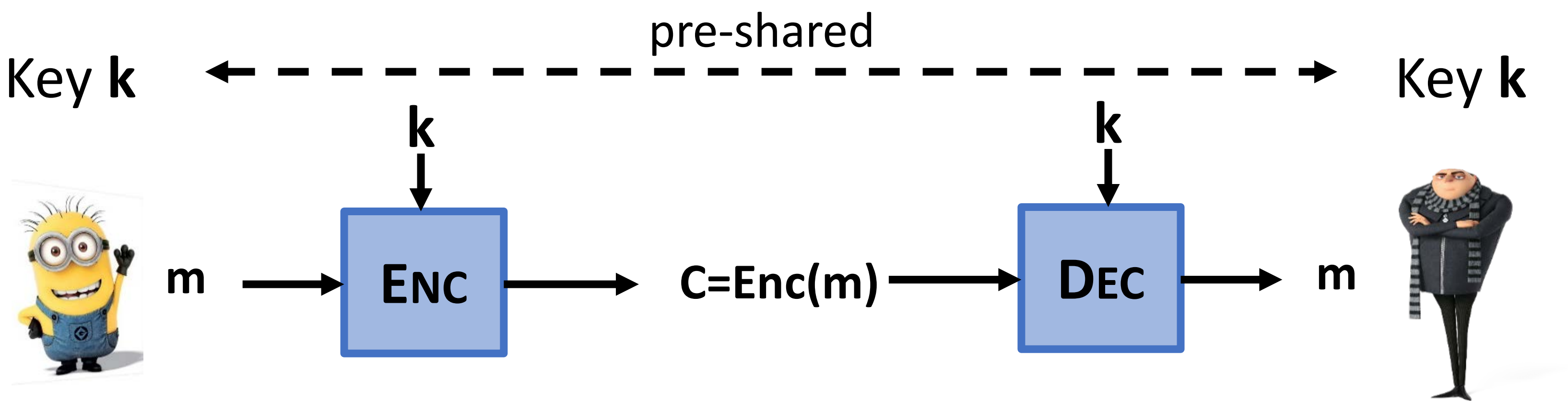


- **Key** k: short random string (e.g. 128 bits)
- **Plaintext (m) / Ciphertext (C)**: short blocks (e.g. 128 bits)

SECURITY ARGUMENT

Without k: same as a random block

Block Ciphers



- **Key** k: short random string (e.g. 128 bits)
- **Plaintext (m) / Ciphertext (C)**: short blocks (e.g. 128 bits)

SECURITY ARGUMENT

Without k: same as a random block

- Encryption algorithm != Decryption algorithm

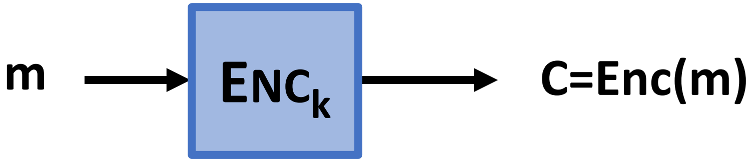
Dec is inverse of Enc → Dec(k; Enc(k; m)) = m

Block Ciphers: modes of operation

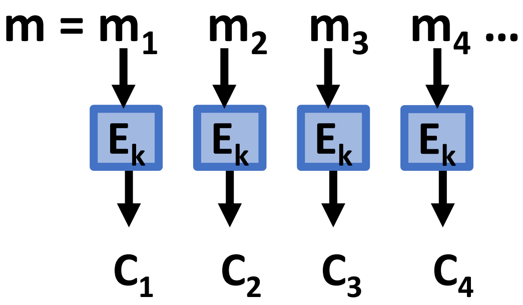
Messages are longer than a block!

Electronic Book Code (ECB)

Straightforward scheme: encrypt & decrypt single blocks



m $\rightarrow$ ENC$_k$ $\rightarrow$ C=Enc(m)



m = m_1 m_2 m_3 m_4 …

E_k E_k E_k E_k

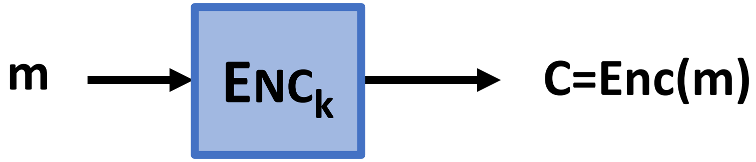
C_1 C_2 C_3 C_4

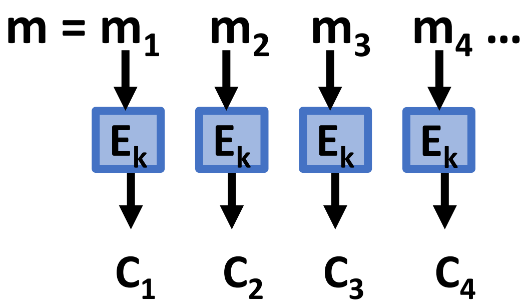
Block Ciphers: modes of operation

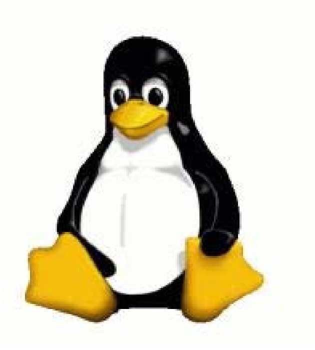
Messages are longer than a block!

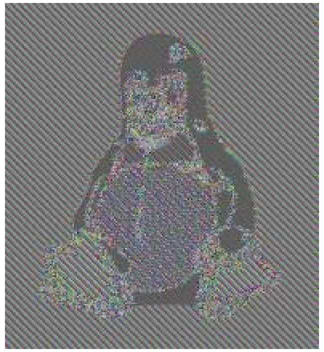
Electronic Book Code (ECB)

Straightforward scheme: encrypt & decrypt single blocks




$m = m_1 \quad m_2 \quad m_3 \quad m_4$ …
E_k
E_k
E_k
E_k
$C_1 \quad C_2 \quad C_3 \quad C_4$





Problematic! $m_1=m_2 \rightarrow C_1=C_2$ **DON'T USE!!**

https://en.wikipedia.org/wiki/Block_cipher#/media/File:Tux_ecb.jpg

Block Ciphers: modes of operation

Messages are longer than a block!

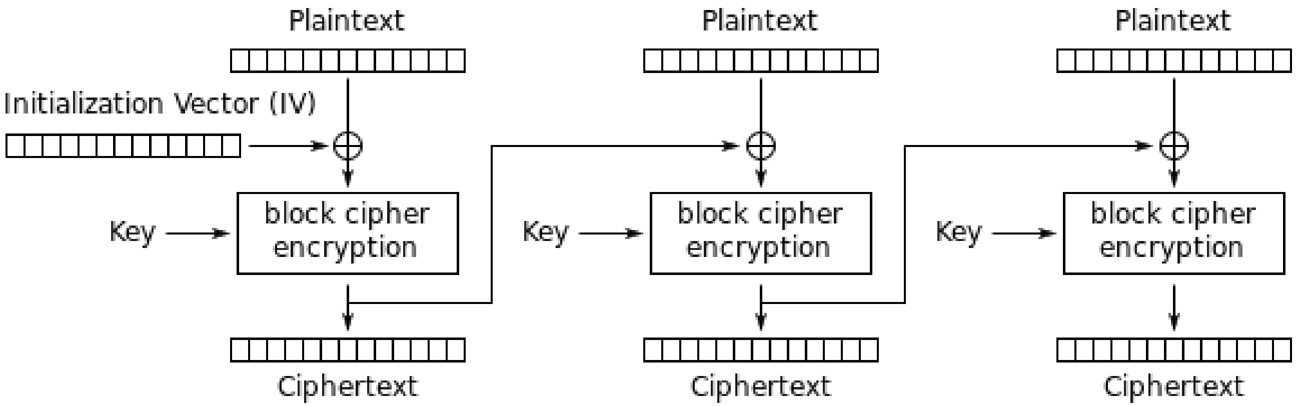
CIPHER BLOCK CHAINING (CBC)

Propagating information across blocks

Encryption

$C_0 = IV$

$C_i = Enc(k; m_i \oplus C_{i-1})$



https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#/media/File:CBC_encryption.svg

Block Ciphers: modes of operation

Messages are longer than a block!

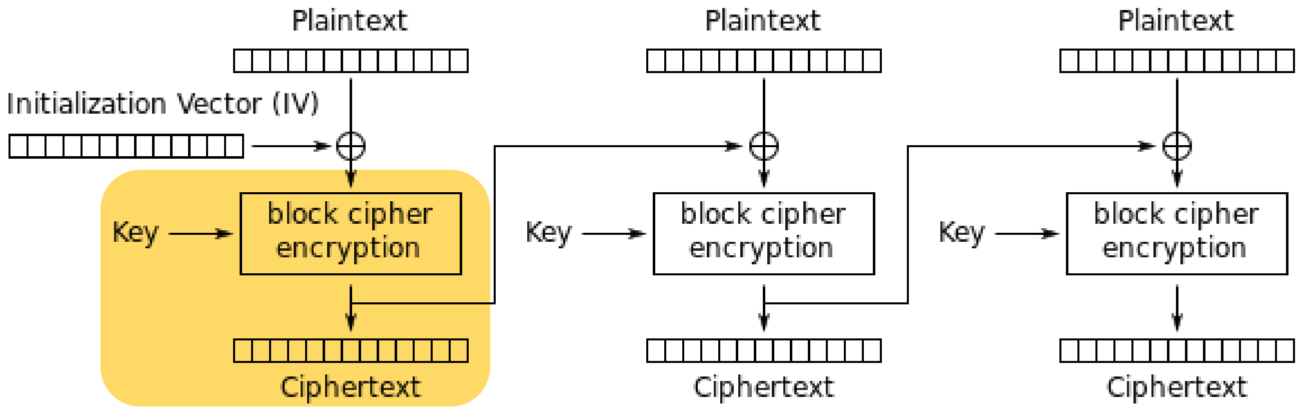
Cipher Block Chaining (CBC)

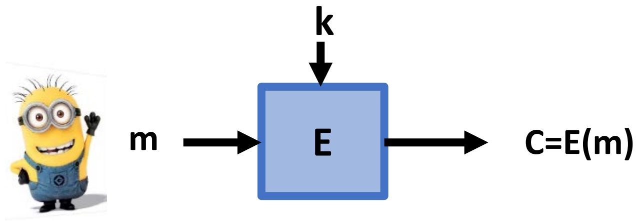
Propagating information across blocks

Encryption

$C_0 = IV$

$C_i = Enc(k; m_i \oplus C_{i-1})$





https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#/media/File:CBC_encryption.svg

Block Ciphers: modes of operation

Messages are longer than a block!

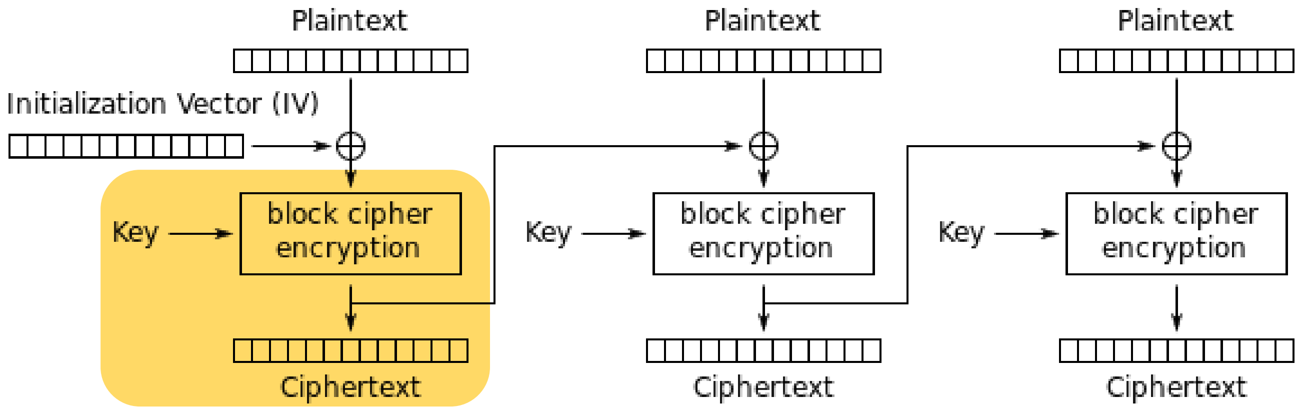
Cipher Block Chaining (CBC)

Propagating information across blocks

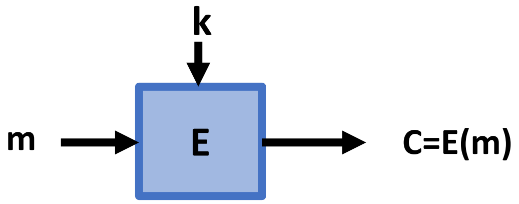
Encryption

$C_0 = IV$

$C_i = Enc(k; m_i \oplus C_{i-1})$



Decryption??



https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#/media/File:CBC_encryption.svg

Block Ciphers: modes of operation

Messages are longer than a block!

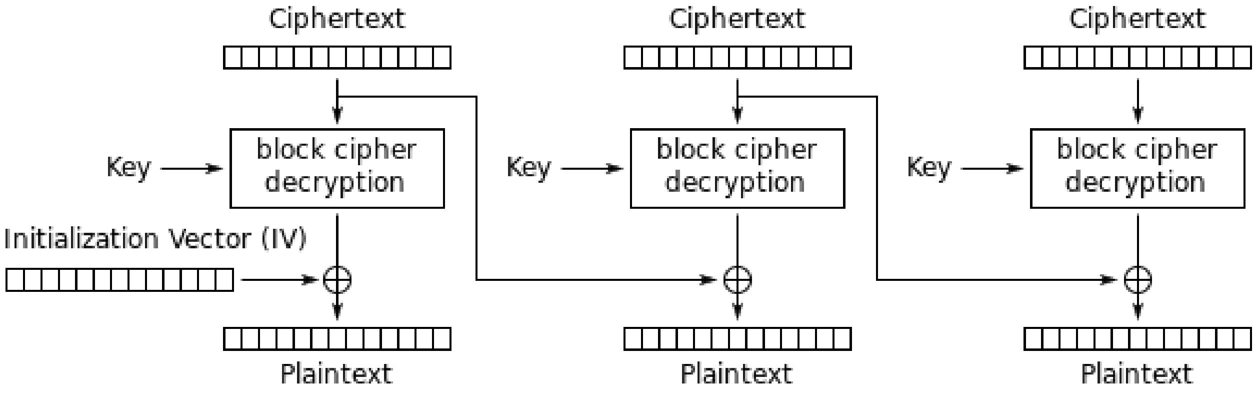
Cipher Block Chaining (CBC)

Propagating information across blocks

Decryption

$C_0 = IV$

$m_i = Dec(k; C_i)$ XOR C_{i-1}



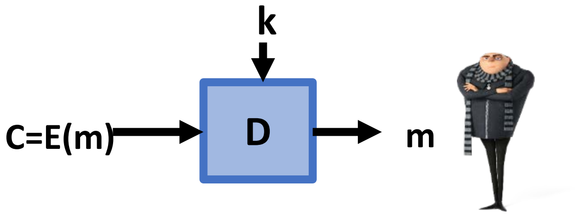
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#/media/File:CBC_decryption.svg

Block Ciphers: modes of operation

Messages are longer than a block!

Cipher Block Chaining (CBC)

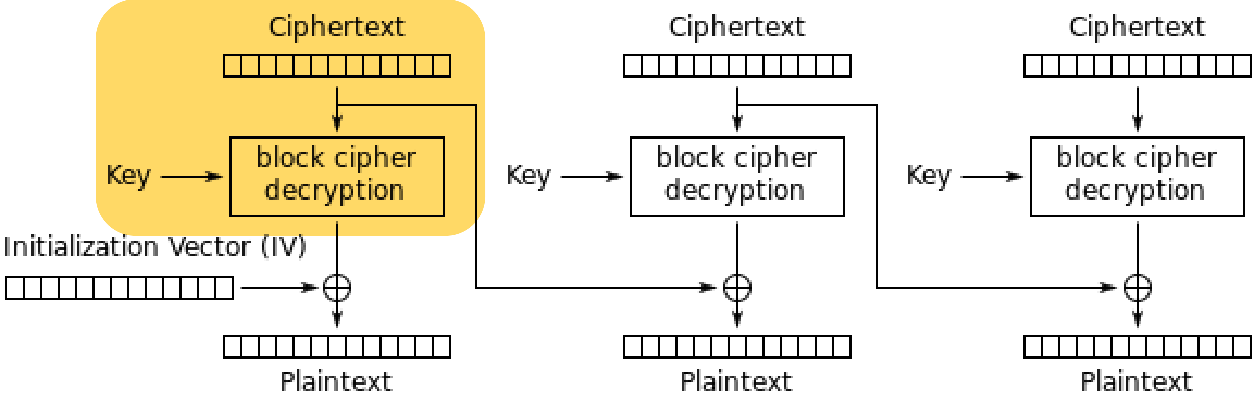
Propagating information across blocks



Decryption

$C_0 = IV$

$m_i = Dec(k; C_i) \text{ XOR } C_{i-1}$



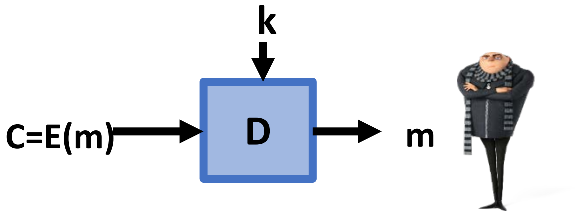
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#/media/File:CBC_decryption.svg

Block Ciphers: modes of operation

Messages are longer than a block!

Cipher Block Chaining (CBC)

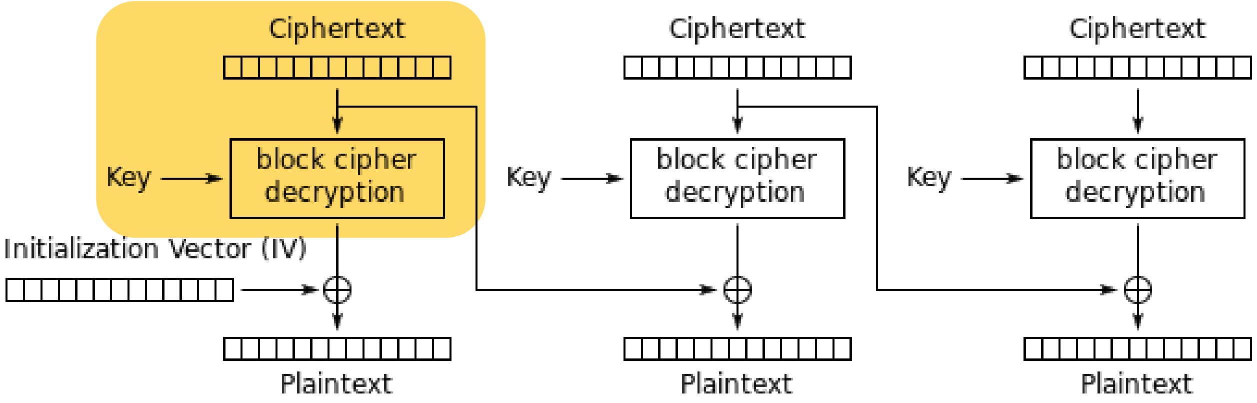
Propagating information across blocks



Decryption

$C_0 = IV$

$m_i = Dec(k; C_i) \text{ XOR } C_{i-1}$



What if IV is incorrect? The full decryption is wrong?

Can you decrypt a block alone? What do you need?

https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#/media/File:CBC_decryption.svg

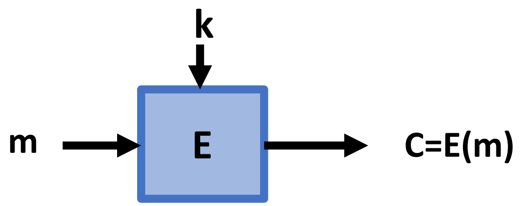
Block Ciphers: modes of operation

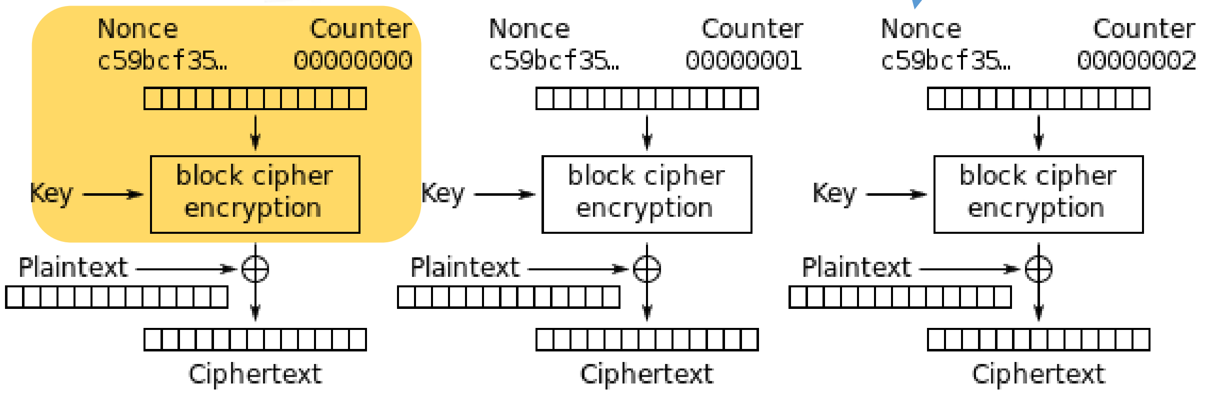
Messages are longer than a block!

Nonce = number used only once

Counter Mode (CTR)

Turning a block cipher into a stream cipher





Encryption

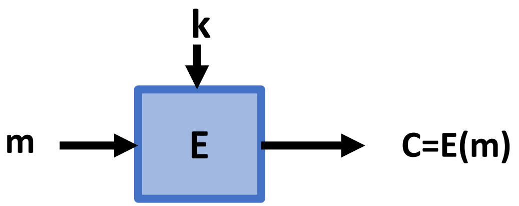
$$C_i = \mathrm{Enc}(k; IV+i) \oplus m_i$$

Block Ciphers: modes of operation

Messages are longer than a block!

Counter Mode (CTR)

Turning a block cipher into a stream cipher

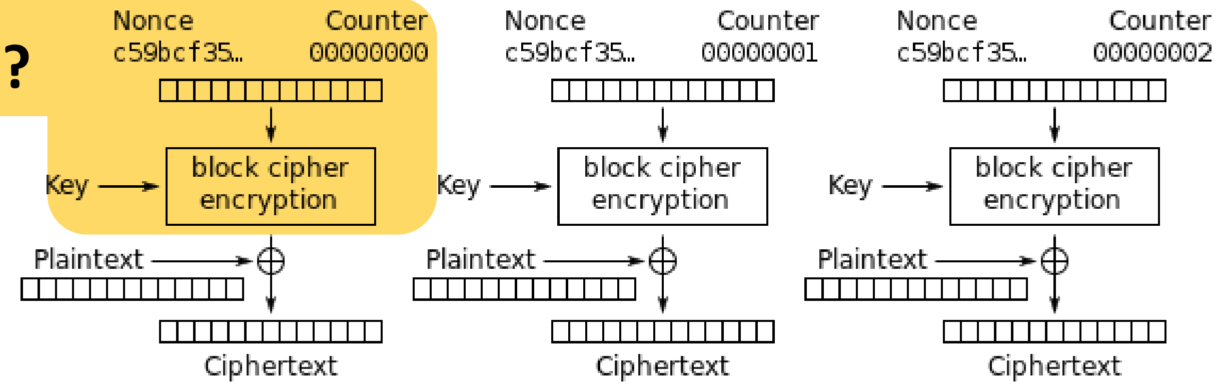


Decryption??

Do we need D?

Encryption

$$C_i = \text{Enc}(k; IV+i) \oplus m_i$$



Block Ciphers: modes of operation

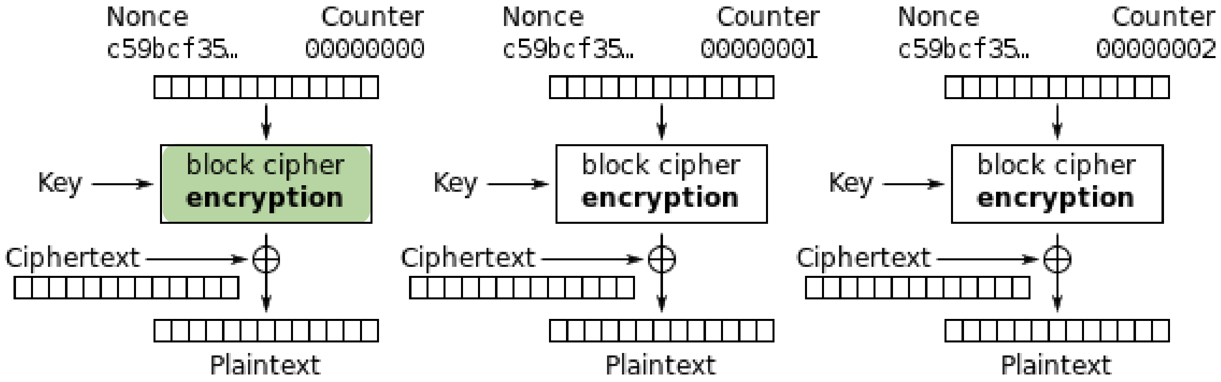
Messages are longer than a block!

Counter Mode (CTR)

Turning a block cipher into a stream cipher

Decryption

$$m_i = \mathbf{Enc}(k; IV+i) \oplus C_i$$



Check at home: Output Feedback Mode (OFB)

Block ciphers

STRENGTHS

High diffusion: information from one plaintext symbol is diffused into several ciphertext symbols

Immunity to tampering: difficult to insert symbols without detection

WEAKNESSES

Slowness of encryption: an entire block must be accumulated before encryption / decryption can begin

Error propagation: in some modes of operation errors affect several bits/blocks

***AES* – The Advanced Encryption Standard**
128/256 bit key, NIST Standard, HW support

More: https://en.wikipedia.org/wiki/Block_cipher#Notable_block_ciphers

Block ciphers

Strengths

High diffusi ed into several ciphertext sy

Immunity to ection

Weaknesses

Slowness of efore encryption / decryption c

Error propagation: difficult to insert symbols without detection

Don’t design your own

***AES* – The Advanced Encryption Standard**
128/256 bit key, NIST Standard, HW support

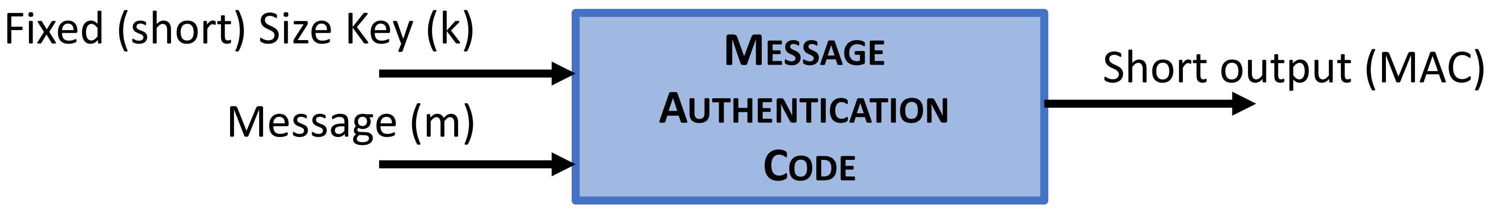
More: https://en.wikipedia.org/wiki/Block_cipher#Notable_block_ciphers

Message Authentication Codes (MAC)

Dealing with integrity

Message Authentication Codes (MAC)

Dealing with integrity

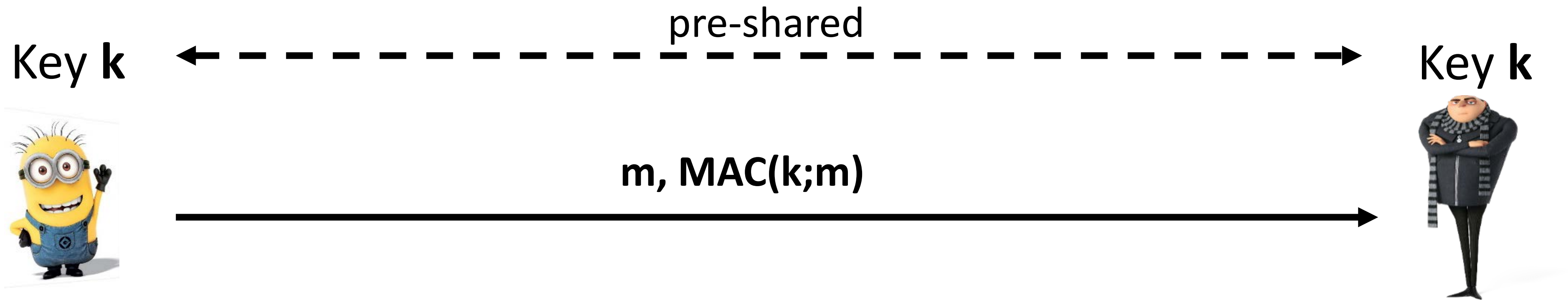


Fixed (short) Size Key (k)

Message (m)

Message Authentication Code

Short output (MAC)



pre-shared

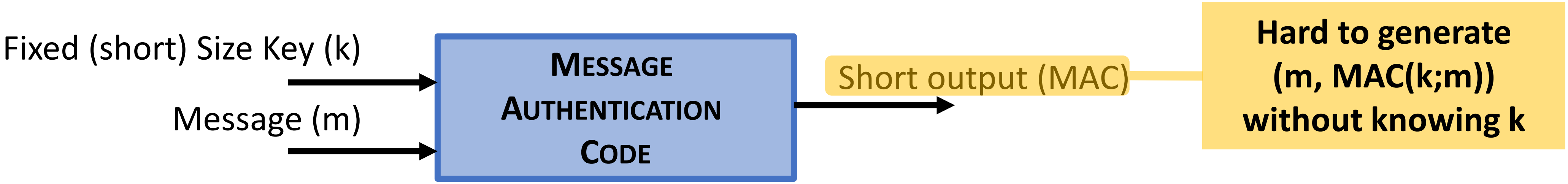
Key **k**

Key **k**

m, MAC(k;m)

Message Authentication Codes (MAC)

Dealing with integrity



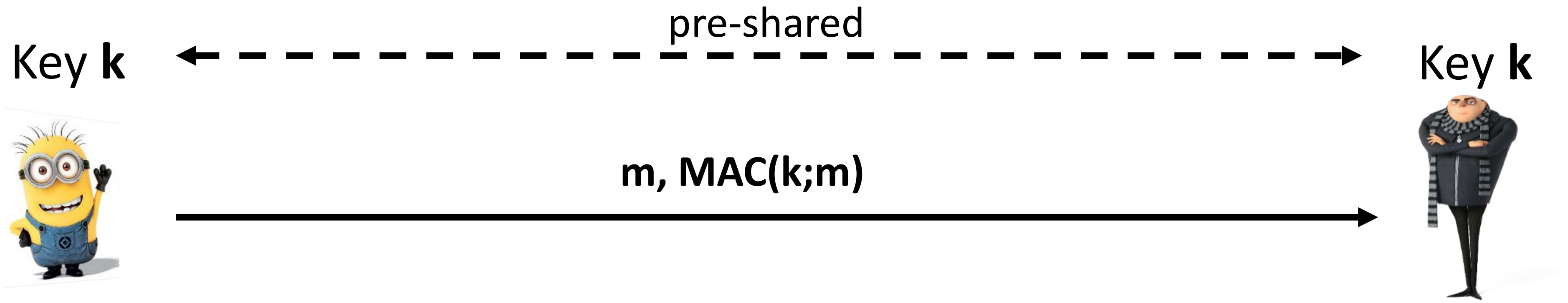
Fixed (short) Size Key (k)

Message (m)

MESSAGE AUTHENTICATION CODE

Short output (MAC)

Hard to generate (m, MAC(k;m)) without knowing k



Key **k**

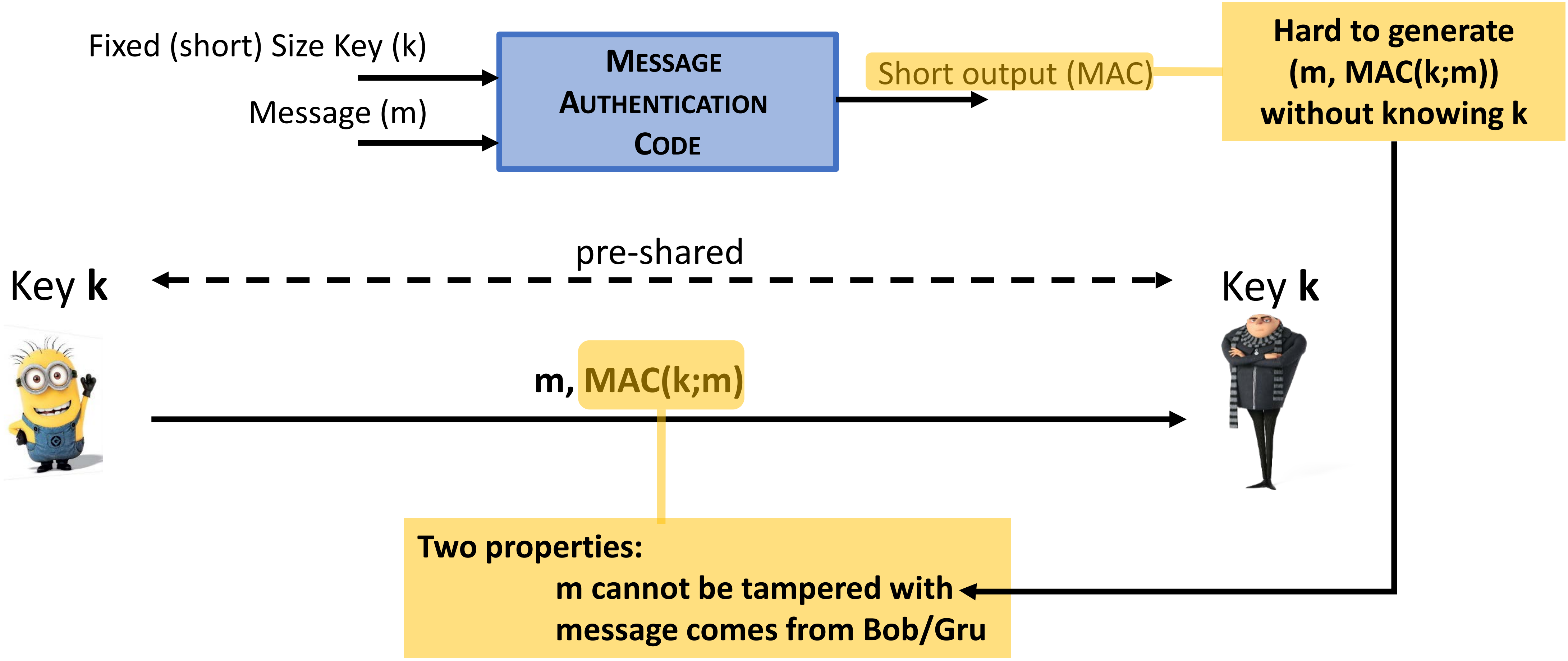
pre-shared

Key **k**

m, MAC(k;m)

Message Authentication Codes (MAC)

Dealing with integrity



Fixed (short) Size Key (k)

Message (m)

MESSAGE AUTHENTICATION CODE

Short output (MAC)

Hard to generate (m, MAC(k;m)) without knowing k

pre-shared

Key **k**

Key **k**

m, MAC(k;m)

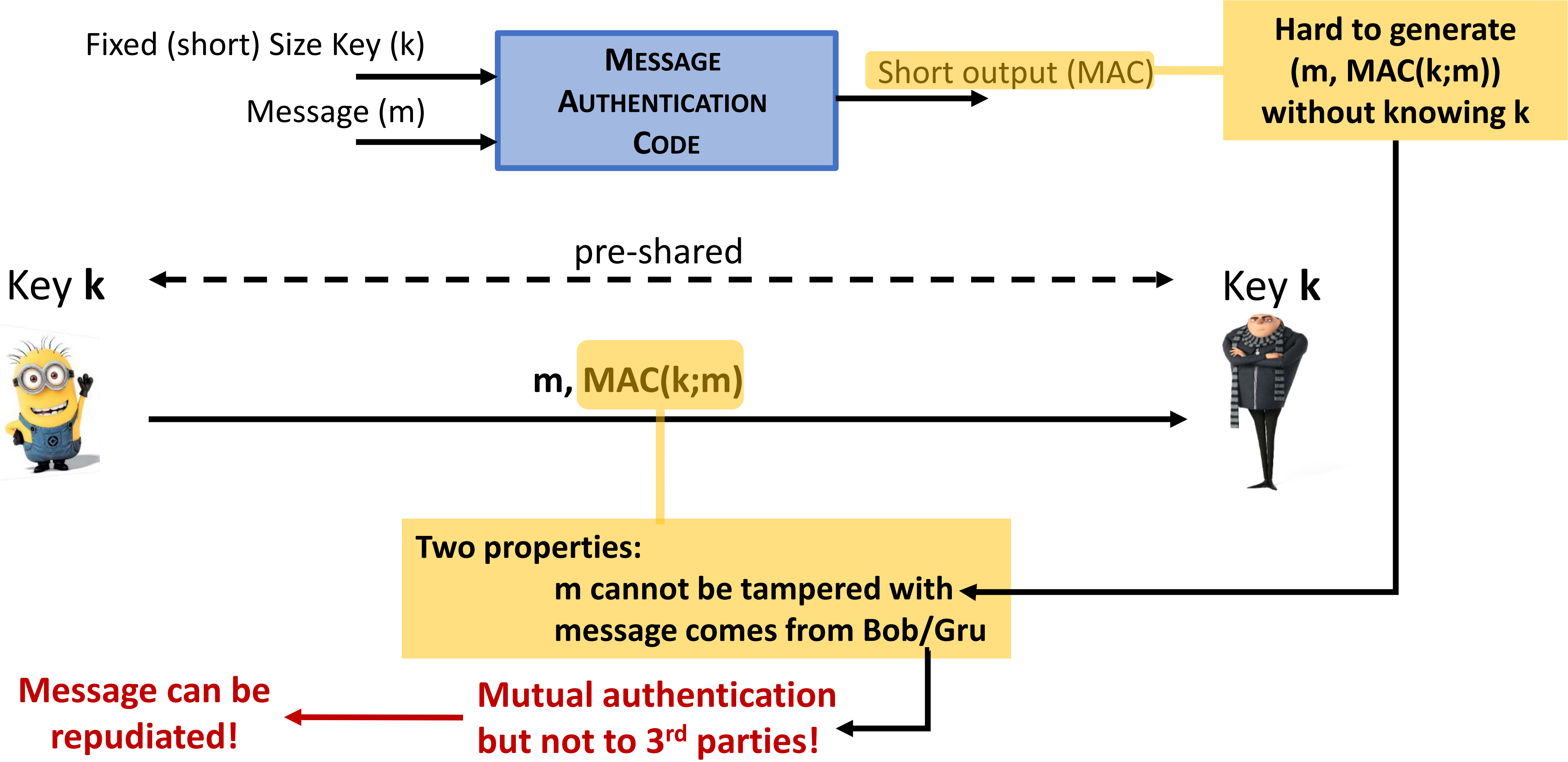
Two properties:

m cannot be tampered with

message comes from Bob/Gru

Message Authentication Codes (MAC)

Dealing with integrity



Fixed (short) Size Key (k)

Message (m)

MESSAGE AUTHENTICATION CODE

Short output (MAC)

Hard to generate (m, MAC(k;m)) without knowing k

pre-shared

Key **k**

Key **k**

m, MAC(k;m)

Two properties:

m cannot be tampered with

message comes from Bob/Gru

Mutual authentication but not to 3rd parties!

Message can be repudiated!

Message Authentication Codes (MAC)

How to turn a block cipher into a MAC

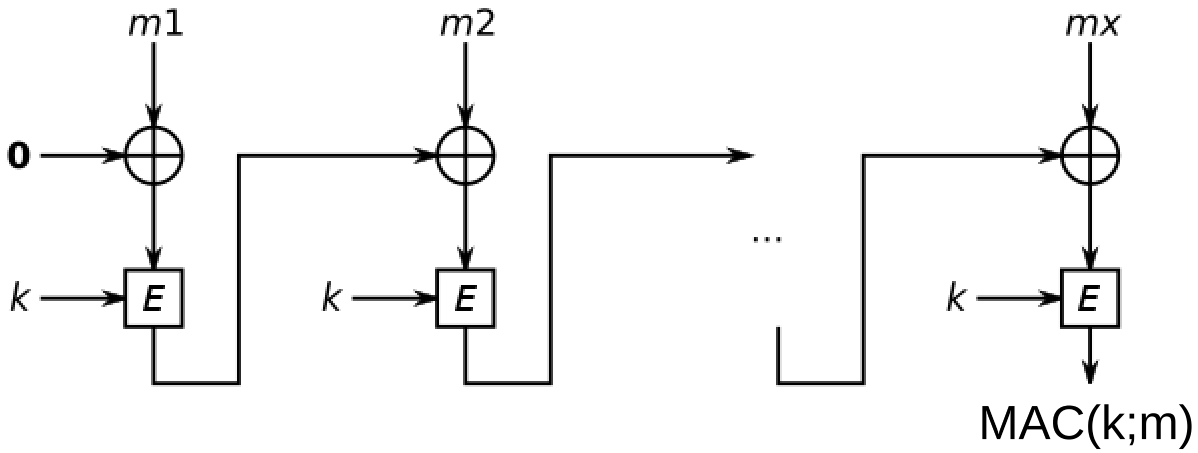
CBC-MAC

Turning a block cipher into a MAC

$C_0 = 0$ **[any fixed IV]**

$C_i = \mathrm{Enc}(k; m_i \oplus C_{i-1})$

$\mathrm{MAC}(k; m_1 \ldots m_x) = C_n$



Message Authentication Codes (MAC)

How to turn a block cipher into a MAC

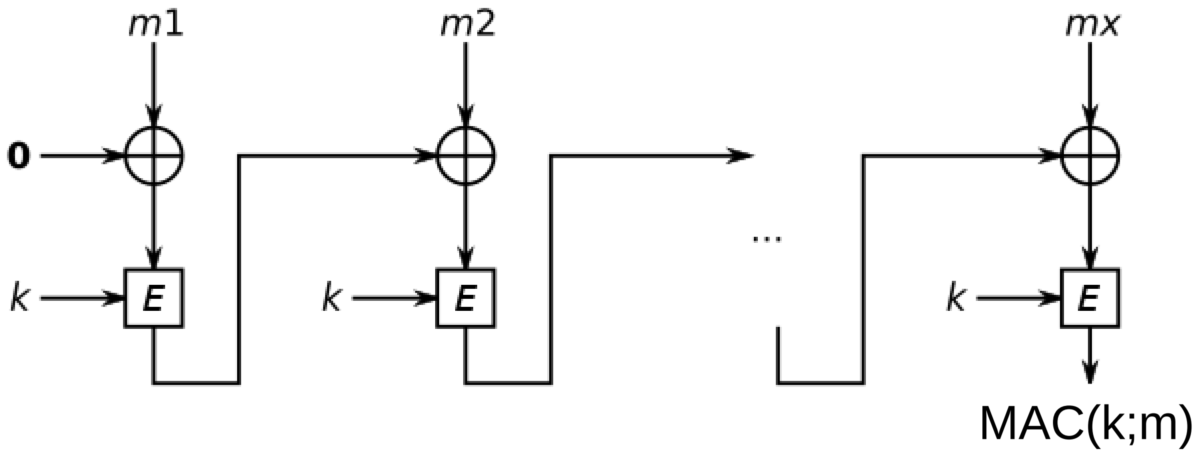
CBC-MAC

Turning a block cipher into a MAC

$C_0 = 0$ **[any fixed IV]**

$C_i = \mathrm{Enc}(k; m_i \oplus C_{i-1})$

$\mathrm{MAC}(k; m_1 \ldots m_x) = C_n$



Differences from CBC

CBC-MAC is deterministic

Only output is the final value!

Message Authentication Codes (MAC)

How to turn a block cipher into a MAC

Limitation:

Only secure if the length of m is known!

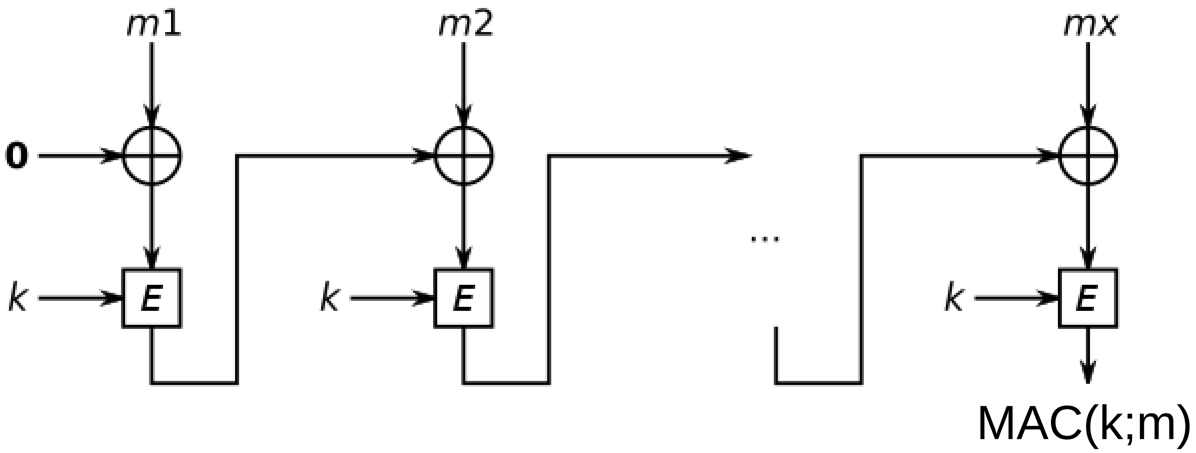
CBC-MAC

Turning a block cipher into a MAC

$C_0 = 0$ **[any fixed IV]**

$C_i = \mathrm{Enc}(k; m_i \oplus C_{i-1})$

$\mathrm{MAC}(k; m_1 \ldots m_x) = C_n$

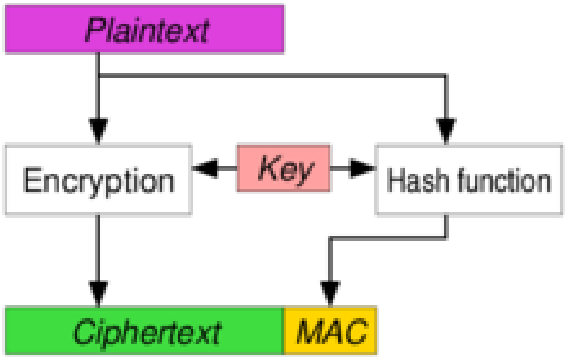


Differences from CBC

- **CBC-MAC** is deterministic
- Only output is the final value!

How to combine confidentiality and integrity?

ENCRYPT-AND-MAC



- ✗ No integrity on the ciphertext → Cipher can be attacked need to decrypt to know if valid

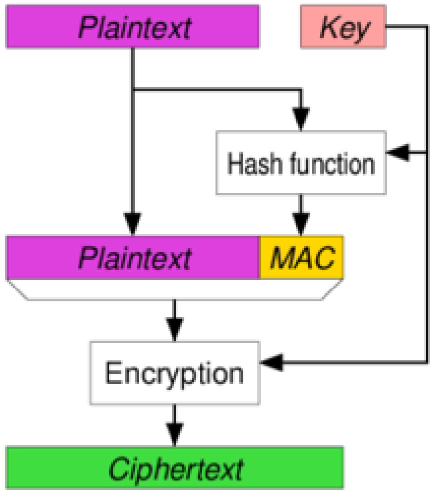
- ✓ Integrity of the plaintext can be verified

- ✗ May reveal information about the plaintext → repeated msg, recall the IV is fixed (can be solved with a counter)

Bellare, M., & Namprempre, C. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *International Conference on the Theory and Application of Cryptology and Information Security*, 2000.
Bellare, M., Kohno, T., & Namprempre, C. Breaking and provably repairing the SSH authenticated encryption scheme: A case study of the Encode-then-Encrypt-and-MAC paradigm. *ACM Transactions on Information and System Security*, 2004.
https://en.wikipedia.org/wiki/Authenticated_encryption

How to combine confidentiality and integrity?

Mac-then-Encrypt



✗ No integrity of ciphertext

(in theory) possible to change ciphertext and have a valid MAC

need to decrypt to know if valid

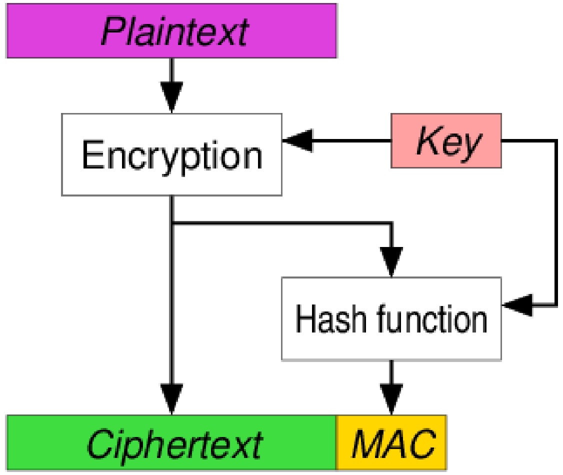
✓ Integrity of the plaintext can be verified

✓ No information on the plaintext either, since it is encrypted

Bellare, M., & Namprempre, C. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *International Conference on the Theory and Application of Cryptology and Information Security*, 2000.
Bellare, M., Kohno, T., & Namprempre, C. Breaking and provably repairing the SSH authenticated encryption scheme: A case study of the Encode-then-Encrypt-and-MAC paradigm. *ACM Transactions on Information and System Security*, 2004.
https://en.wikipedia.org/wiki/Authenticated_encryption

How to combine confidentiality and integrity?

Encrypt-then-Mac



✓ Integrity of ciphertext → ensures you only read valid messages! Cipher cannot be attacked!

✓ Integrity of the plaintext can be verified

✓ No information on the plaintext either, since it is encrypted

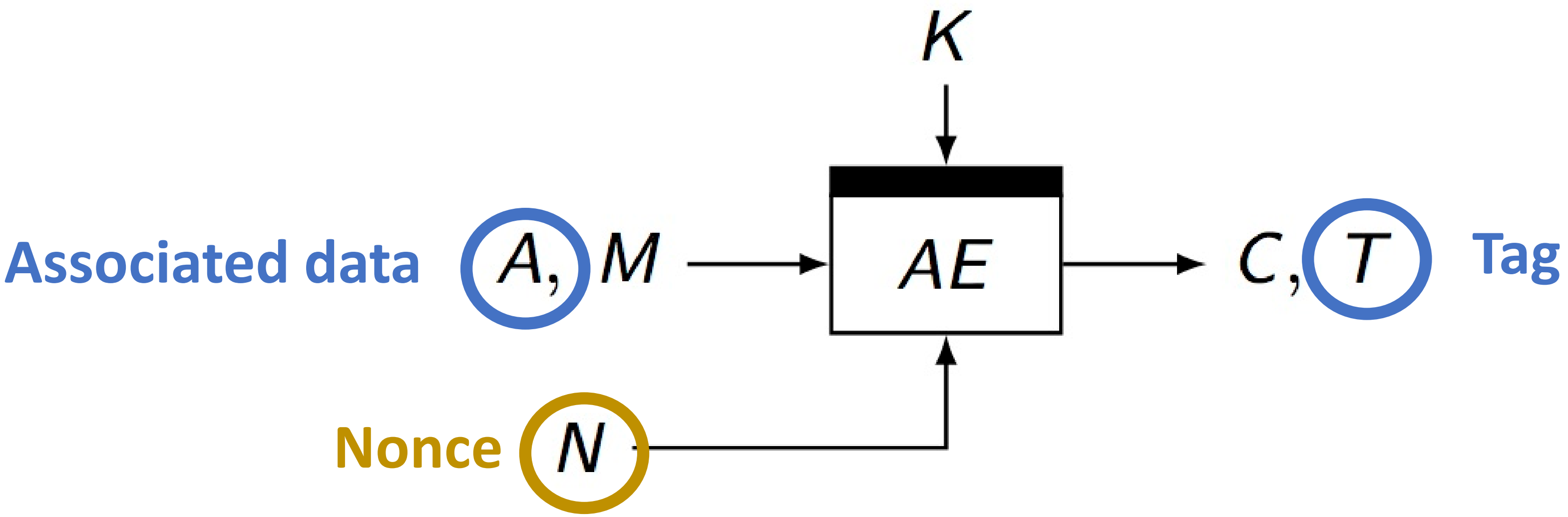
Bellare, M., & Namprempre, C. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *International Conference on the Theory and Application of Cryptology and Information Security*, 2000.

Bellare, M., Kohno, T., & Namprempre, C. Breaking and provably repairing the SSH authenticated encryption scheme: A case study of the Encode-then-Encrypt-and-MAC paradigm. *ACM Transactions on Information and System Security*, 2004.

https://en.wikipedia.org/wiki/Authenticated_encryption

Authenticated Encryption with Associated Data (AEAD)

Home-made crypto recipes are dangerous!!!



Galois counter mode - **GCM** (one pass)

Encrypt-then-authenticate-then-translate - **EAX** (Two passes)

https://www.fi.muni.cz/~xsvenda/docs/AE_comparison_ipics04.pdf

Asymmetric Cryptography

Block ciphers, Stream Ciphers, MACs:

Alice and Bob need to **share** a **secret** key

Secure key distribution is a problem!

If only we could have a public keys…

Diffie, Whitfield, and Martin Hellman. "New directions in cryptography.“ Information Theory, IEEE Transactions on 22.6 (1976): 644-654

Asymmetric cryptography

Secret Key: SK

Public Key: PK

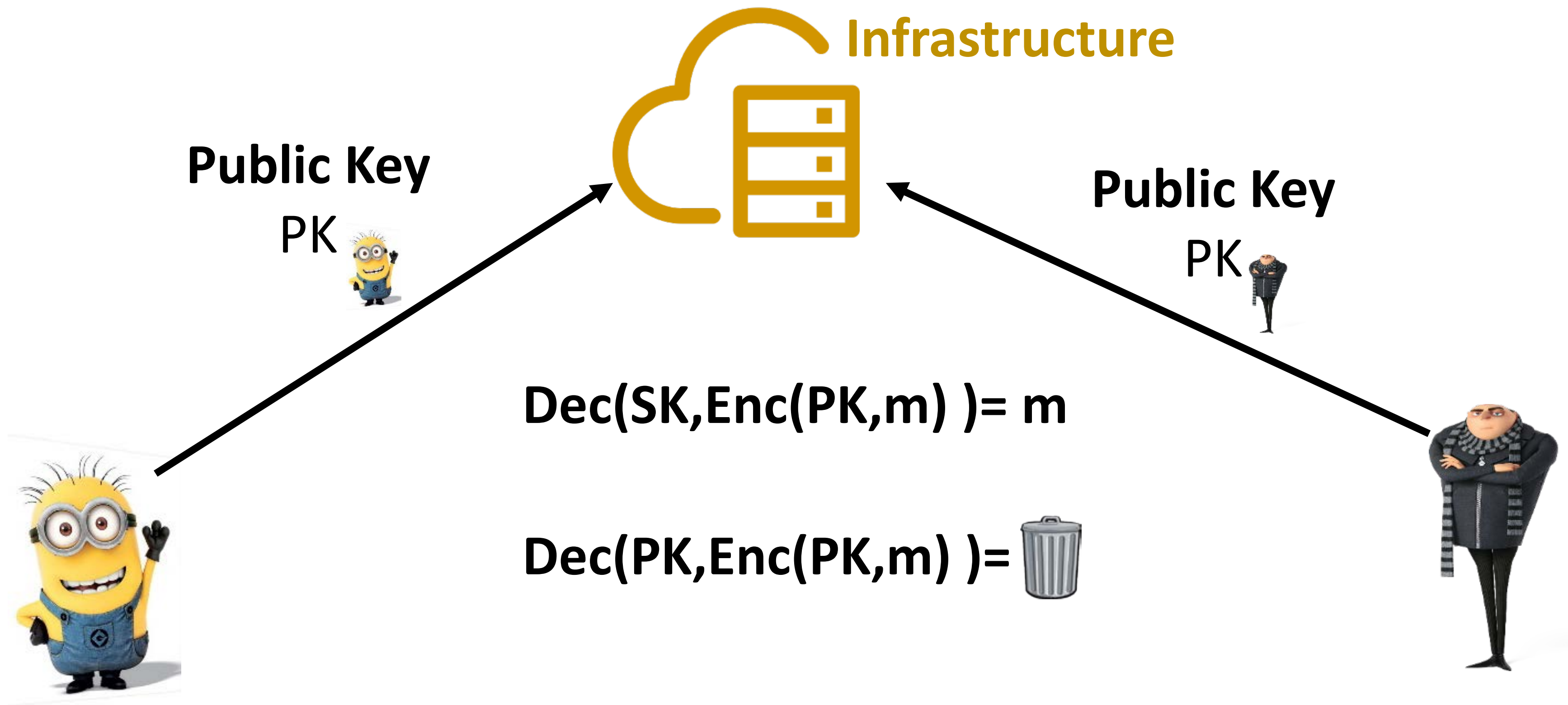
Dec(SK,Enc(PK,m))= m

Dec(PK,Enc(PK,m))=🗑

Secret Key: SK

Public Key: PK

Asymmetric cryptography



Dec(SK,Enc(PK,m))= m

Dec(PK,Enc(PK,m))= 🗑

Secret Key: SK

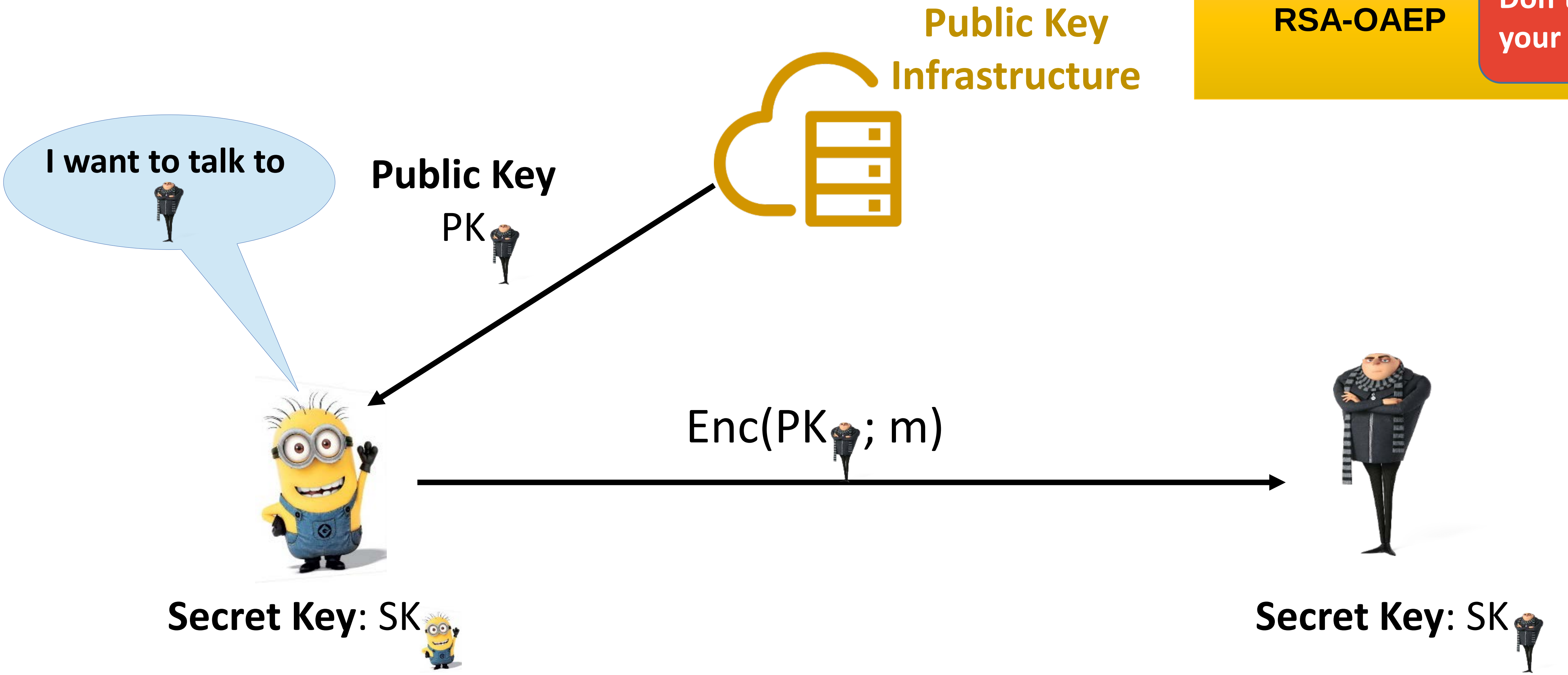
Secret Key: SK

Asymmetric cryptography

Examples:

RSA-OAEP

Don't design your own ⚠



Asymmetric cryptography

Examples:

RSA-OAEP

Don't design your own ⚠

Public Key Infrastructure

I want to talk to

Public Key

PK

Enc(PK ; m)

Secret Key: SK

Secret Key: SK

Who wrote the message?

Digital Signatures

Cannot "forge" a signature (m, s, PK) that verifies without knowing sk

Sign(SK,m)= s

Verify(PK, s)= Yes or No

Secret Key: SK

Public Key: PK

Secret Key: SK

Public Key: PK

Digital signatures

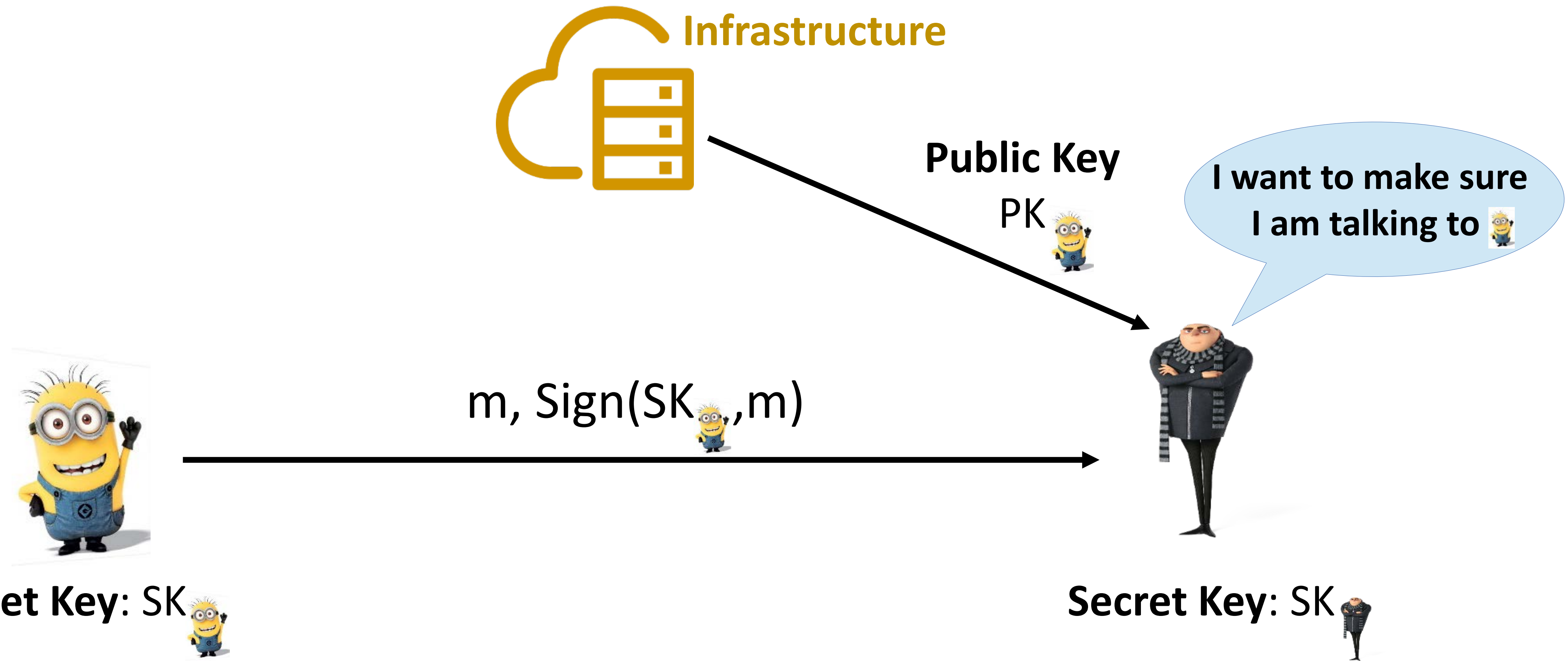
Examples:

NIST DSA

RSA-PSS

Don't design your own ⚠

Public Key Infrastructure



Public Key

PK

I want to make sure I am talking to

m, Sign(SK,m)

Secret Key: SK

Secret Key: SK

Digital Signatures

Properties:

Authenticity – of message and of sender!

Non-repudiation (why are they different from MACs?)

Applications:

PKI: Certificates

(1) Authority signs a mapping between names, or names and Encryption public keys.

(2) Authority signs mapping between names and Verification Keys.

Digital Signatures

Properties:

Authenticity – of message and of sender!

Non-repudiation (why are they different from MACs?)

Applications:

PKI: Certificates

(1) Authority signs a mapping between names, or names and Encryption public keys.

(2) Authority signs mapping between names and Verification Keys.

Encryption key pair != Signature key pair ⚠

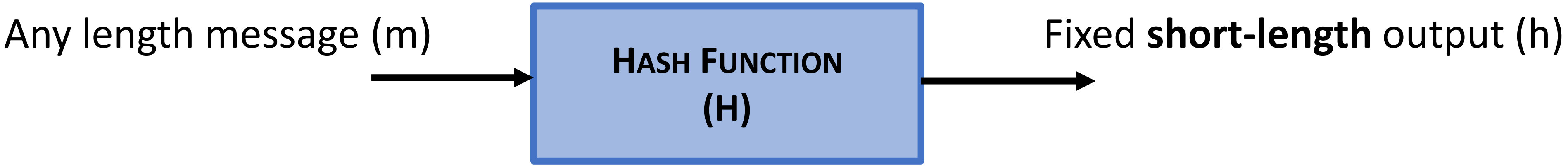
Asymmetric cryptography limitations

Computationally costly compared with most symmetric key algorithms of equivalent security

Signing and encrypting **is slow**

Not suitable to encrypt **large amounts of data**

Hash functions

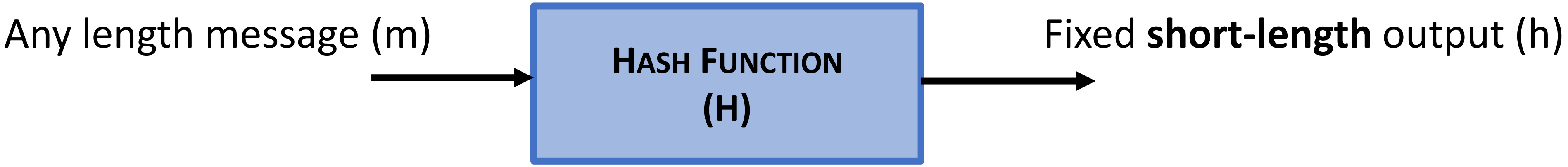


Any length message (m)

HASH FUNCTION
(H)

Fixed **short-length** output (h)

Hash functions



Three security properties

Pre-image resistance

Given H(m), difficult to get m

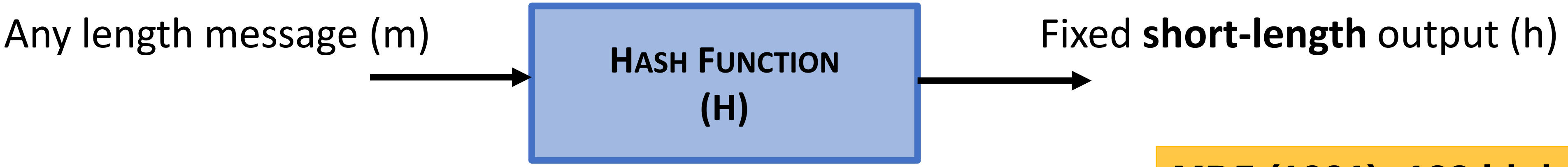
Second pre-image resistance

Given H(m), difficult to get an m' such that H(m') = H(m)

Collision resistance

Difficult to find any m, m' such that H(m) = H(m')

Hash functions



MD5 (1991): 128 bit hash – insecure

SHA0, SHA1: 160 bits – insecure

SHA-2 (224/256 /384/512) – OK but slow

New NIST standard by competition

SHA-3 (224/256 /384/512)

Three security properties

Pre-image resistance

Given H(m), difficult to get m

Second pre-image resistance

Given H(m), difficult to get an m' such that H(m') = H(m)

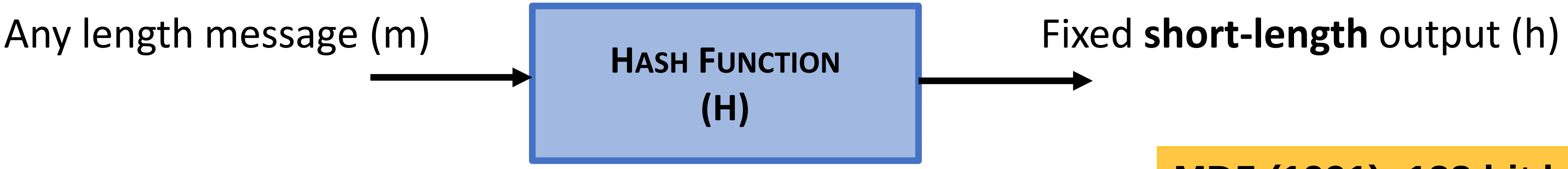
Collision resistance

Difficult to find any m, m' such that H(m) = H(m')

Uses

Support digital signatures, build HMAC, password storage, file integrity, secure commitments, secure logging, blockchain,…

Hash functions


Any length message (m)

Hash Function (H)

Fixed **short-length** output (h)

Three security properties

Pre-image resistance

Given H(m), difficult to get m

Second pre-image resistance

Given H(m), difficult to get an m' such that H(m') = H(m)

Collision resistance

Difficult to find

Don't design your own

HMAC != H(k||m)

Uses

Support digital signatures, build HMAC, password storage, file integrity, secure commitments, secure logging, blockchain,…

MD5 (1991): 128 bit hash – insecure

SHA0, SHA1: 160 bits – insecure

SHA-2 (2) – OK but slow

Don't design your own

New NIST standard by competition

SHA-3 (224/256 /384/512)

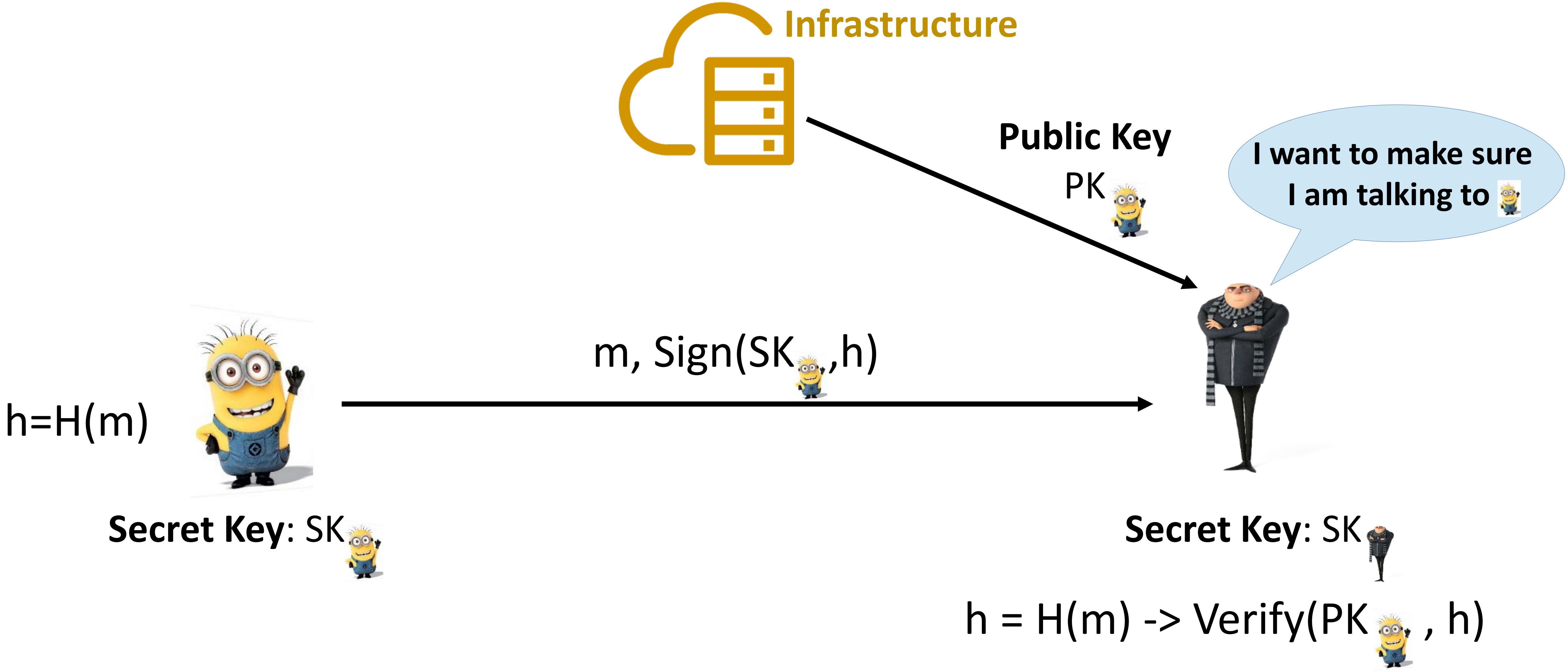
Digital signatures

Examples:

NIST DSA

RSA-PSS

Don't design your own



Public Key Infrastructure

Public Key

PK

I want to make sure I am talking to

m, Sign(SK,h)

h=H(m)

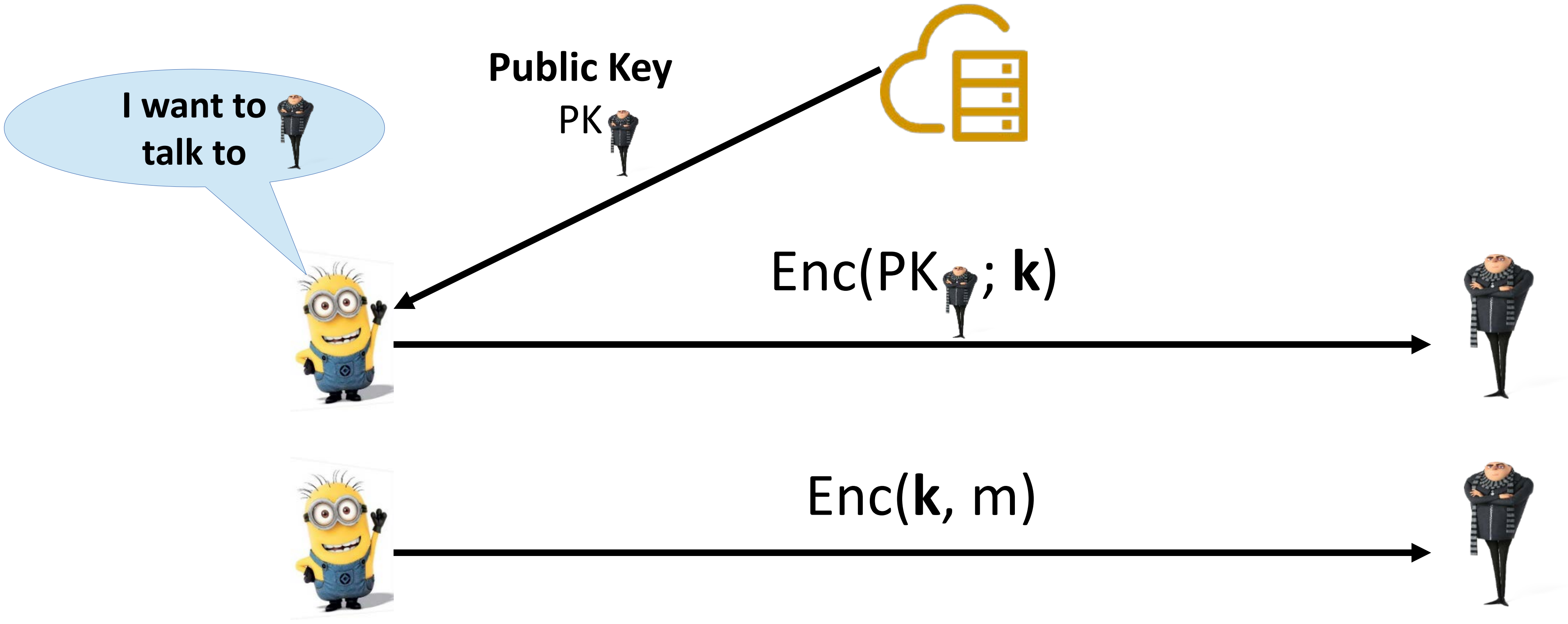
Secret Key: SK

Secret Key: SK

h = H(m) -> Verify(PK , h)

Hybrid encryption

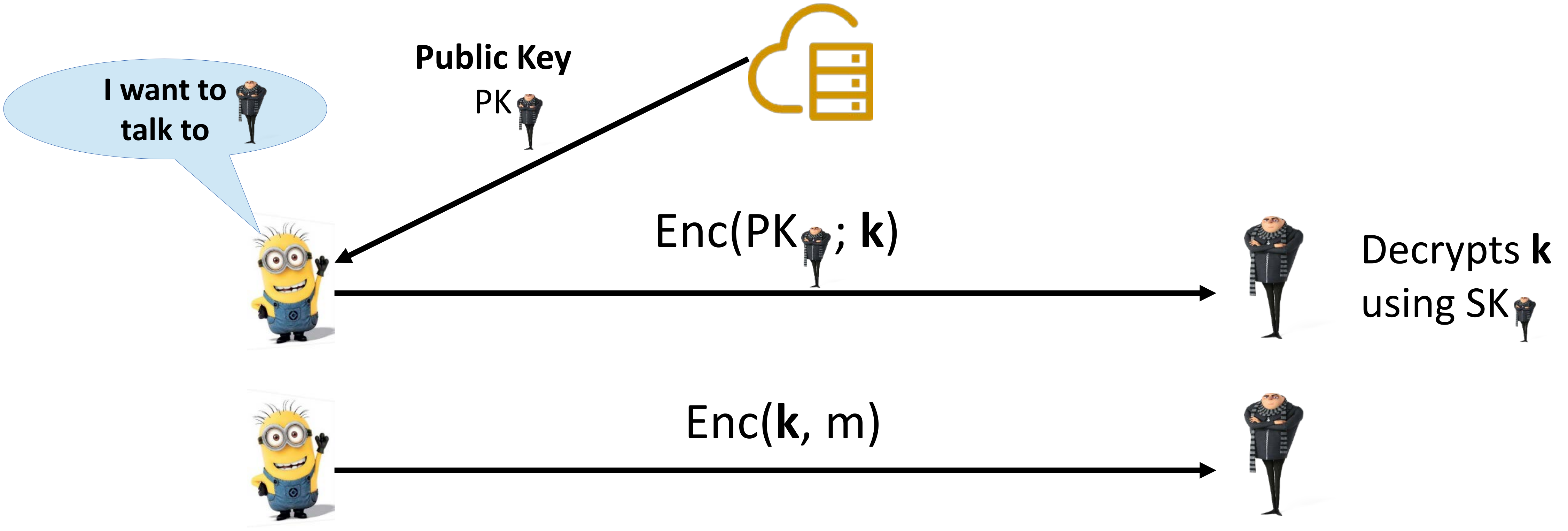
Asymmetric encryption **is slow,** but symmetric **is fast!**



For authentication add signatures!!

Hybrid encryption

Asymmetric encryption **is slow,** but symmetric **is fast!**



I want to talk to

Public Key

PK

Enc(PK ; **k**)

Decrypts **k** using SK

Enc(**k**, m)

For authentication add signatures!!

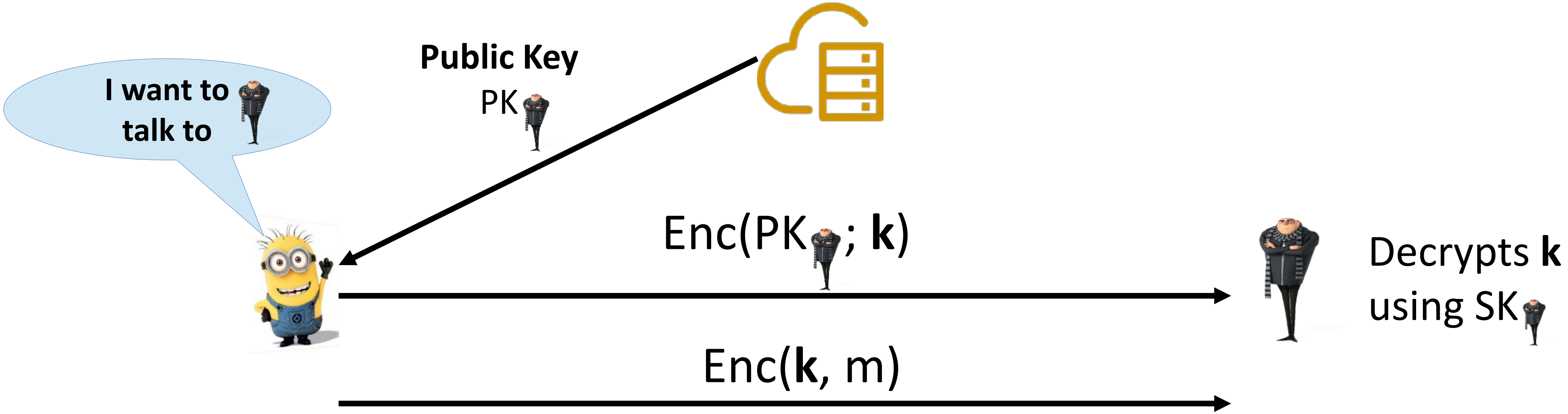
Don't design your own

NOT SO SIMPLE!

e.g. ISO 9798-3

TLS

What if the adversary gets the asymmetric secret key?



I want to talk to

Public Key

PK

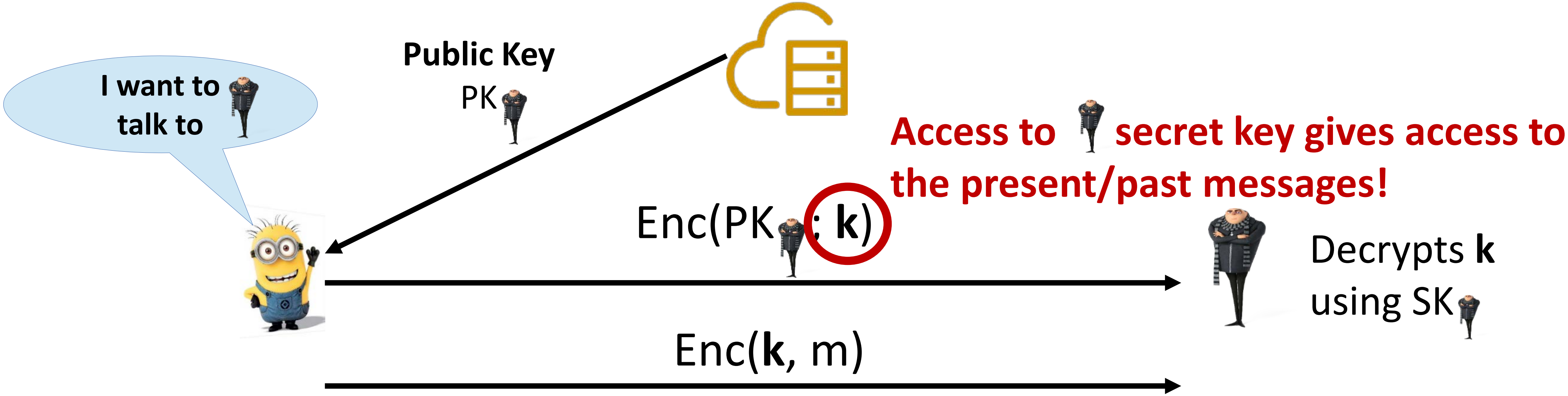
Enc(PK; **k**)

Enc(**k**, m)

Decrypts **k** using SK

What if the adversary gets the asymmetric secret key?

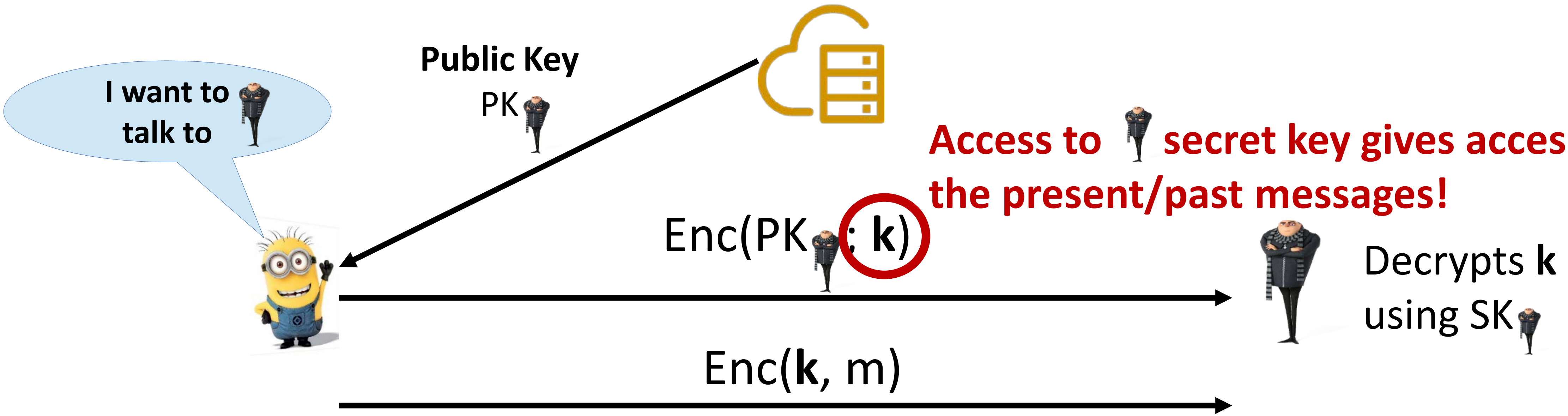
FORWARD SECRECY: the secrecy of the messages in a session is kept even if long term keys are compromised



Access to secret key gives access to the present/past messages!

What if the adversary gets the asymmetric secret key?

FORWARD SECRECY: the secrecy of the messages in a session is kept even if long term keys are compromised



I want to talk to

Public Key

PK

Enc(PK; **k**)

Decrypts **k** using SK

Enc(**k**, m)

Access to secret key gives access to the present/past messages!

How can we obtain this property??

The math you need for the basics

Arithmetic modulo a number: clock arithmetic

6 (mod 12) = 6 (mod 12)

12 (mod 12) = 0 (mod 12)

14 (mod 12) = 2 (mod 12)

Arithmetic modulo a large prime p (>1024 bits)

Addition and multiplication (mod p) can be computed

Exponentiation can be computed [Given (a, x) → a^x mod p?]

Discrete logarithms are **HARD**! [Given (a, a^x mod p) → x?]

Basic Diffie-Hellman key exchange

Shared **public** parameters p , g ⚠

Public Key
$P_b = g^x \bmod p$

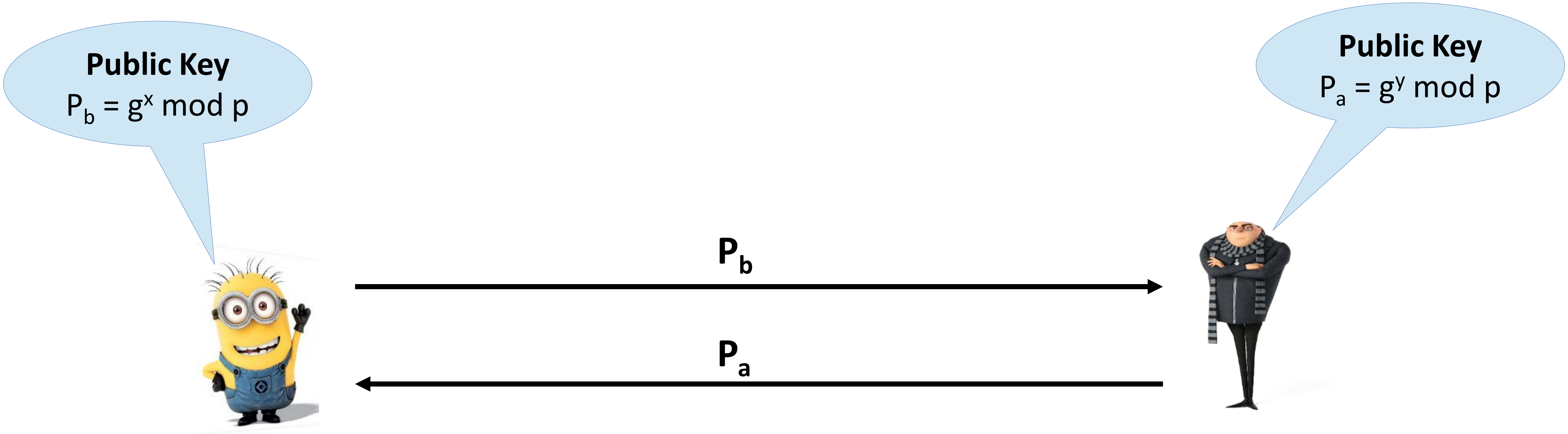
Secret Key: x (random!)

Public Key
$P_a = g^y \bmod p$

Secret Key: y (random!)

Basic Diffie-Hellman key exchange

Shared **public** parameters p , g ⚠



Public Key
$P_b = g^x \bmod p$

Public Key
$P_a = g^y \bmod p$

$\mathbf{P_b}$

$\mathbf{P_a}$

Secret Key: x (random!)

$(P_b)^x = \mathbf{g^{xy}}$ **(mod p)**

Shared secret!!

Secret Key: y (random!)

$(P_a)^y = \mathbf{g^{xy}}$ **(mod p)**

Summary of the lecture

Symmetric cryptography

- Confidentiality: Stream ciphers, Block ciphers (modes of operation!)
- Integrity / Authentication: Message Authentication Codes (MACs)

Asymmetric cryptography

- Confidentiality: Encryption
- Integrity / Authentication: Digital signatures

Hash functions

- Three security properties
- Support Digital Signatures + other functions

Hybrid encryption

best both worlds!

Forward secrecy

Diffie Hellman

Unanswered questions

- How do I build a block cipher?
- How do I build a stream cipher?
- How do I build a hash function?
- How do I implement those?

On the basis of this course: **Do not!**

And only use well established and standardised modes of operation and protocols

Use well established, audited libraries