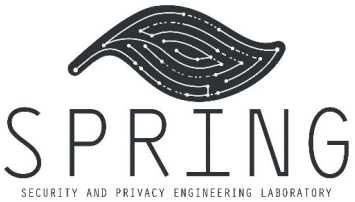


Computer Security and Privacy (COM-301)

Applied cryptography

Carmela Troncoso

SPRING Lab

carmela.troncoso@epfl.ch

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

Important: you will not become a cryptographer

High level introduction to applied cryptography does not qualify you to design cryptographic primitives or protocols!

What you will learn?

What security properties different algorithms offer, and how can algorithms be combined to secure a system

What you will NOT learn?

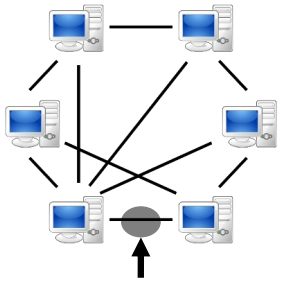
Cryptanalysis

How to prove formally that a scheme is secure

How to securely implement cryptographic schemes

To do these you need a cryptographer (or to become one)

Why does cryptography matter?



Data in transit

Data at rest

What would be the TCB?

What can we do with cryptography?

ENSURE SECURITY PROPERTIES

Cryptography can be used to ensure the **confidentiality** and **integrity** of data in transit or at rest

BUILD SECURE FUNCTIONALITY

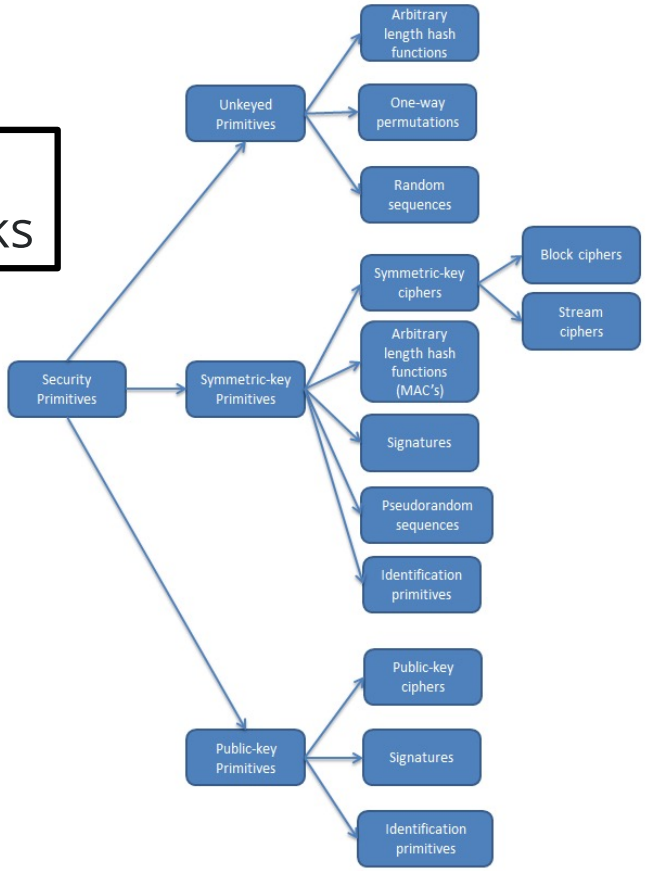
Cryptography can be used, among many others, to build **authentication** protocols**,** to protect from **denial of service,** or to support **anonymous** communications

Key Vocabulary

Cryptographic primitives
universal, exchangeable cryptographic building blocks

Secure functions where
- either you can't break it down any further or
- either there is no security argument for its individual parts

(What exactly a primitive is depends on the level of abstraction)

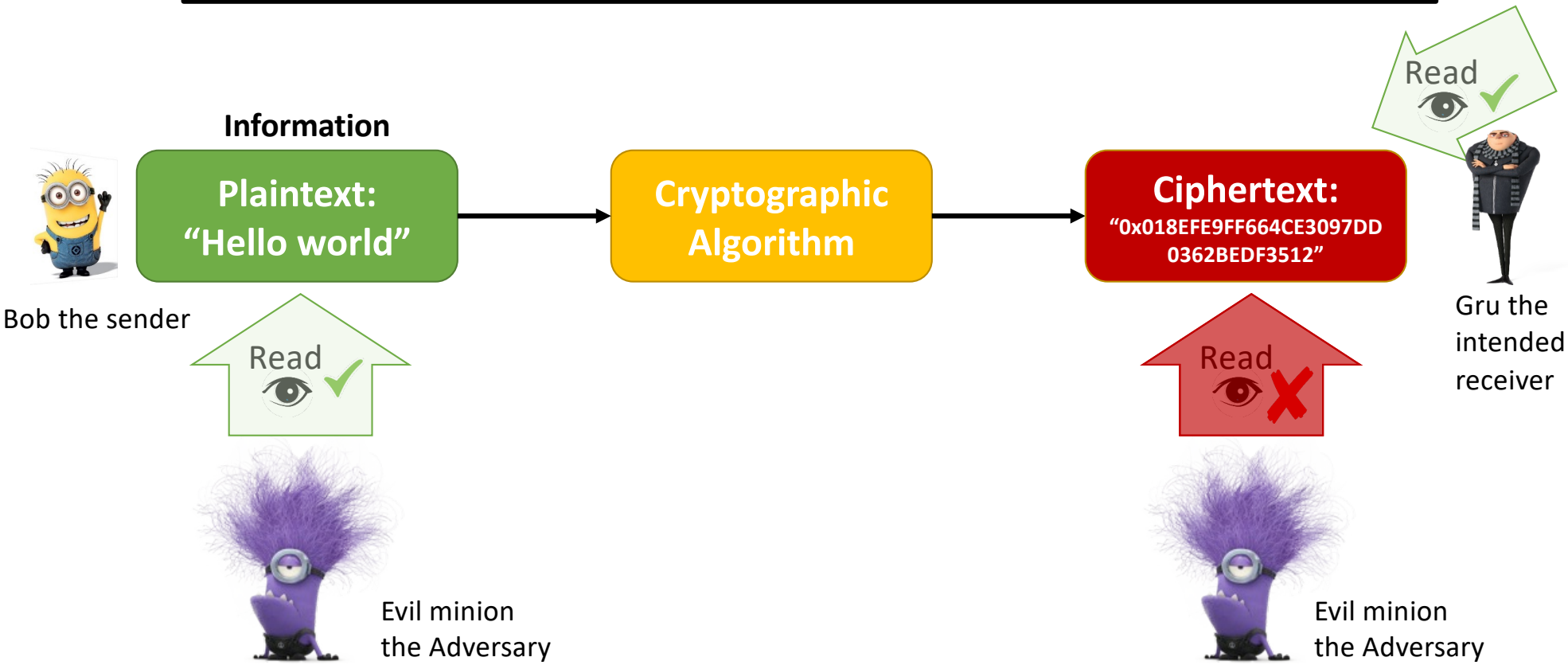


https://crypto.stackexchange.com/questions/39735/whats-a-cryptographic-primitive-really

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

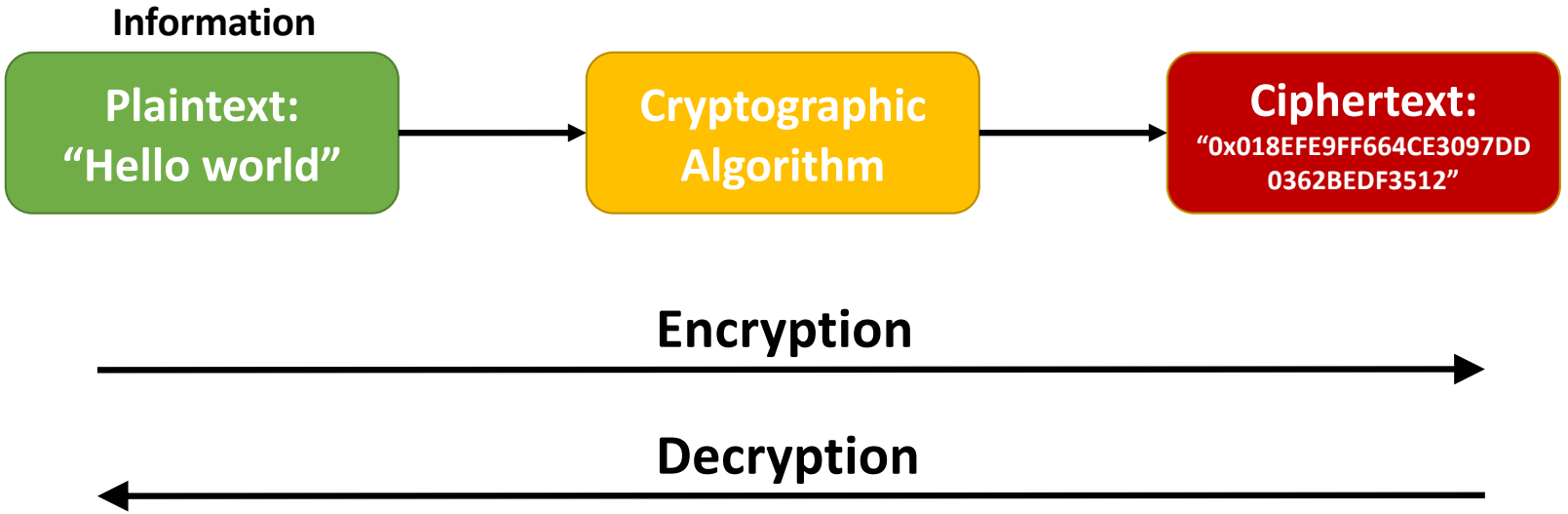
The origins of cryptography: the quest for confidentiality

Confidentiality: information cannot be accessed by unauthorized parties



The origins of cryptography: the quest for confidentiality

Confidentiality: information cannot be accessed by unauthorized parties



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

Cryptographic algorithms for confidentiality

1. Generate key **k** (and make sure intended receiver has it)

   Requires secure generation and sharing protocols

2. Encrypt message **m -> Enc(k,m)**

3. Send encrypted message **Enc(k,m)**

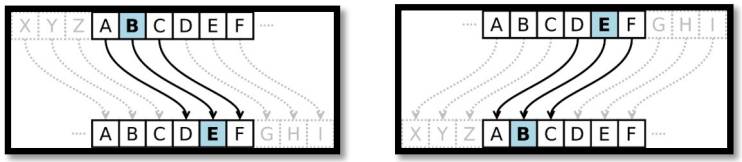
4. Decrypt message **Dec(k,m) -> m**

The first cryptographic algorithms

Caesar's cipher (50 BC)

Rotate the alphabet

Key: number of positions to shift (Julius Caesar used 3)



Encrypt Decrypt

hello world ⟶ khoor zruog

Kamasutra cipher (400 AD)

Permute the alphabet

Key: HOWBUGIACRYEVZXPJQMSNTFDKL

HOWBUGIACRYEV
ZXPJQMSNTFDKL

Encrypt/Decrypt: substitute by opposite letter

hello world ⟶ zkvvx pxfvy

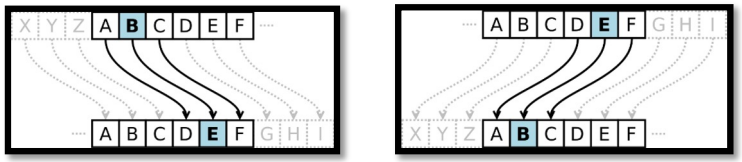
Problem??

The first cryptographic algorithms

Caesar's cipher (50 BC)

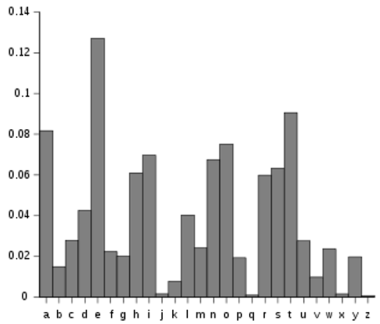
Rotate the alphabet

Key: number of positions to shift (Julius Caesar used 3)



Encrypt Decrypt

hello world ⟶ khoor zruog



Kamasutra cipher (400 AD)

Permute the alphabet

Key: HOWBUGIACRYEVZXPJQMSNTFDKL

HOWBUGIACRYEV
ZXPJQMSNTFDKL

Encrypt/Decrypt: substitute by opposite letter

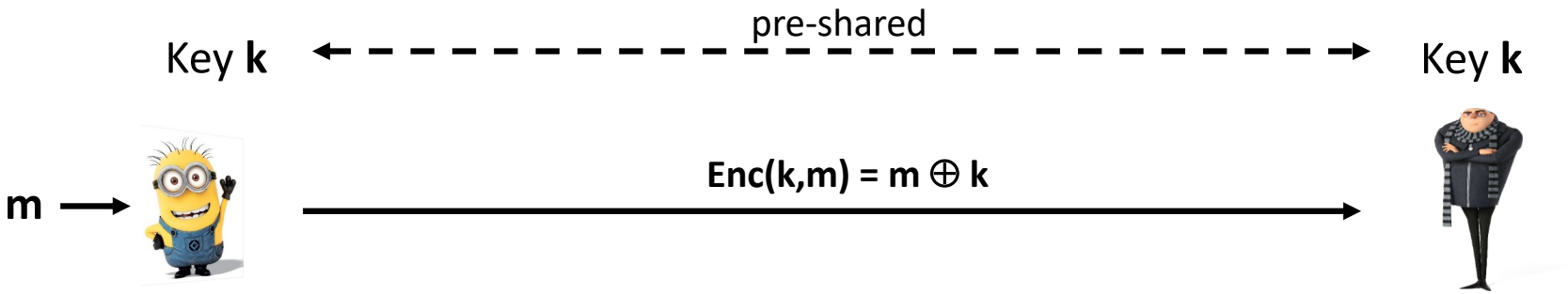
hello world ⟶ zkvvx pxfvy

Problem??

Frequency analysis!

Obtaining perfect secrecy: One Time Pad (OTP)

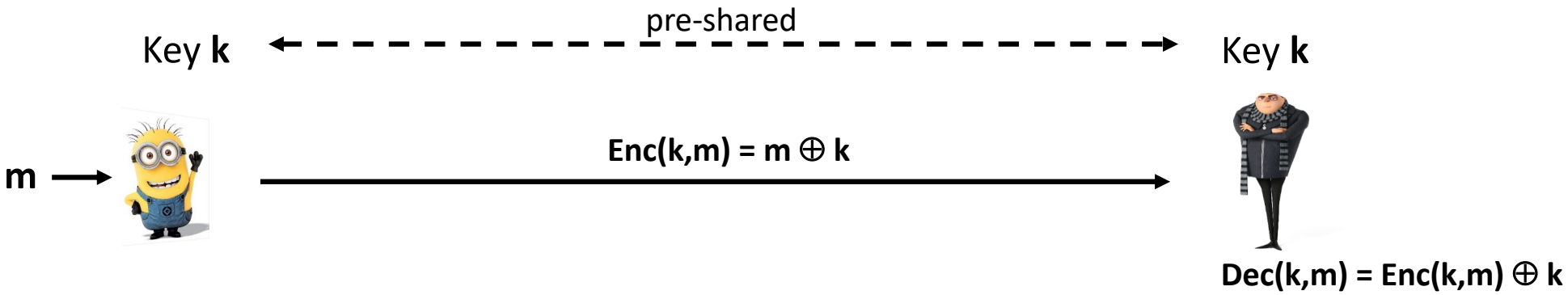
Key = string **k** of **random** bits **as long as the message**



Message	YEAH (ASCII Hex: 59454148)
Binary	01011001010001010100000101001000
OTP-Key ⊕	01110101000111010100101001001010
Encryption	00101100010110000000101100000010

Obtaining perfect secrecy: One Time Pad (OTP)

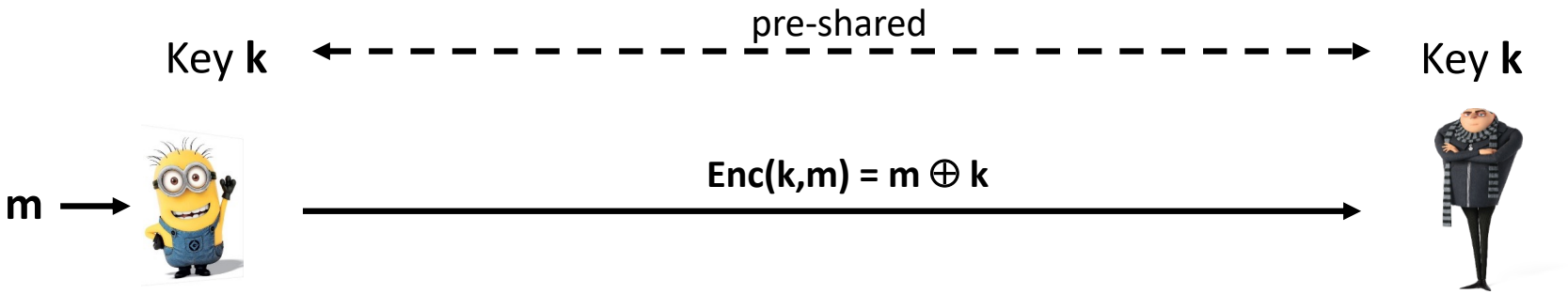
Key = string **k** of **random** bits **as long as the message**



Message	YEAH (ASCII Hex: 59454148)
Binary	01011001010001010100000101001000
⊕ **OTP-Key**	01110101000111010100101001001010
Encryption	00101100010110000000101100000010
⊕	01110101000111010100101001001010 (**same**)
	01011001010001010100000101001000

Obtaining perfect secrecy: One Time Pad (OTP)

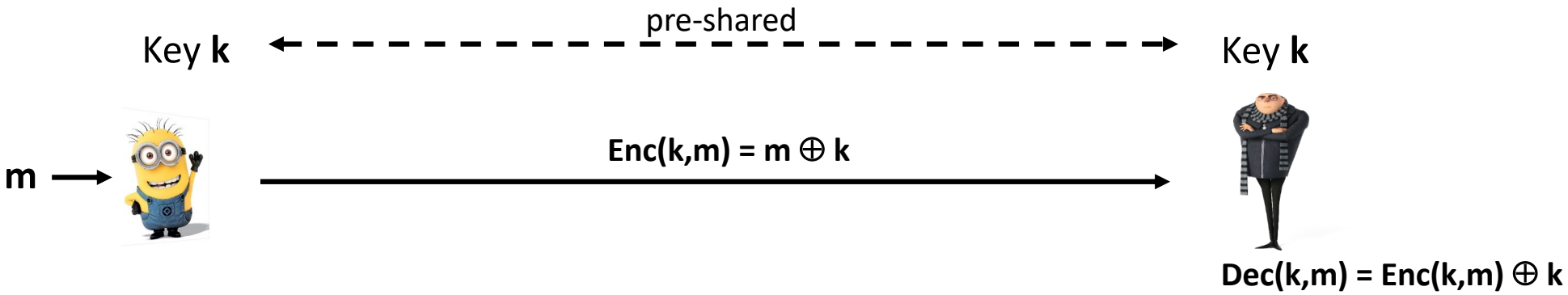
Key = string **k** of **random** bits **as long as the message**



Message	`YEAH (ASCII Hex: 59454148)`			`NOPE (ASCII Hex: 4e4f5045)`
Binary	01011001010001010100000101001000			01001110010011110101000001000101
OTP-Key	⊕ 01110101000111010100101001001010	**same**	⊕	01110101000111010100101001001010
Encryption	00101100010110000000101100000010			00111011010100100001101000001111

Obtaining perfect secrecy: One Time Pad (OTP)

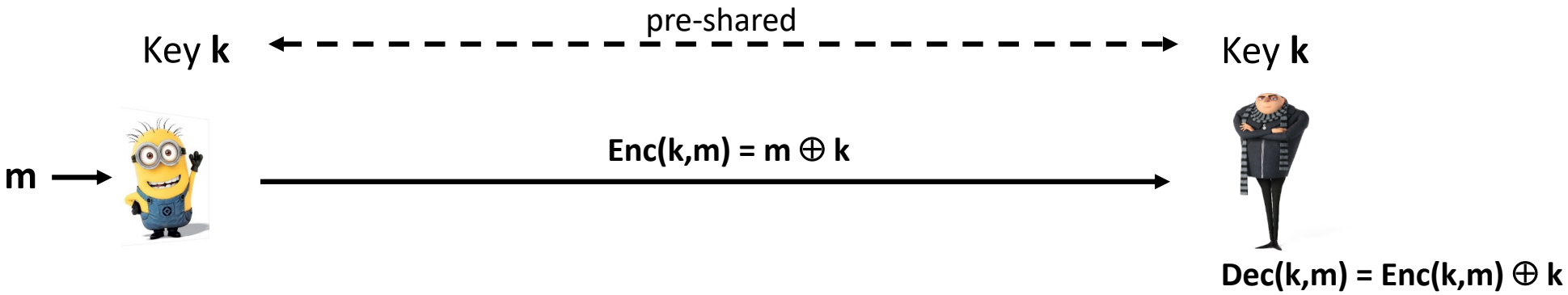
Key = string **k** of **random** bits as long as the message



Message	YEAH		YEAH
Binary (ASCII)	01011001010001010100000101001000	same	01001110010011110101000001000101
OTP-Key	01110101000111010100101001001010	different	11100001010000010111101011010001
Encryption	00101100010110000000101100000010		10101111000011100010101010010100

Obtaining perfect secrecy: One Time Pad (OTP)

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



Message	YEAH		YEAH
Binary (ASCII)	01011001010001010100000101001000	**same**	01001110010011110101000001000101
OTP-Key	01110101000111010100101001001010	**different**	11100001010000010111101011010001
Encryption	00101100010110000000101100000010		10101111000011100010101010010100

Delete "k" – it must never be reused!

$(\mathtt{msg1} \oplus \mathtt{pad}) \oplus (\mathtt{msg2} \oplus \mathtt{pad}) \rightarrow (\mathtt{msg1} \oplus \mathtt{msg2})$

- **Reveals where msg differ**
- **Frequency analysis works**
- **ASCII patterns (space or letter)**
  - **00-**
  - **01-**

Obtaining perfect secrecy: One Time Pad (OTP)

Why do we not use OTPs?

Key as long as the message (nowadays USBs contain several GB) and pre-shared!

Moscow–Washington hotline

Key **must** be random!

”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

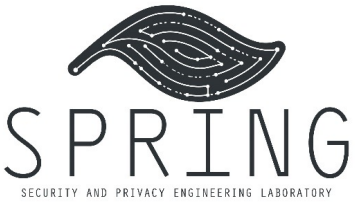
No integrity!

Modern cryptography

Security should not depend on the secrecy of the encryption method (or algorithm), only the secrecy of the keys.

Modern algorithms are based on mathematically difficult problems - for example, prime number factorization, discrete logarithms, etc.

Modern cryptographic algorithms are too complex to be executed by humans.



Computer Security and Privacy (COM-301)

Applied cryptography

Symmetric encryption

Carmela Troncoso

SPRING Lab

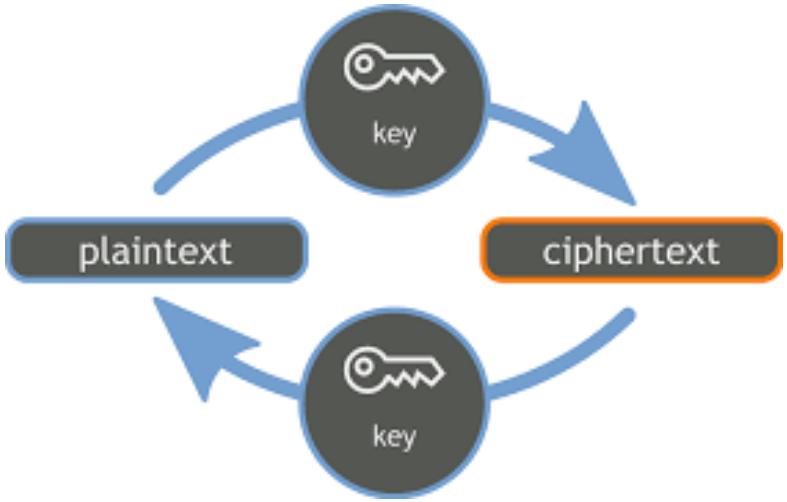
carmela.troncoso@epfl.ch

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

Symmetric encryption ciphers

Encryption of plaintext and decryption of ciphertext are done using

THE SAME KEY



Two types of ciphers:

- Stream ciphers
- Block ciphers

Integrity mechanism:

- Message Authentication Code (MAC)

What is a symmetric cryptographic key?

Fixed-size input to symmetric cryptographic primitives.

The size of the key influences the level of security provided

Key properties

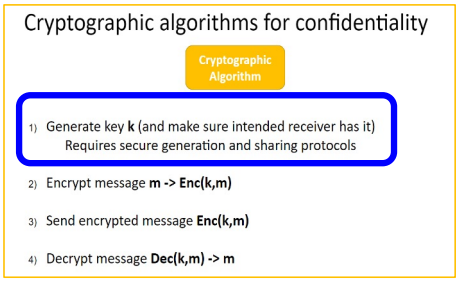
Known to both parties

Partners must agree on the key **before** starting using the primitive

It is reused

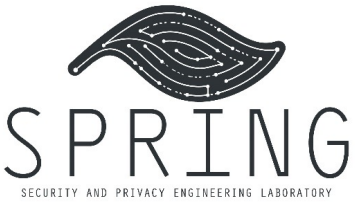
The key is pre-shared once* and then reused

* keys do have a “duration”



It must be secret

Revealing the key eliminates any protection provided by the primitive



Computer Security and Privacy (COM-301)

Applied cryptography

Symmetric encryption - Confidentiality

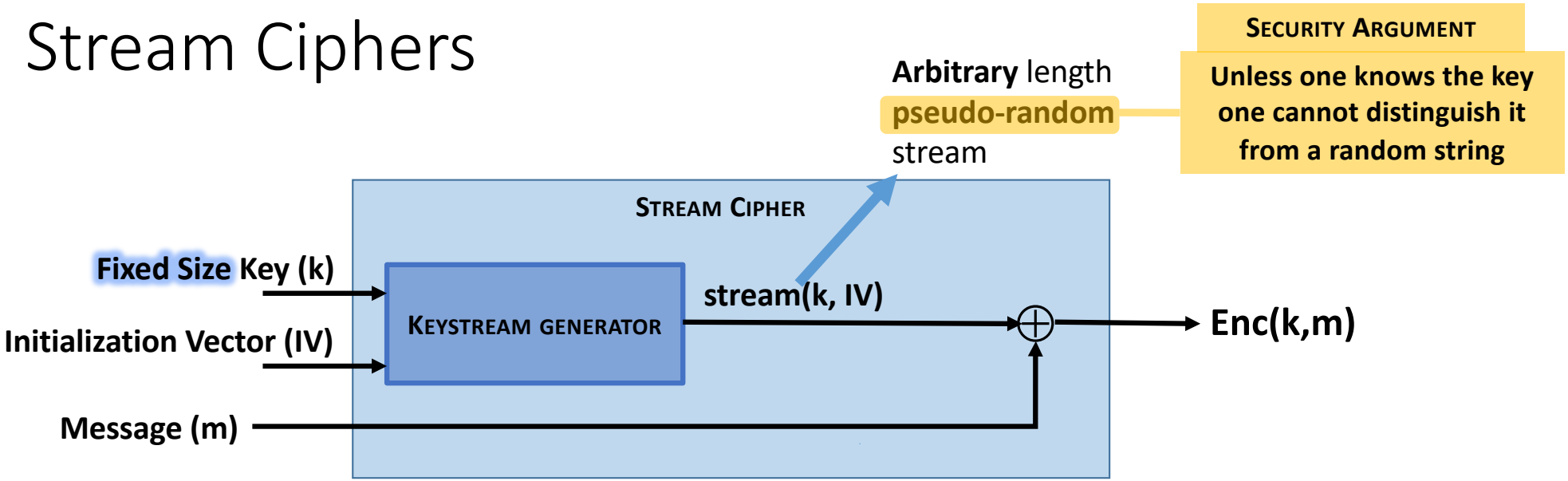
Carmela Troncoso

SPRING Lab

carmela.troncoso@epfl.ch

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

Stream Ciphers



What is an Initialization Vector (IV)?

Initialization Vector: Fixed-size input to iterative cryptographic primitives

Important properties:

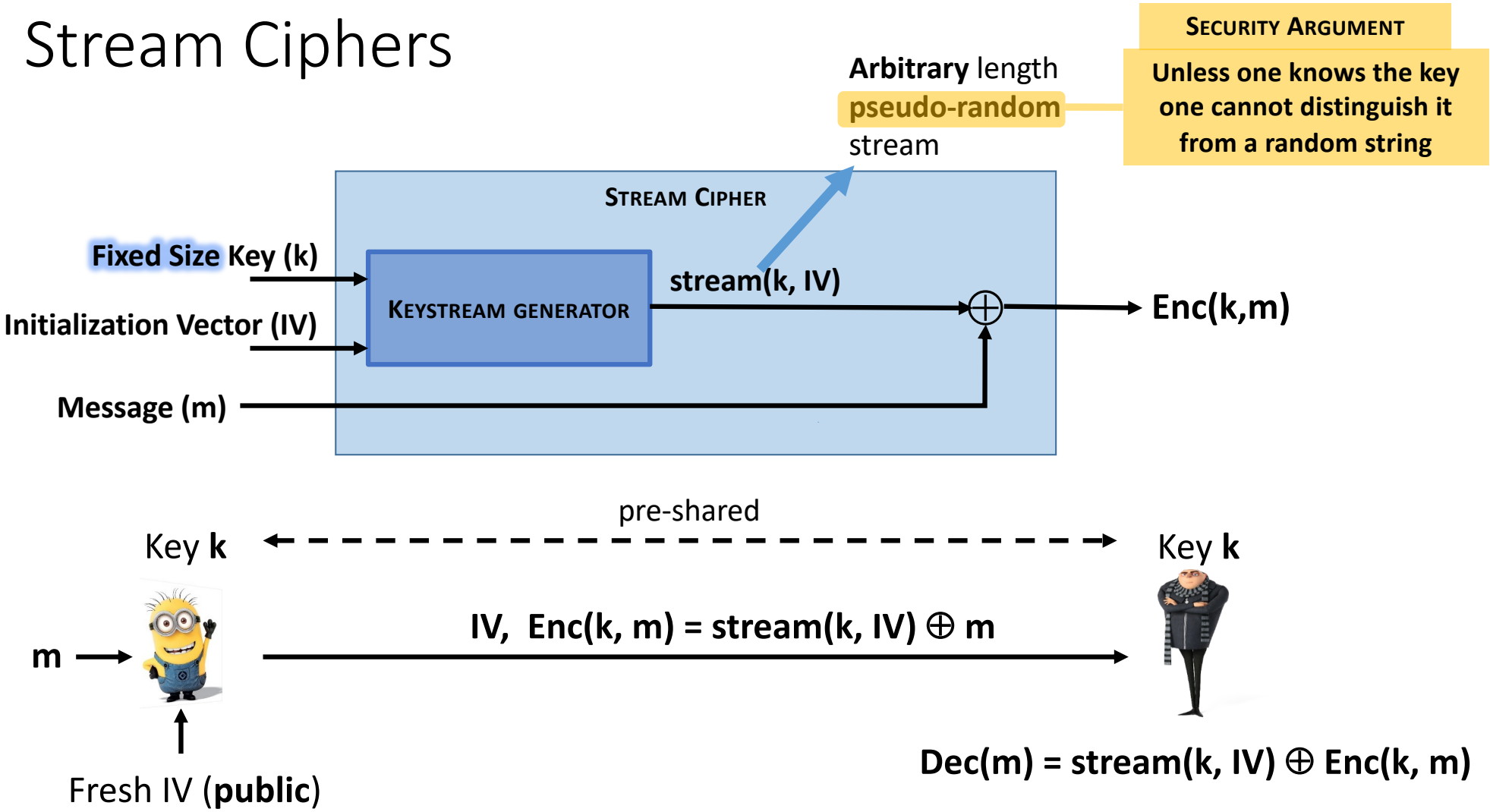
No IV reuse under the same key

Goal: messages encrypted with the same key look different (even the same message)

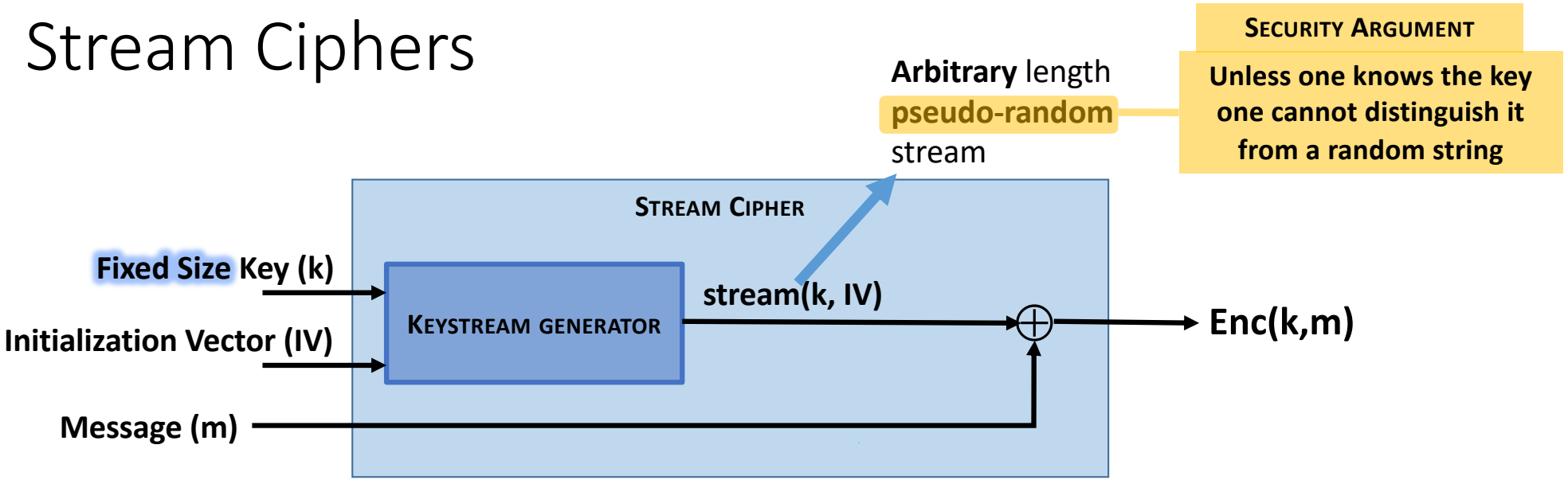
It **does not need to be secret**! Keeping the key secret is enough

But must be *unpredictable* in some block cipher modes

Stream Ciphers



Stream Ciphers



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

Stream ciphers

STRENGTHS

Speed: 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

Stream ciphers

STRENGTHS

Speed: algorith

Low error prop t symbols

WEAKNESSES

Low diffusion: a one encrypted symbol

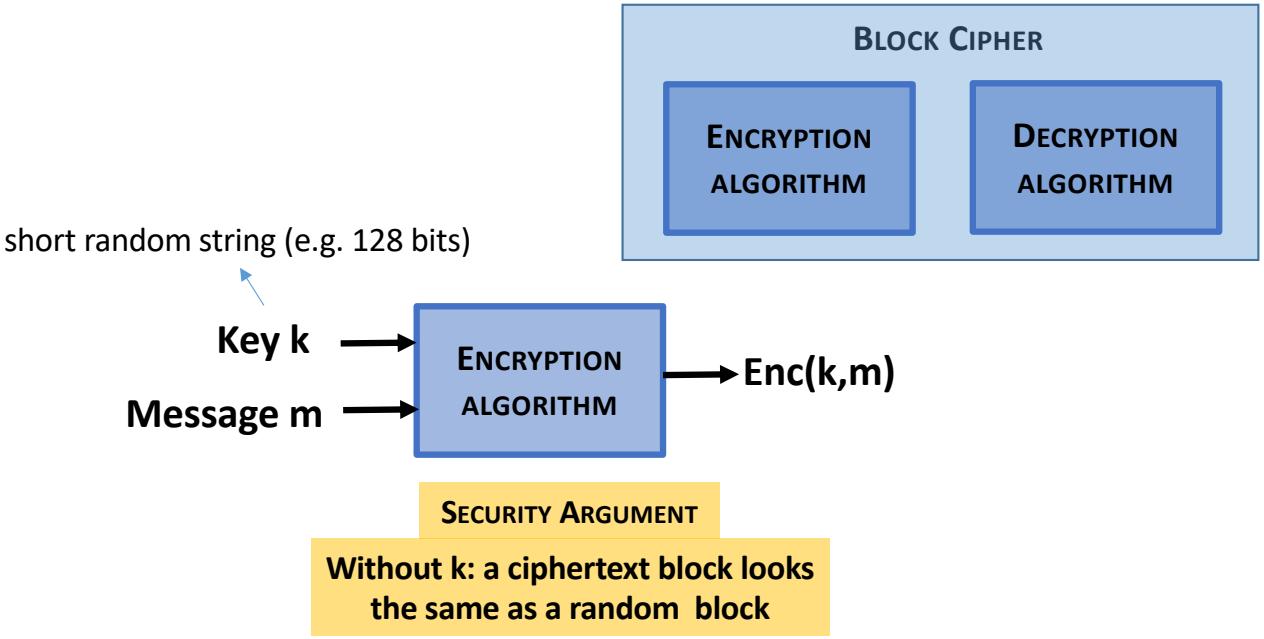
Susceptibility to fficult to detect

Don’t design your own

Trivium **(80 bit key, < 4000 gates in HW)**
Salsa20 **(128/256 bit key, Random access)**

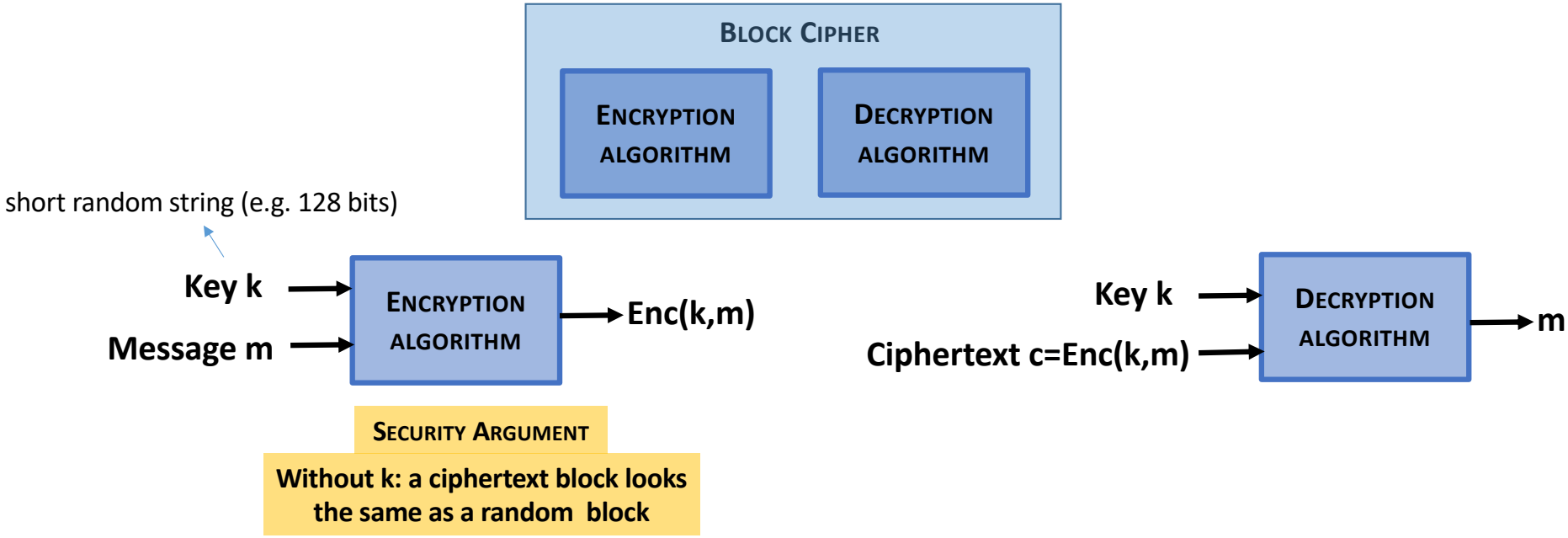
More stream ciphers: https://en.wikipedia.org/wiki/ESTREAM

Block Ciphers



Encryption algorithm: Converts plaintext m to ciphertext c

Block Ciphers

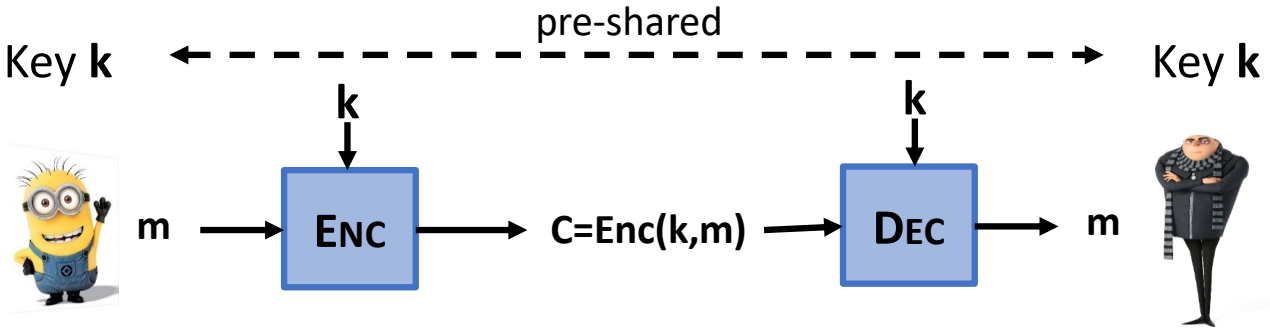


Encryption algorithm: Converts plaintext m to ciphertext c

Decryption algorithm: Concerts ciphertext c to plaintext m.

The inverse of Encryption → Dec(k; Enc(k; m)) = m

Block Ciphers



The algorithms work on blocks that are the size of the key

Typically 128/256 bits

Messages are longer than a block! Requires iteration

Block ciphers' **mode of operation**

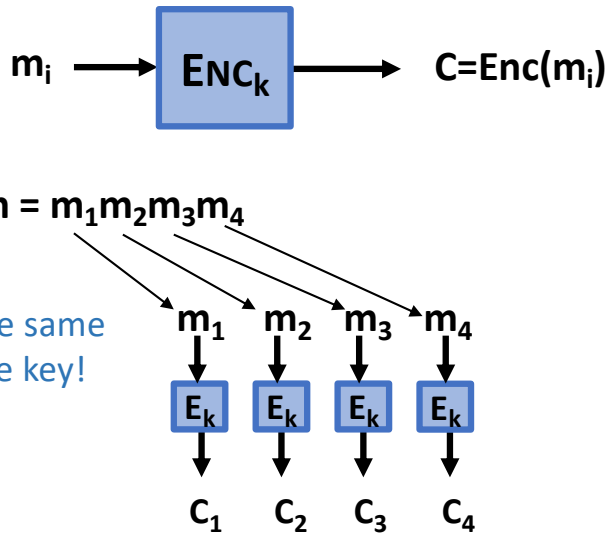
Mode 1: Electronic Code Book (ECB)

Straightforward scheme: encrypt & decrypt single blocks

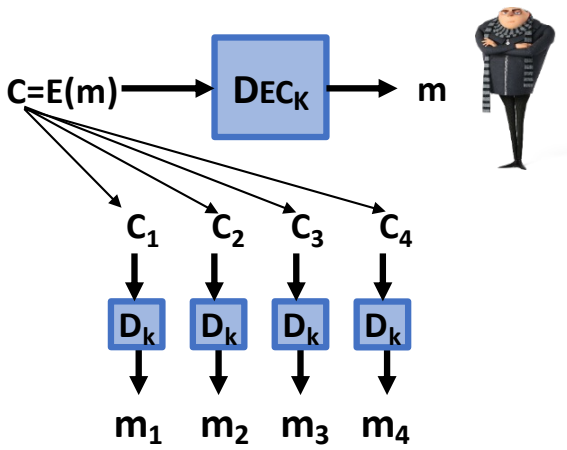
m_i are the same size as the key!

Mode 1: Electronic Code Book (ECB)

Straightforward scheme: encrypt & decrypt single blocks

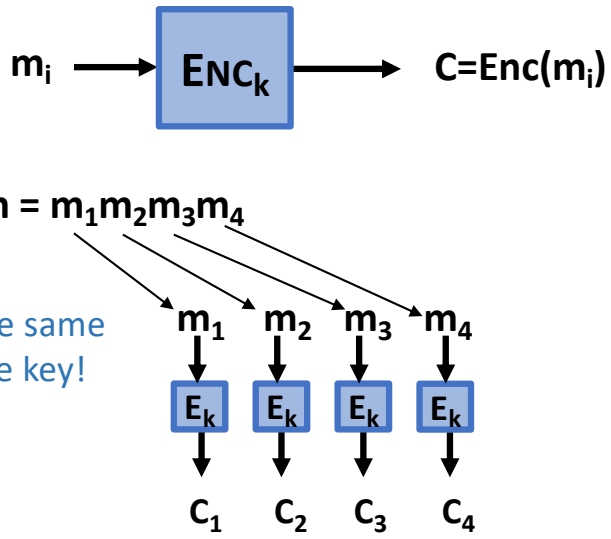


m_i are the same size as the key!

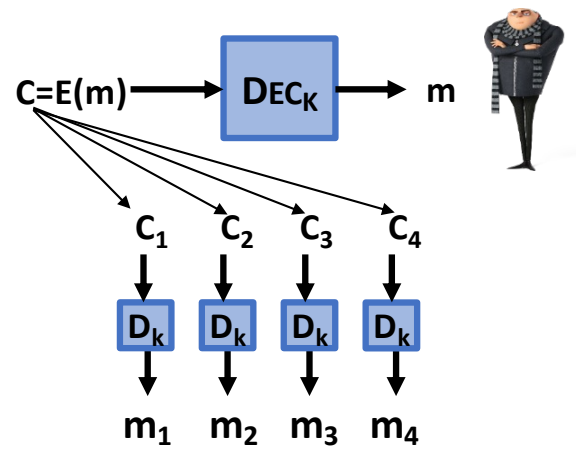


Mode 1: Electronic Code Book (ECB)

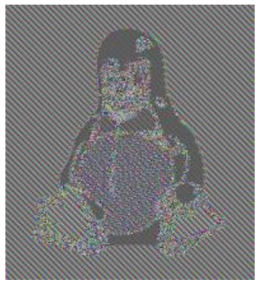
Straightforward scheme: encrypt & decrypt single blocks



m_i are the same size as the key!



Problematic! $m_1=m_2 \rightarrow C_1=C_2$ **Don't use!!**

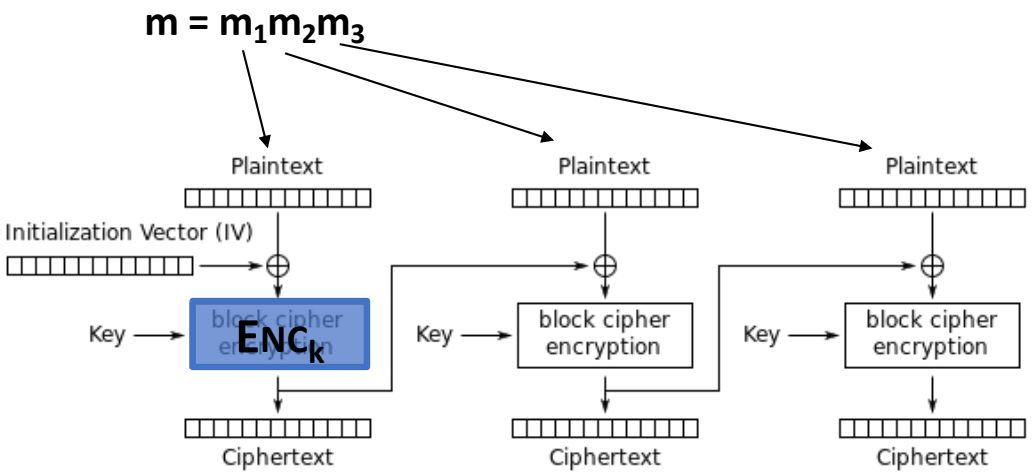


https://en.wikipedia.org/wiki/Block_cipher#/media/File:Tux_ecb.jpg

Mode 2: Cipher Block Chaining (CBC)

Add IV and propagate information across blocks to introduce randomness

m_i → ENC_k → $C=Enc(m_i)$



CBC Encryption

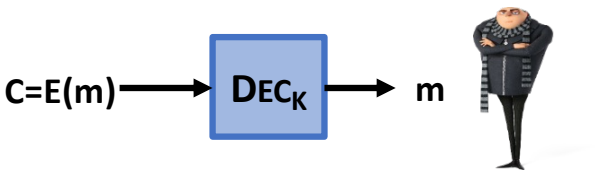
$C_0 = IV$

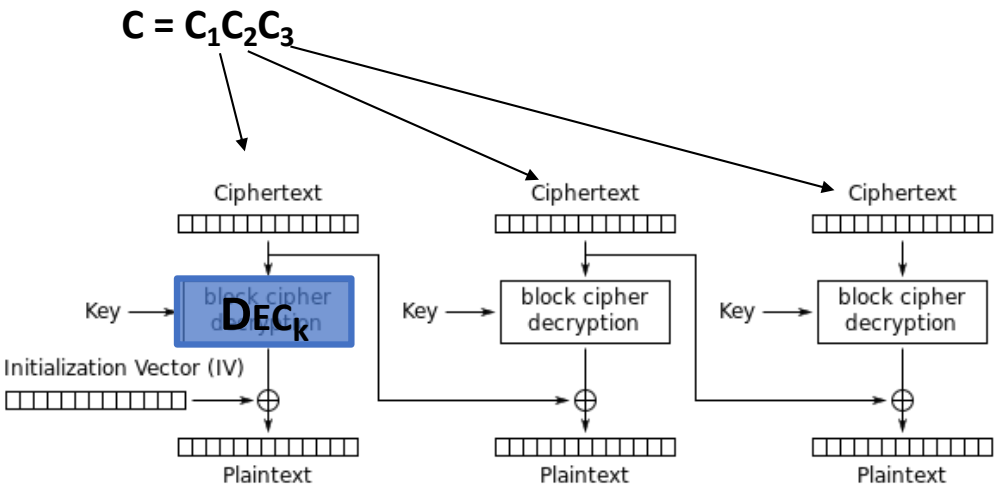
$C_i = Enc(k; m_i \oplus C_{i-1})$

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

Mode 2: Cipher Block Chaining (CBC)

Add IV and propagate information across blocks to introduce randomness





CBC Decryption

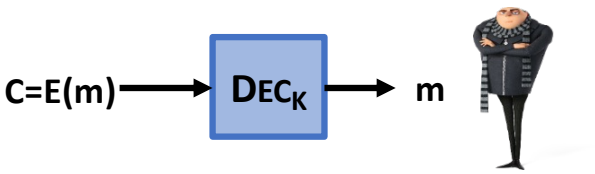
$C_0 = IV$

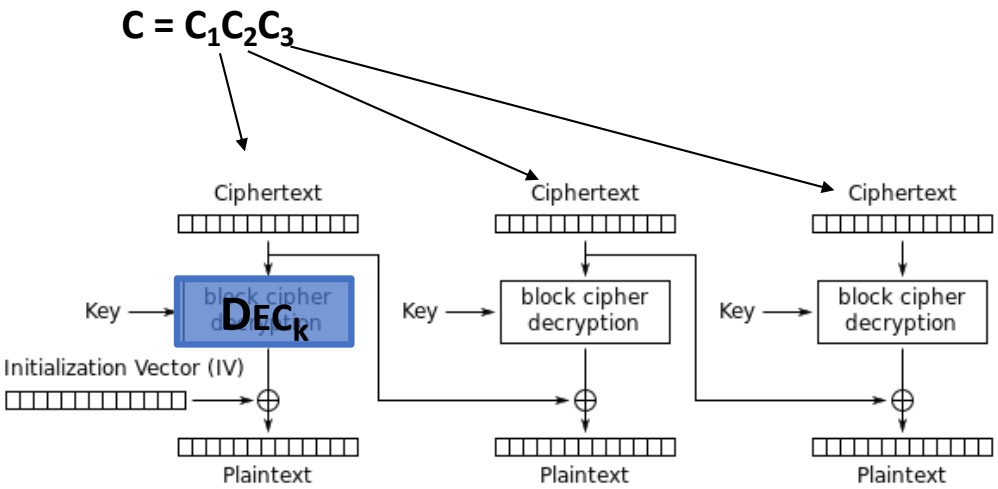
$m_i = Dec(k; C_i)$ XOR C_{i-1}

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

Mode 2: Cipher Block Chaining (CBC)

Add IV and propagate information across blocks to introduce randomness





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

CBC Decryption

$C_0 = IV$

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

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

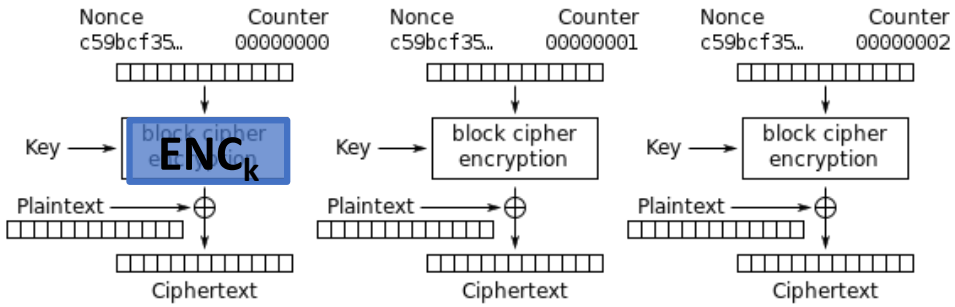
Can you decrypt a block alone? What do you need?

Mode 3: Counter Mode (CTR)

Use increasing nonce to add randomness without dependencies between blocks

m_i → ENC_k → $C=Enc(m_i)$

$m = m_1 m_2 m_3$



CTR Encryption

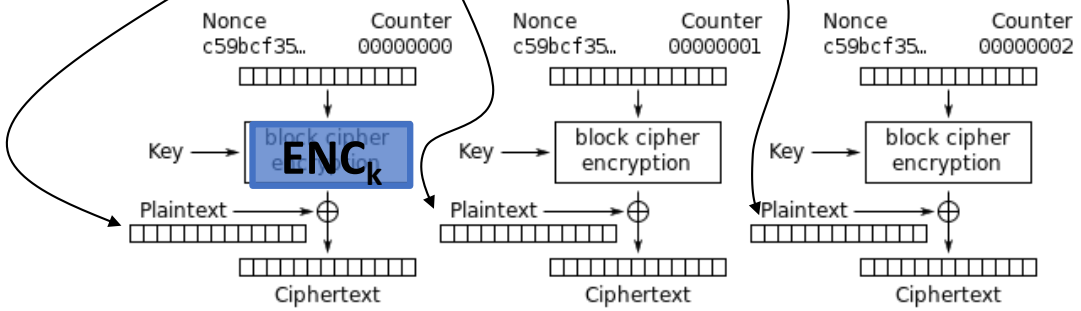
$C_i = Enc(k; Nonce+i) \oplus m_i$

Mode 3: Counter Mode (CTR)

Use increasing nonce to add randomness without dependencies between blocks

m_i → ENC_k → $C=Enc(m_i)$

$m = m_1 m_2 m_3$



CTR Encryption

$C_i = Enc(k; Nonce+i) \oplus m_i$

Counter

Mode 3: Counter Mode (CTR)

Use increasing nonce to add randomness without dependencies between blocks

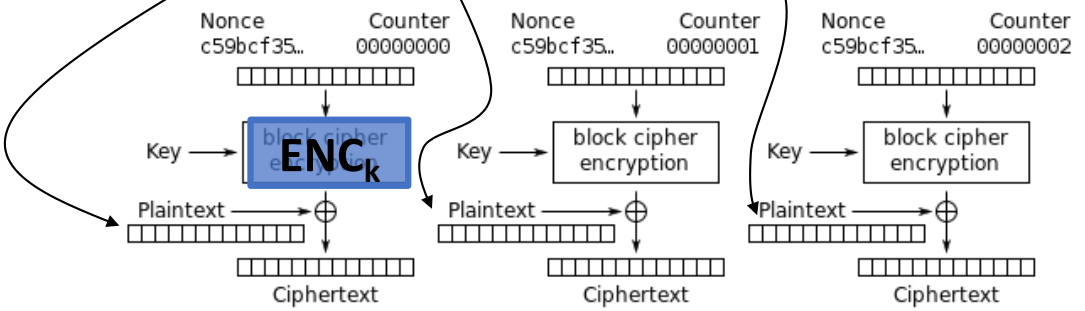
m_i → ENC_k → $C=Enc(m_i)$

Nonce = number used only once

$m = m_1 m_2 m_3$

CTR Encryption

$C_i = Enc(k; Nonce+i) \oplus m_i$



Counter

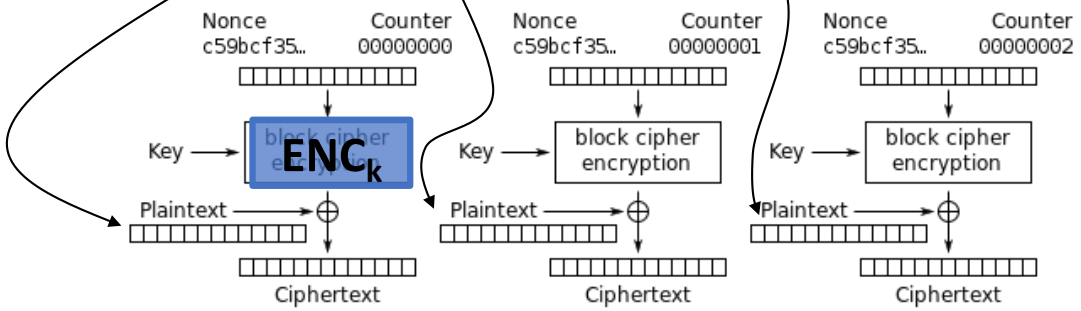
Mode 3: Counter Mode (CTR)

Use increasing nonce to add randomness without dependencies between blocks

m_i → ENC_k → $C=Enc(m_i)$

Nonce = number used only once

$m = m_1 m_2 m_3$



CTR Encryption

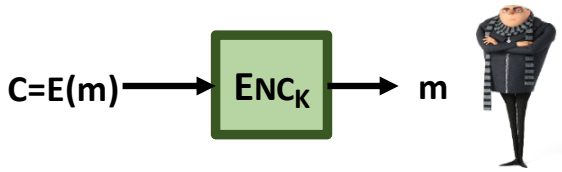
$C_i = Enc(k; Nonce+i) \oplus m_i$

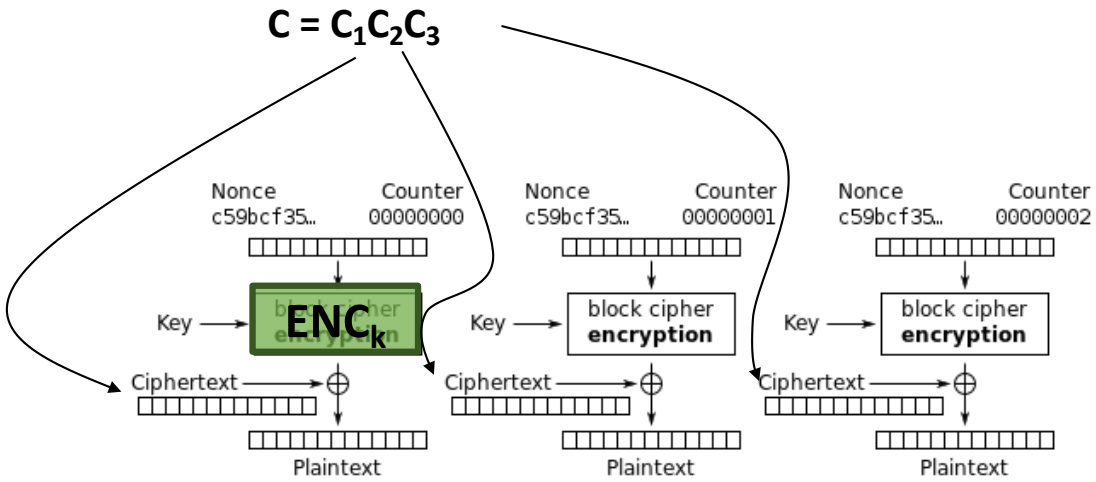
Counter

Do we need the decryption algorithm??

Mode 3: Counter Mode (CTR)

Use increasing nonce to add randomness without dependencies between blocks





CTR Decryption

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

Summary: 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

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

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

*Different modes of operation offer different trade-offs and these weaknesses/strengths may actually not apply.

Summary: Modes of operation

Electronic Code Book (**ECB**)

- ✅ Directly encrypt and decrypt single blocks
- ❌ Large information leakage due to lack of randomness across ciphertext blocks

Cipher Block Chaining (**CBC**)

- ✅ Avoids ECB problems: Each ciphertext block adds randomness to encryption of following block
- ❌ Propagates errors and no parallel encryption

Counter mode (**CTR)**

- ✅ Uses a nonce and an increasing counter to introduce randomness across ciphertext blocks
- ✅ Parallel encryption and decryption

Summary: Block ciphers

STRENGTHS

High diffusied into several ciphertext sy

Immunity toection

WEAKNESSES

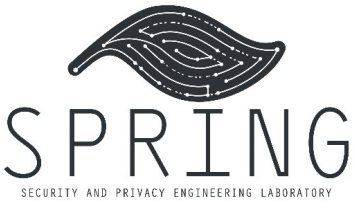
Slow: an ent decryption can begin

Error propagn

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



Computer Security and Privacy (COM-301)

Applied cryptography

Symmetric encryption - Integrity

Carmela Troncoso

SPRING Lab

carmela.troncoso@epfl.ch

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

Cryptography for integrity

Integrity: information cannot be modified by unauthorized parties

Information

Plaintext: "Banana"

Plaintext: "Apple"

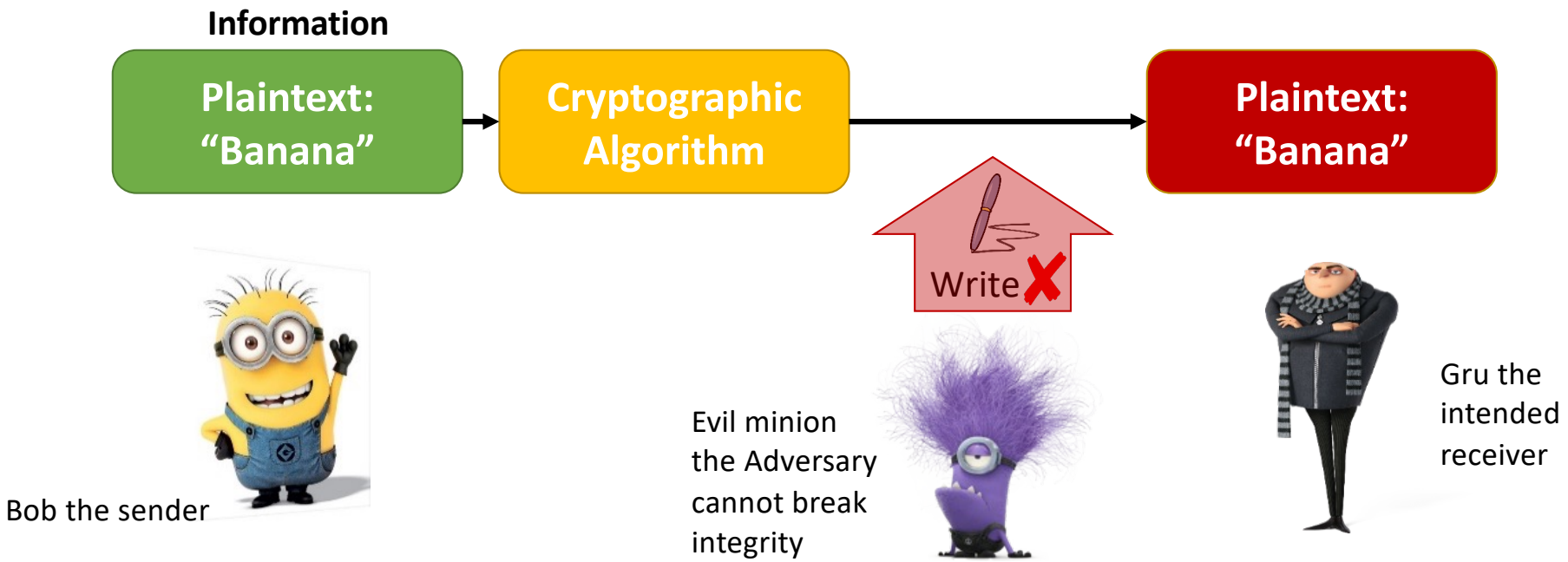
Bob the sender

Evil minion
the Adversary
breaking integrity

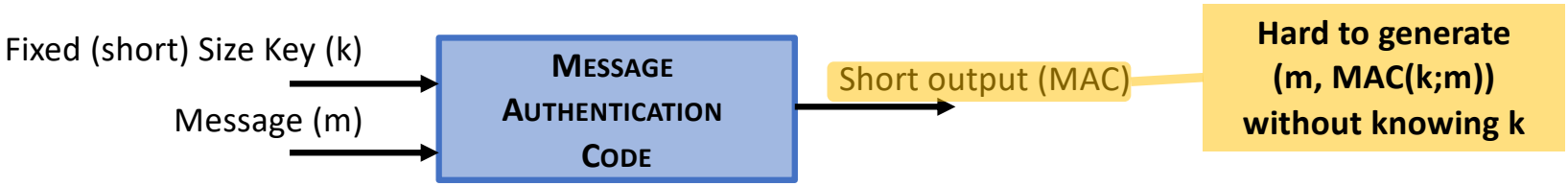
Gru the
intended
receiver

Cryptography for integrity

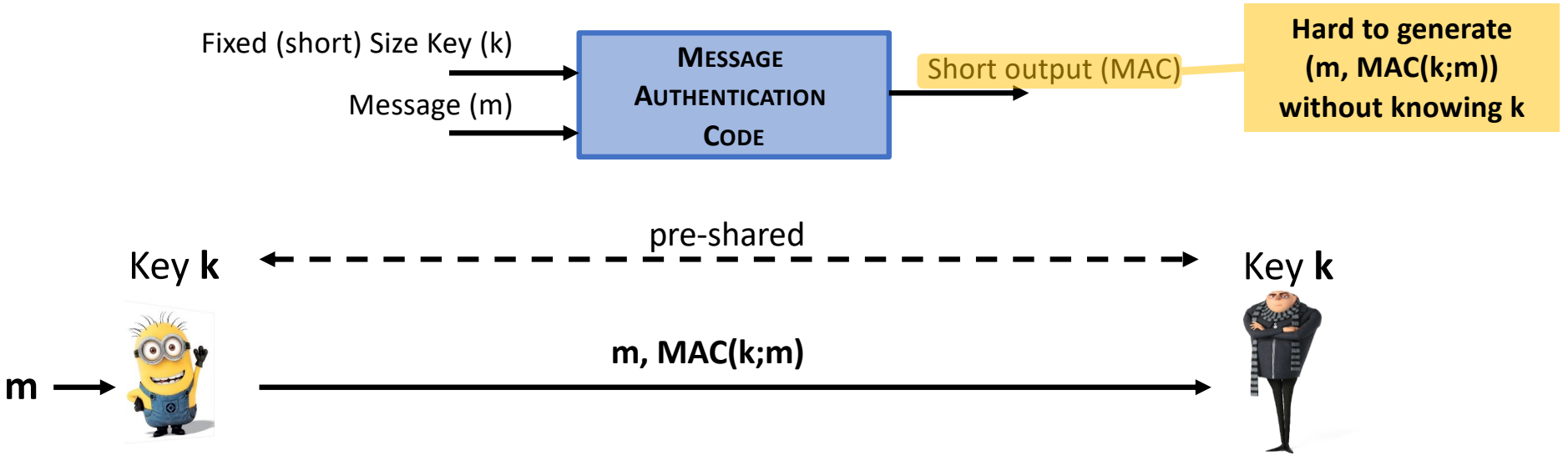
Integrity: information cannot be modified by unauthorized parties



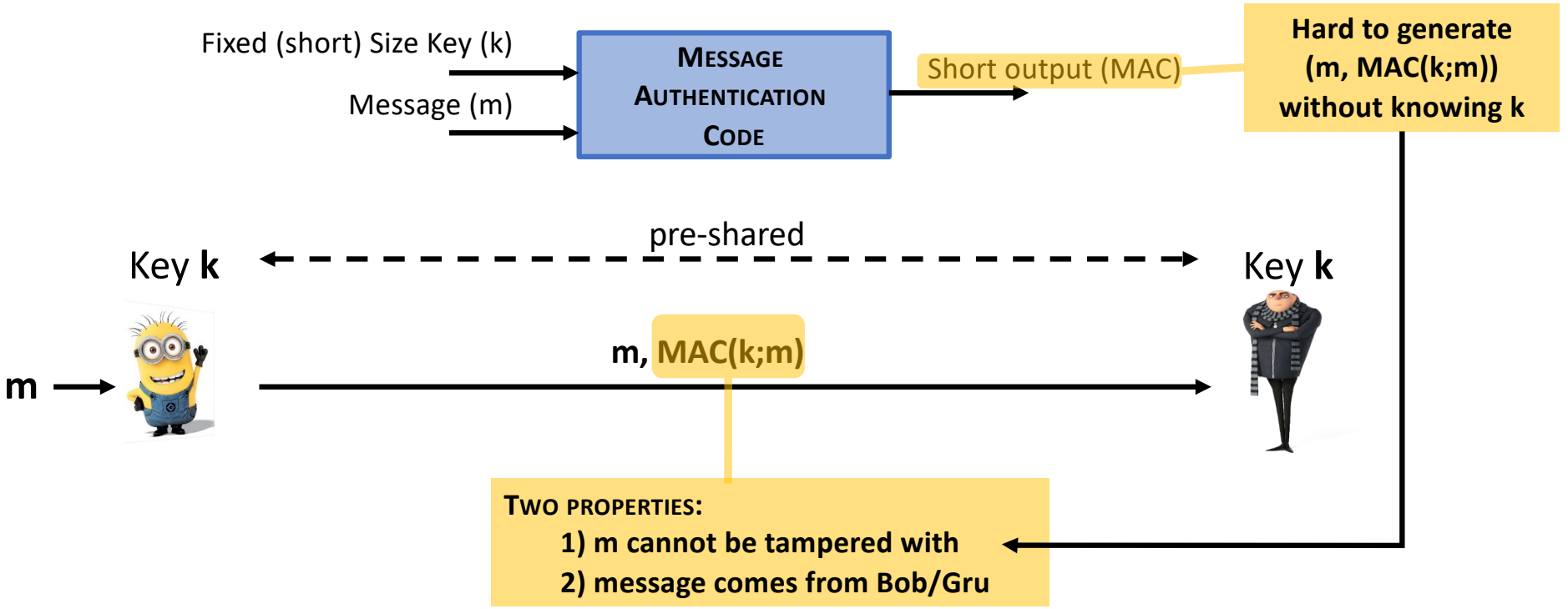
Integrity from symmetric encryption: Message Authentication Codes (MAC)



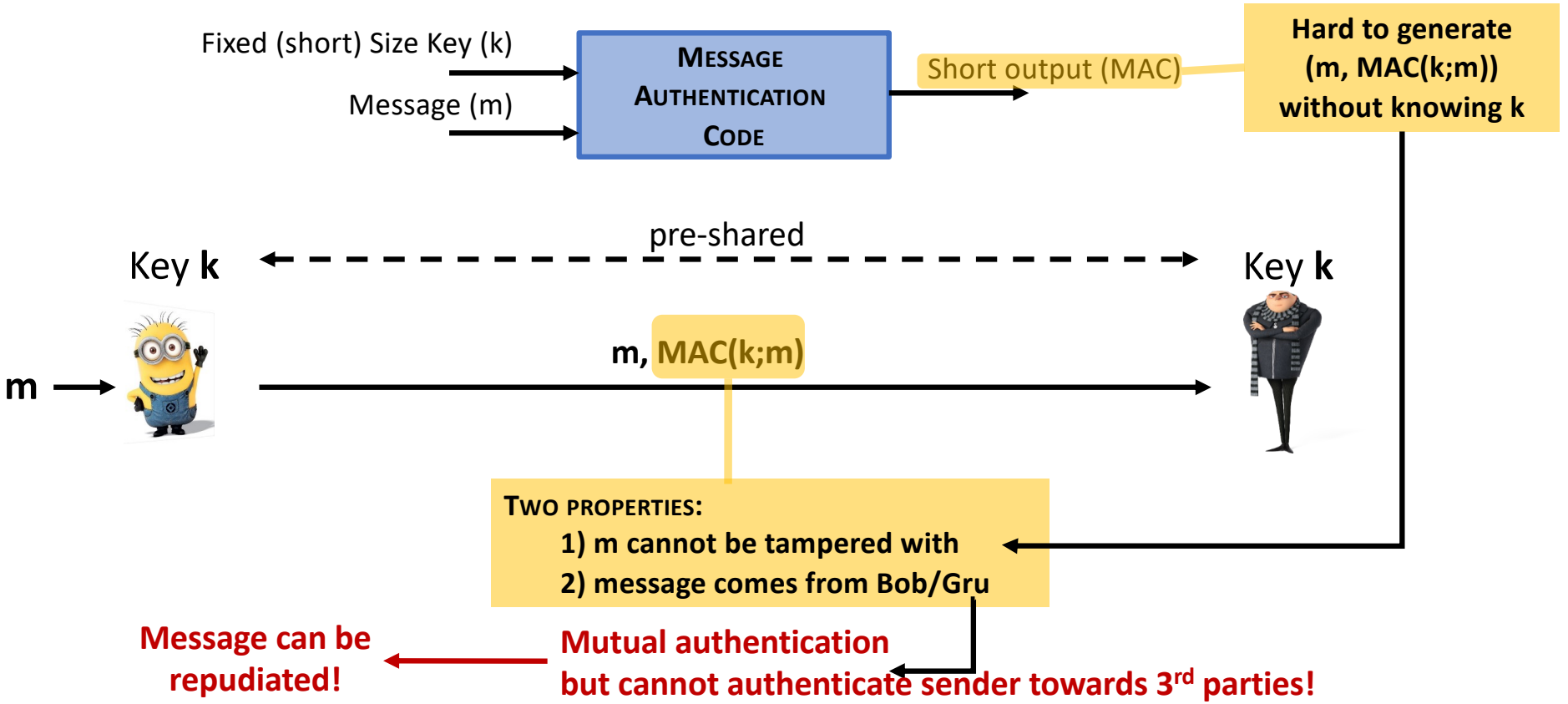
Integrity from symmetric encryption: Message Authentication Codes (MAC)



Integrity from symmetric encryption: Message Authentication Codes (MAC)



Integrity from symmetric encryption: Message Authentication Codes (MAC)



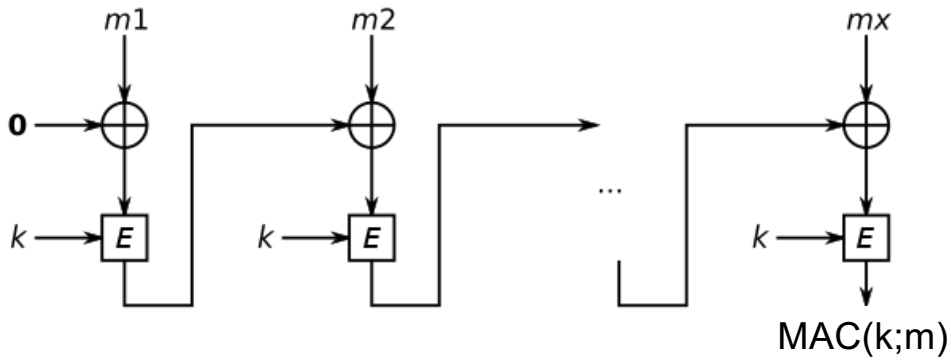
Example MAC: CBC-MAC

Turning a block cipher into a MAC

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

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

$MAC(k; m_1 ... m_x) = C_n$



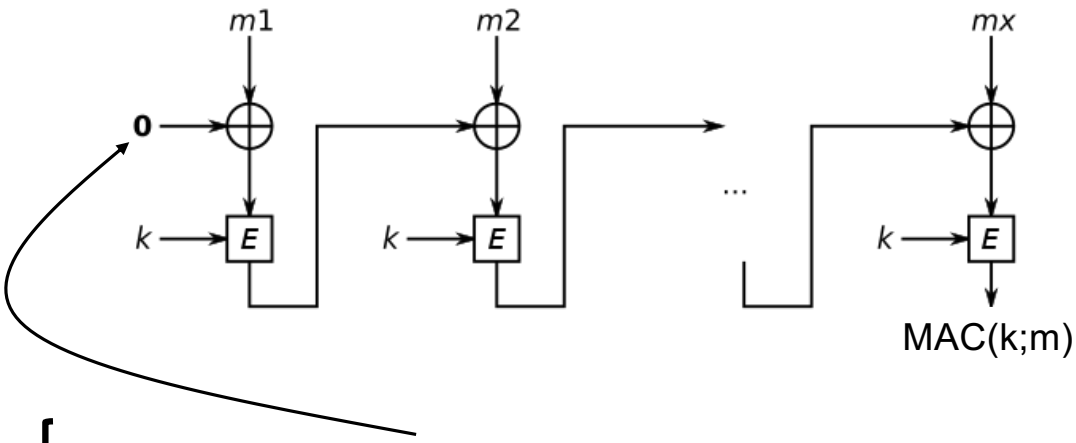
Example MAC: CBC-MAC

Turning a block cipher into a MAC

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

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

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



Differences with respect to CBC

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

Example MAC: CBC-MAC

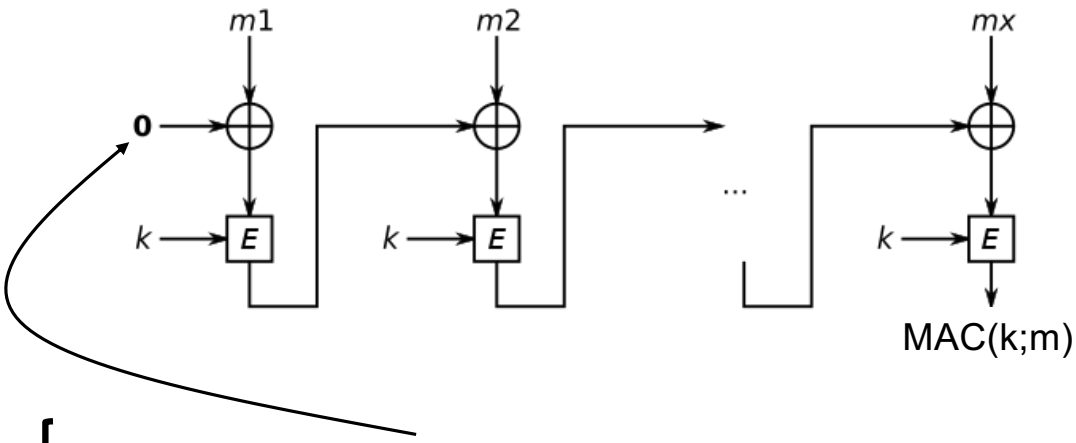
Turning a block cipher into a MAC

Limitation:
Only secure if the length of m is known!

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

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

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

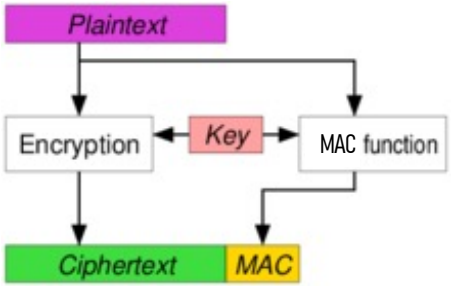


Differences with respect to CBC

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

How to obtain confidentiality and integrity?

ENCRYPT-AND-MAC



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

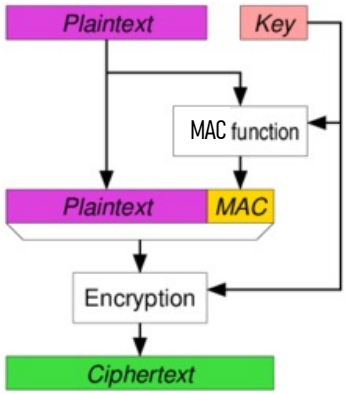
✓ Integrity of the plaintext can be verified

✗ May reveal information about the plaintext → repeated msg, recall the IV of the MAC 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 obtain 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

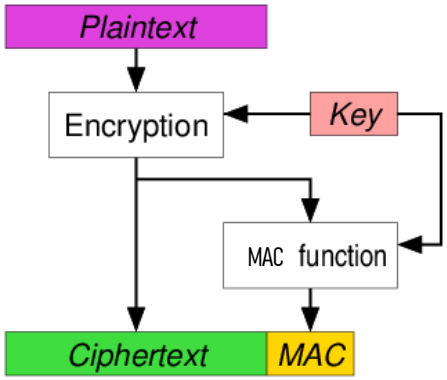
✓ Integrity of the plaintext can be verified

✓ 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 obtain confidentiality and integrity?

ENCRYPT-THEN-MAC

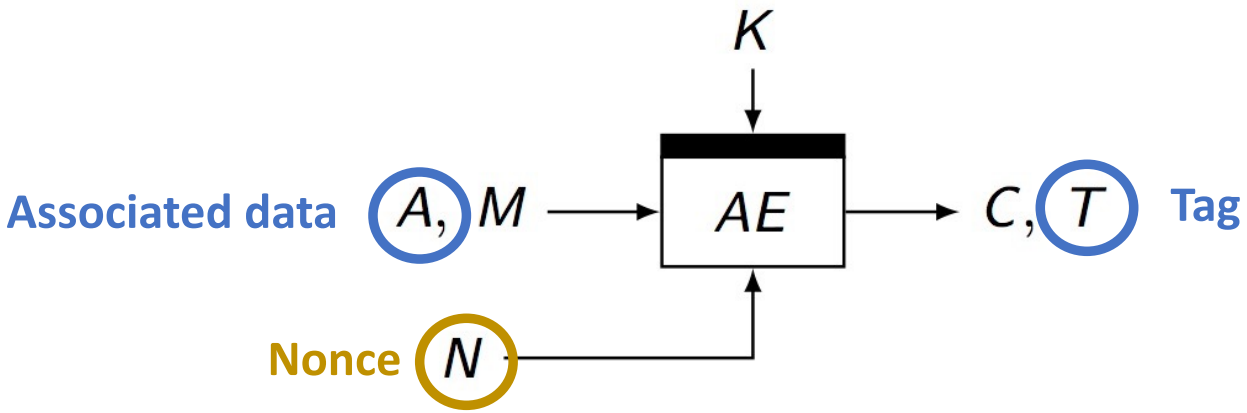


- ✓ Integrity of ciphertext → ensures you only read valid messages! Cipher cannot be attacked!
- ✓ Integrity of the plaintext can be verified
- ✓ No information on the plaintext either, since it is encrypted

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

In practice… (out of the course scope)
Authenticated Encryption with Associated Data (AEAD)

New constructions to avoid home-made combinations



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