Computer Security and Privacy (COM-301)

Applied cryptography II

Carmela Troncoso

SPRING Lab

carmela.troncoso@epfl.ch

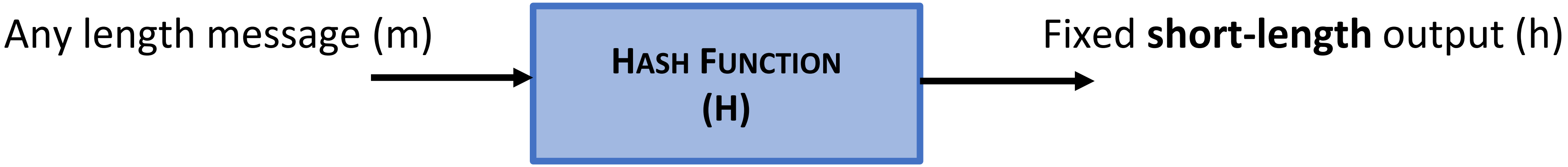
Some slides/ideas adapted from: George Danezis, Yoshi Kohno

Computer Security (COM-301)
Applied cryptography II
Hash functions

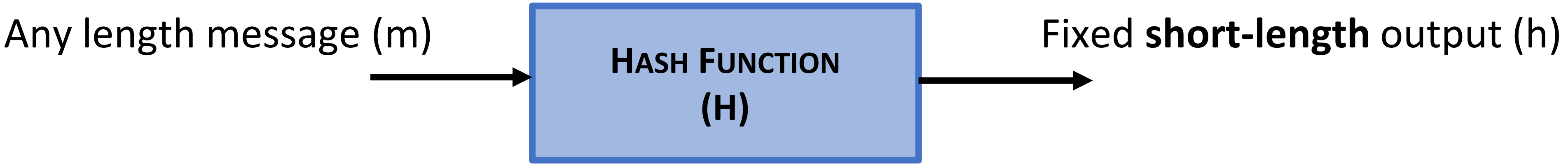
Carmela Troncoso
SPRING Lab
carmela.troncoso@epfl.ch

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

Hash functions



Hash functions



THREE SECURITY PROPERTIES

PRE-IMAGE RESISTANCE

Given H(m), difficult to find m

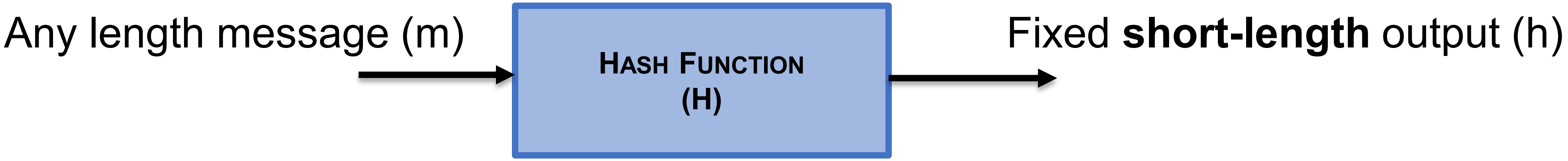
SECOND PRE-IMAGE RESISTANCE

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

COLLISION RESISTANCE

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

Hash functions



THREE SECURITY PROPERTIES

PRE-IMAGE RESISTANCE

Given H(m), difficult to find m

SECOND PRE-IMAGE RESISTANCE

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

COLLISION RESISTANCE

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

MD5 (1991): 128 bit hash – insecure

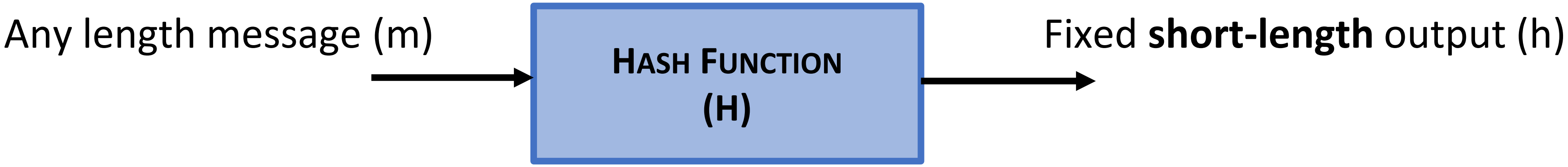
SHA0, SHA1: 160 bits – insecure

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

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

Don't design your own

Hash functions



THREE SECURITY PROPERTIES

PRE-IMAGE RESISTANCE

Given H(m), difficult to find m

SECOND PRE-IMAGE RESISTANCE

Given m, difficult to find an m'!=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, blockchains, …

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

COLLISION RESISTANCE

Difficult to f ')

Don’t design your own

HMAC != H(K||M)

USES

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

Computer Security (COM-301)
Applied cryptography
Asymmetric cryptography

Carmela Troncoso
SPRING Lab
carmela.troncoso@epfl.ch

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

Symmetric Cryptography

Block ciphers, Stream Ciphers, MACs

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

Secure key distribution is a problem!

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

Asymmetric cryptography

Each participant has **two** keys:

One **secret** key that only they know

One **public** key that they can reveal

Pairs of (secret, public) keys are created with specific algorithms

Secret Key: SK_{Bob}

Public Key: PK_{Bob}

Secret Key: SK_{Gru}

Public Key: PK_{Gru}

Asymmetric cryptography

Public keys can be stored in a **public repository**

Public Key Infrastructure

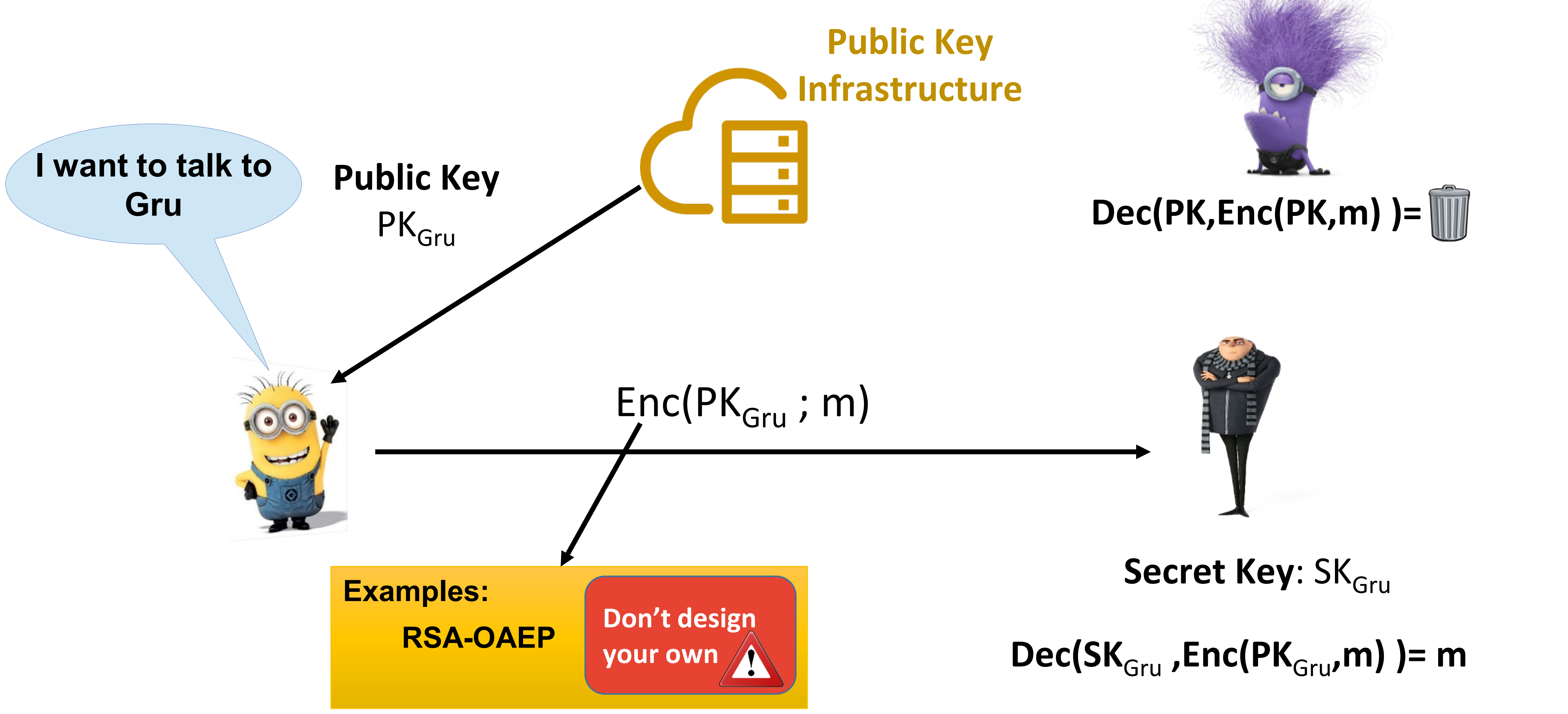
Public Key
PK_{Bob}

Public Key
PK_{Gru}

Secret Key: SK_{Bob}

Secret Key: SK_{Gru}

Asymmetric cryptography: confidentiality



Asymmetric cryptography: integrity

Public Key Infrastructure

Public Key
PK_{Bob}

m, Sign(SK_{Bob},m)

Secret Key: SK_{Bob}

Verify(PK_{Bob}, m, Sign(SK_{Bob},m))= Yes or No

Examples:

- **NIST DSA**
- **RSA-PSS**

Don't design your own

Asymmetric cryptography: integrity

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

Public Key Infrastructure

Public Key

PK_{Bob}

Secret Key: SK_{Bob}

m, Sign(SK_{Bob},m)

Verify(PK_{Bob}, m, Sign(SK_{Bob},m))= Yes or No

Examples:

NIST DSA

RSA-PSS

Don't design your own

Digital Signatures

Properties:

Integrity of message

Authenticity sender

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

Application: **Public Key Infrastructure**: 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:

Integrity of message

Authenticity sender

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

Encryption key pair != Signature key pair

Application: **Public Key Infrastucture**: Certificates

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

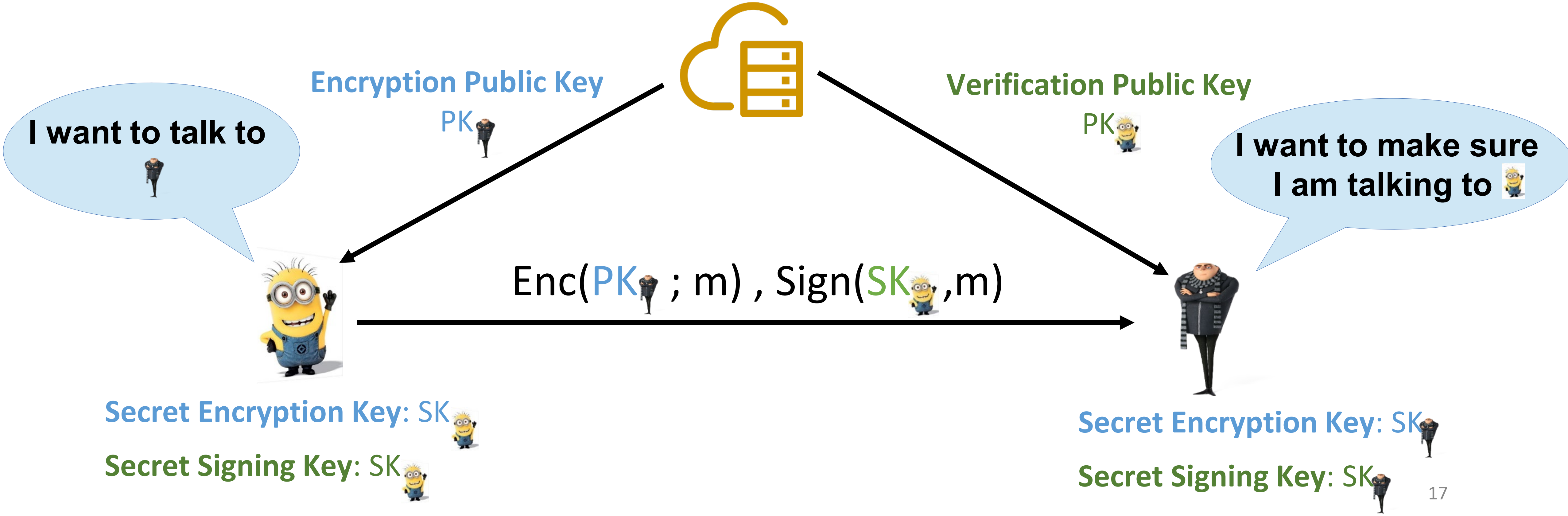
(2) Authority signs mapping between names and verification keys.

All together

ASYMMETRIC CRYPTOGRAPHY

Users have two pairs of keys (secret key SK, public key PK)

Confidentiality	**Dec(SK,Enc(PK,m))= m**
Integrity/Authentication	**Sig(SK,m) = s; Verify(PK,Sig(SK,m)) = YES/NO**



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**

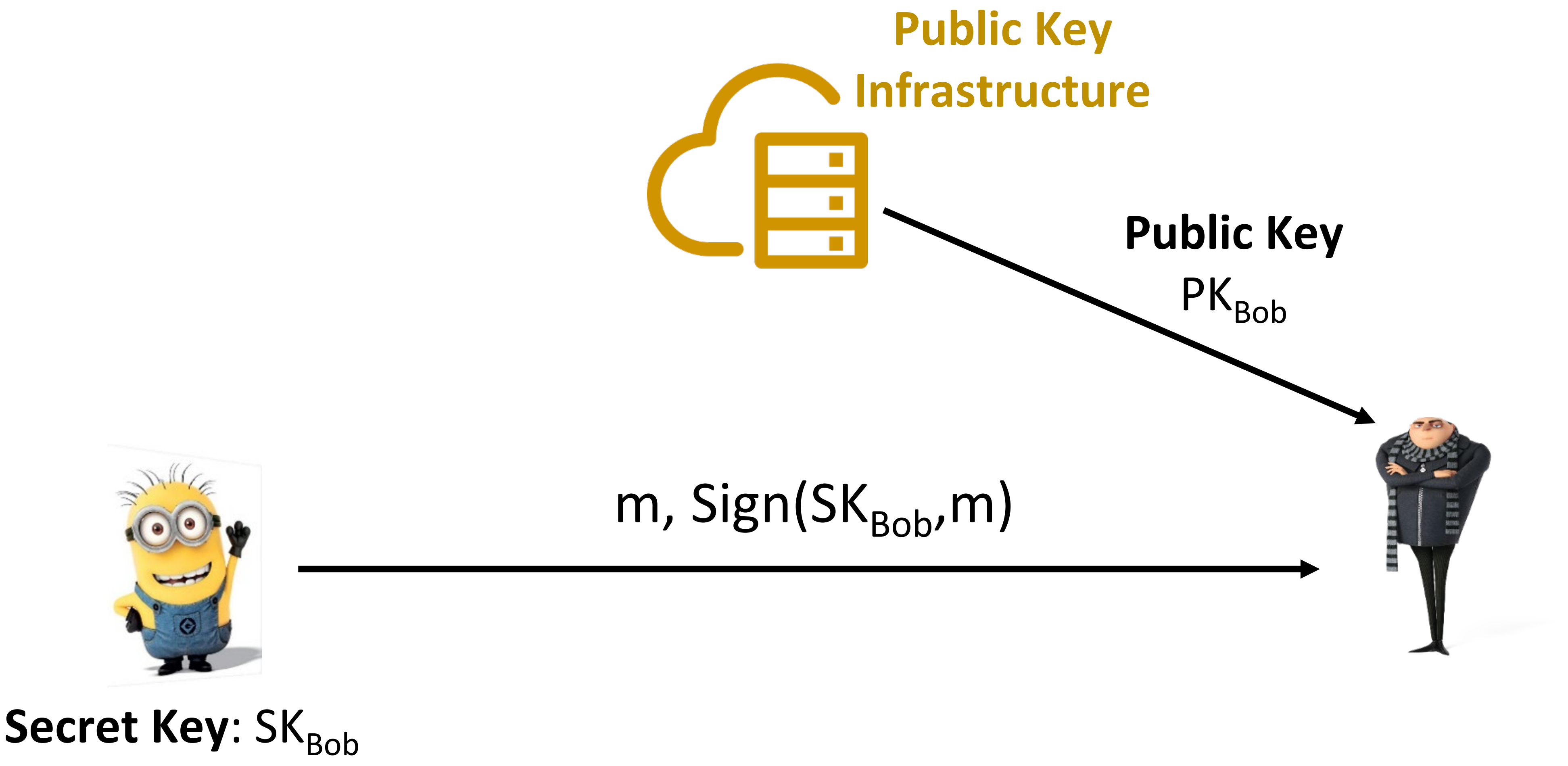
There are not good “cipher modes”

In practice

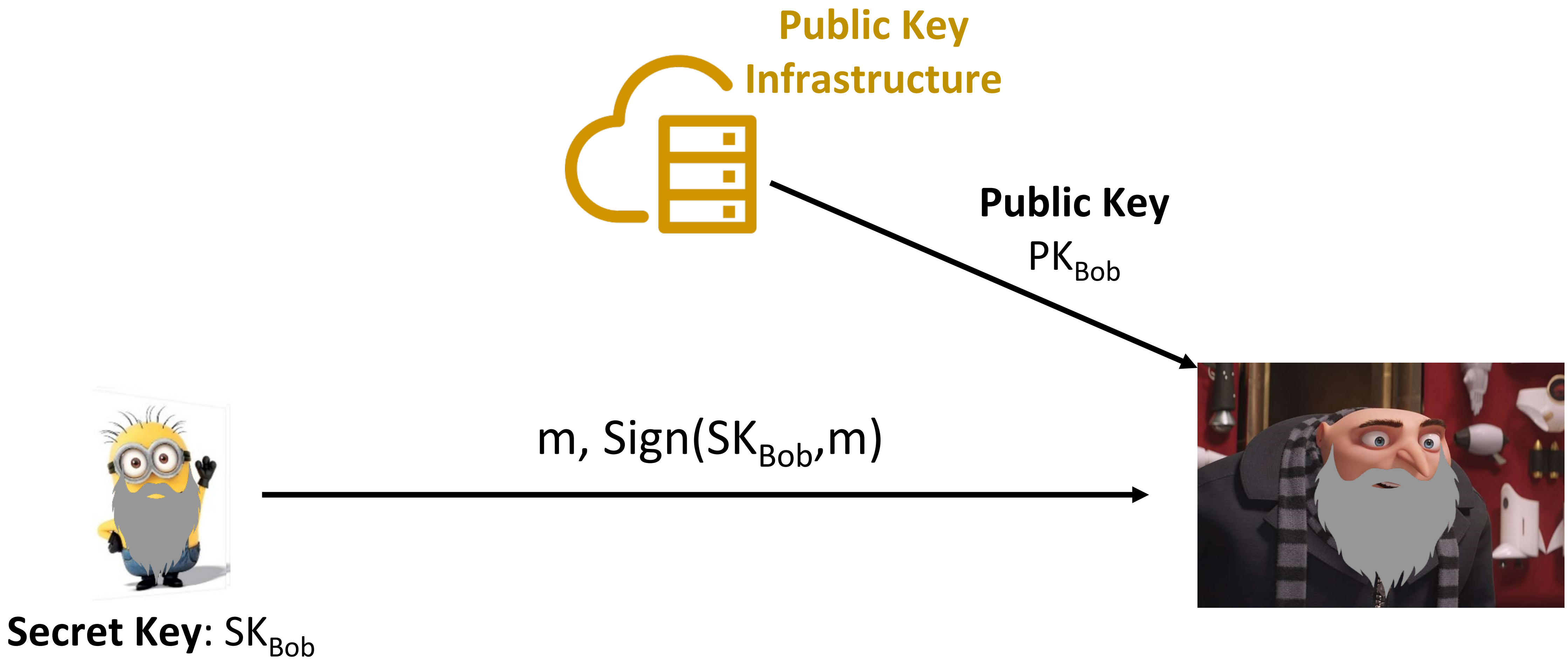
Sign hash of messages

Hybrid encryption
(only encrypt small symmetric key)

Digital signatures on hash functions



Digital signatures on hash functions



Digital signatures on hash functions

Refresher

~~**PRE-IMAGE RESISTANCE**~~

~~Given H(m), difficult to get m~~

SECOND PRE-IMAGE RESISTANCE

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

COLLISION RESISTANCE

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

Public Key Infrastructure

Public Key

PK_{Bob}

I want to make sure I am talking to

$h=H(m)$

Secret Key: SK_{Bob}

m, Sign(SK_{Bob},h)

$h = H(m)$

Verify(PK_{Bob}, h, Sign(SK_{Bob},h))= Yes or No

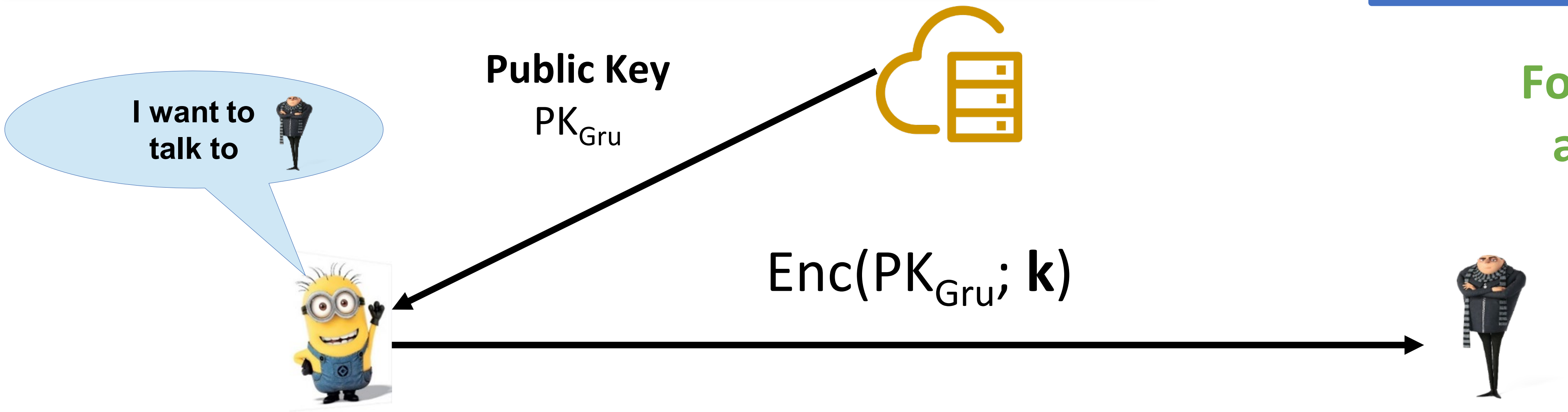
Hybrid encryption

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

Don't design your own

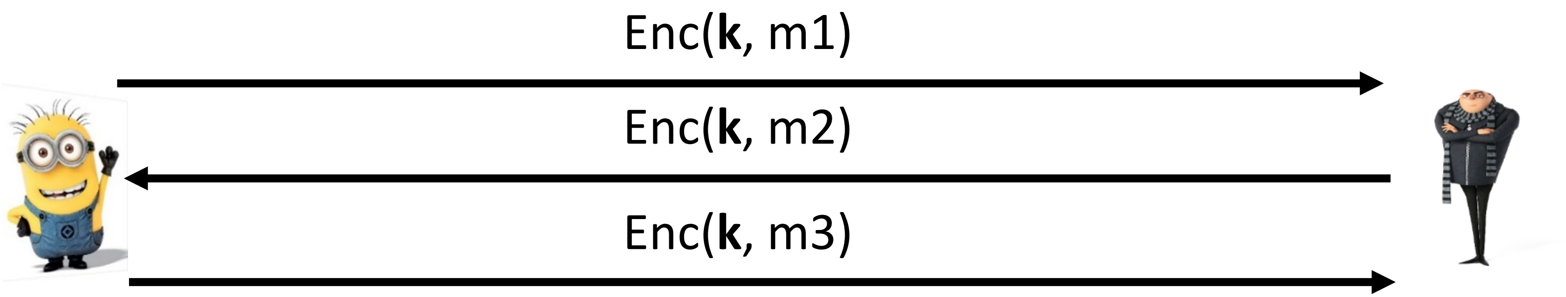
NOT SO SIMPLE!
e.g. ISO 9798-3
TLS

Step 1: establish a shared symmetric key k using "key transport"

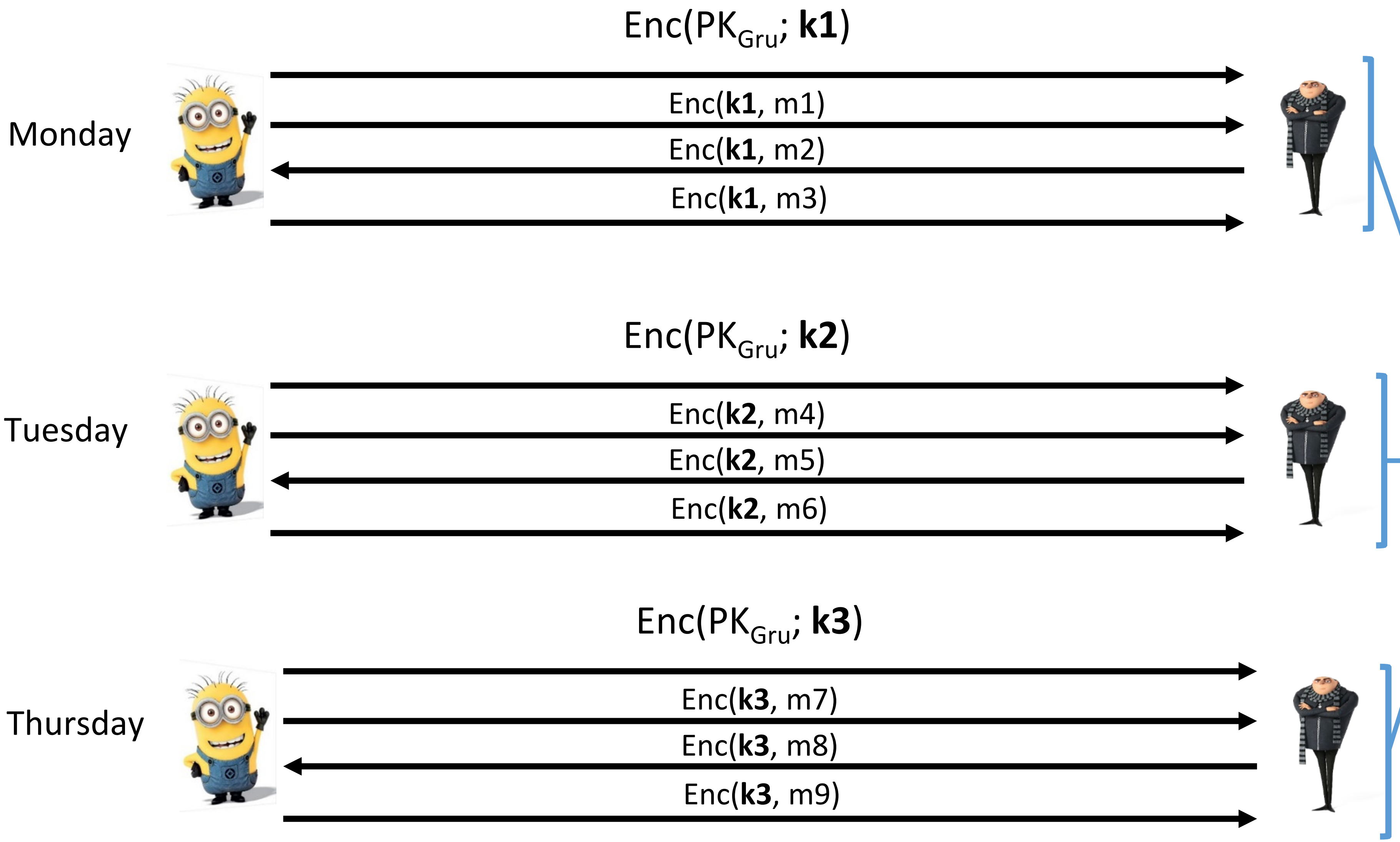


For authentication, add signatures!!

Step 2: use the shared symmetric key k to encrypt the rest of the communication



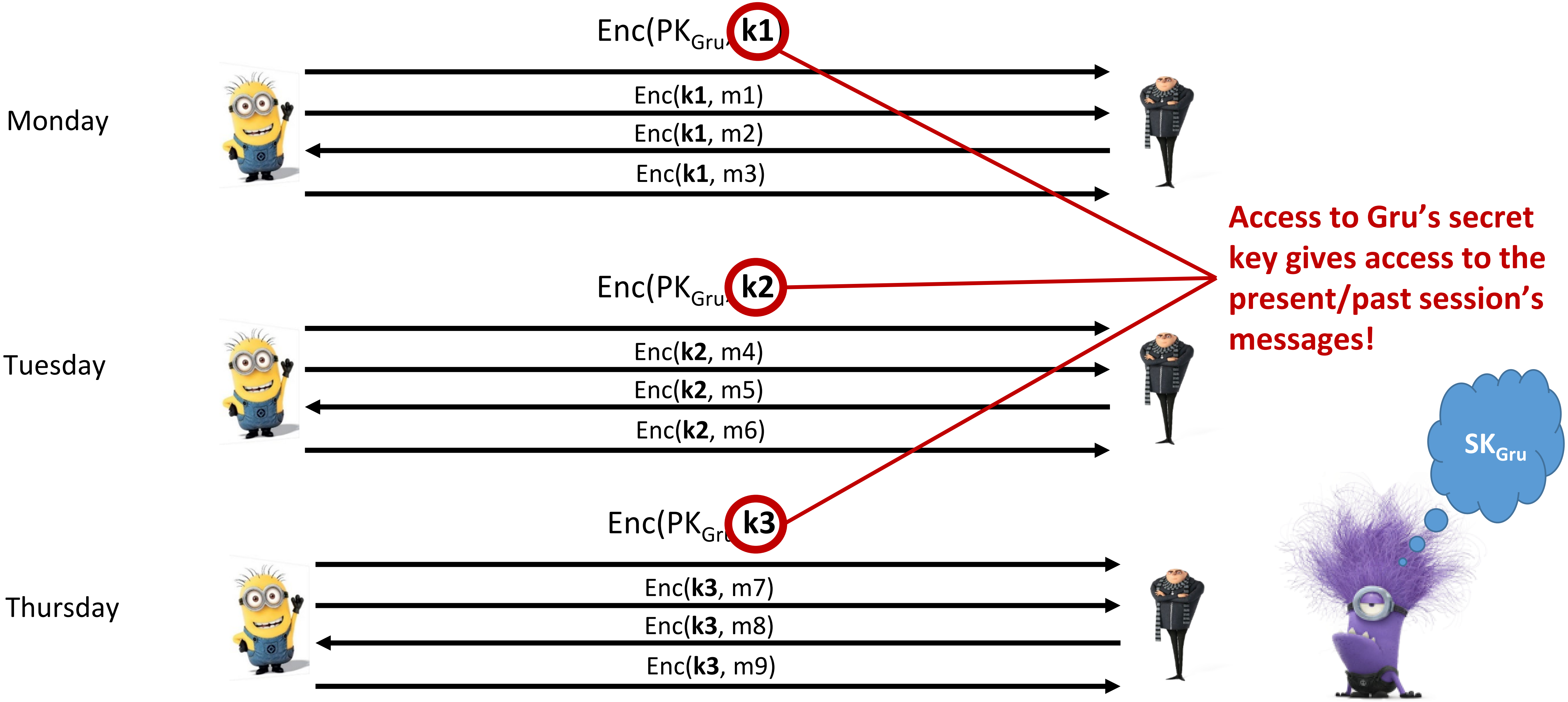
This process is repeated every time Bob wants to talk to Gru



Each of these exchanges with a new key is called "**session**"

The keys k1,k2,... are called "**Session keys**"

What happens if the adversary gets access to Gru's asymmetric key on Thursday?



What happens if the adversary gets access to Gru's asymmetric key on Thursday?

Enc(PK_{Gru}, **k1**)

Monday

Enc(**k1**, m1)

Enc(**k1**, m2)

Enc(**k1**, m3)

Access to Gru's secret key gives access to the present/past session's messages!

Tuesday

DESIRABLE PROPERTY

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

If the adversary learns the key of Thursday's session, Monday and Tuesday should still be safe

Thursday

Enc(**k3**, m7)

Enc(**k3**, m8)

Enc(**k3**, m9)

Key agreement for forward secrecy – The Math

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

Every time Bob wants to talk to Gru…

Shared **public** parameters p , g

Because of the discrete logarithm hardness, an adversary observing these values cannot recover x and y, therefore cannot compute k

P_b

P_a

Secret Key: x (random!)

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

Shared secret!!

$k = g^{xy}$ (mod p)

To encrypt messages for the session

Secret Key: y (random!)

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

Basic Di

Every tim

After the session has ended, delete the secrets x and y.
The key can never be recovered.
Forward secrecy is achieved!!

Shared **public** parameters p , g

Because of the discrete logarithm hardness, an adversary observing these values cannot recover x and y, therefore cannot compute k

P_b →

← **P_a**

Secret Key: x (random!)

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

Shared secret!!
$k = g^{xy}$ (mod p)
To encrypt messages for the session

Secret Key: y (random!)

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

Summary of the crypto lectures

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

Why a MAC should not be constructed as Hash(k||m)

TAs & Carmela: Why are they designing their own crypto?!?!?
Why is this designing Crypto? You go from Hash to MAC.

COM-208 Computer networks!!
“Computer Networking: A Top-Down Approach”: MAC = Hash(k||I am Alice)

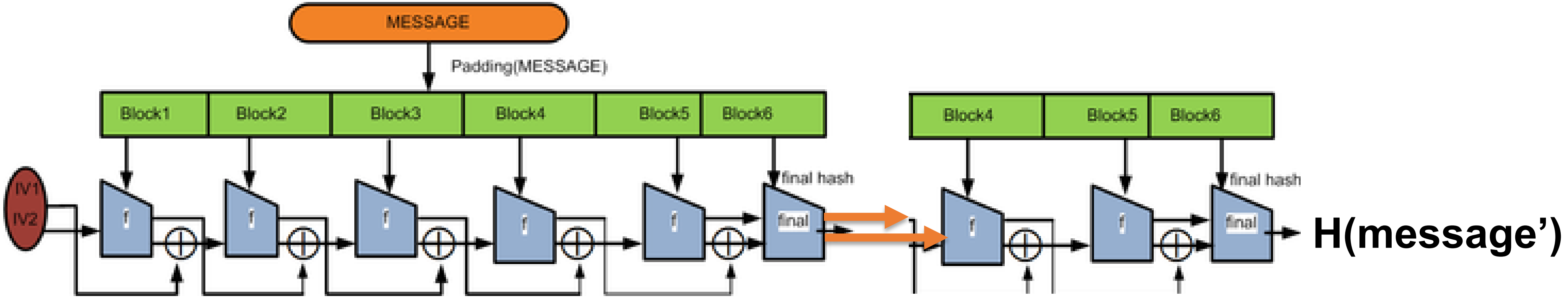
Is it wrong?!?!?!?
Short answer: Not always
So when is it wrong?

Not exam material!!

https://www.whitehatsec.com/blog/hash-length-extension-attacks/

Why a MAC should not be constructed as Hash(k||m)

Many Hash functions (MD5, SHA1, SHA256) are built using **the Merkle-Damgard paradigm**



Hash length extension attacks! (given H(m) obtain H(m||stuff)

Example: H(k||this is Alice) → H(k||this is Alice, First of her name, Queen of the Andals and the First Men)

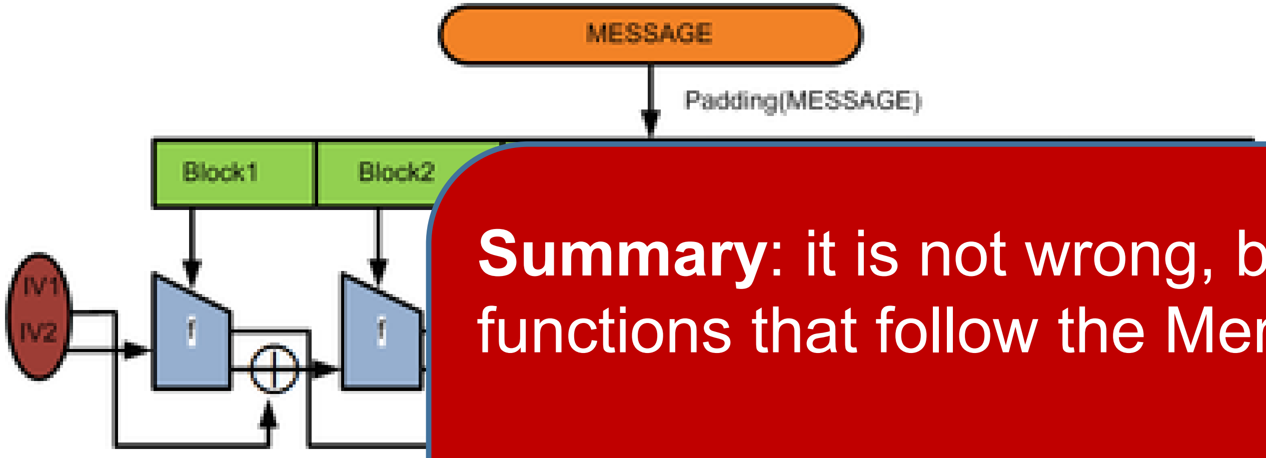
MAC=H(k||Alice) does not guarantee integrity
for Merkle-Damgard hash functions

Not exam material!!

https://www.whitehatsec.com/blog/hash-length-extension-attacks/

Why a MAC should not be constructed as Hash(k||m)

Many Hash functions (MD5, SHA1, SHA256) are built using **the Merkle-Damgard paradigm**



nessage')

Hash le

||stuff)

Example: H(k||this is

e First Men)

Summary: it is not wrong, but there are **many** hash functions that follow the Merkle-Damgard paradigm.

Safe choice: don't do this.

If you want an HMAC, use a standardized one
e.g. https://datatracker.ietf.org/doc/html/rfc2104

MAC=H(k||Alice) does not guarantee integrity
for Merkle-Damgard hash functions

https://www.whitehatsec.com/blog/hash-length-extension-attacks/

Not exam material!!