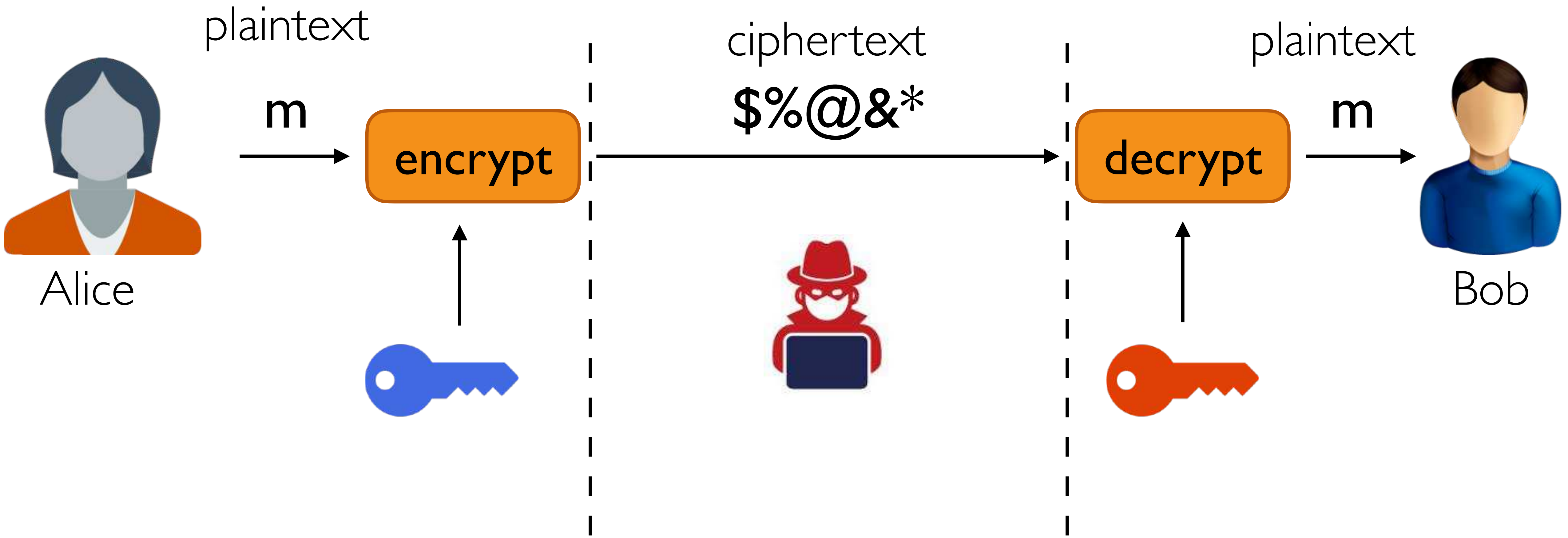
Computer Security: Cryptography module

Dario Fiore, IMDEA Software Institute

Encrypted communication (Confidentiality)



plaintext

m

encrypt

ciphertext

$%@&*

decrypt

plaintext

m

Alice

Bob

encrypt combines plaintext with an *encryption key*

decrypt combines ciphertext with a *decryption key*

—"security" (confidentiality) relies on **secrecy of the key**!—

(i.e. no way to obtain *m* w/o having the proper key)

Kerckhoff’s principle (1883)

security based only on **secrecy of the key**

all protocols/algorithms/methods are publicly known

“Security by obscurity” is BAD

- home-brewed solutions are BAD
- standardized, widely-accepted, extensively and publicly analyzed solutions are GOOD

Private-Key Encryption

Functional definition: two algorithms (E, D)

Encryption algorithm

takes a key K and message (plaintext) and outputs a ciphertext

E(K, m)→c

Decryption algorithm

takes a key and a ciphertext and outputs a message (or perhaps an error)

D(K, c)→m / error

Correctness: for all K,   D(K, E(K, m))=m

Message Authentication Codes (MACs)

Syntax: two algorithms (MAC, VER)

<u>MAC algorithm</u>

takes a key and message (plaintext) and outputs a tag

MAC(K, m) → t

<u>Verification algorithm</u>

takes a key, a message and a tag and outputs accept (1) or reject (0)

VER(K, m, t) → 0/1

Correctness: for all K, VER(K, m, MAC(K, m))=1

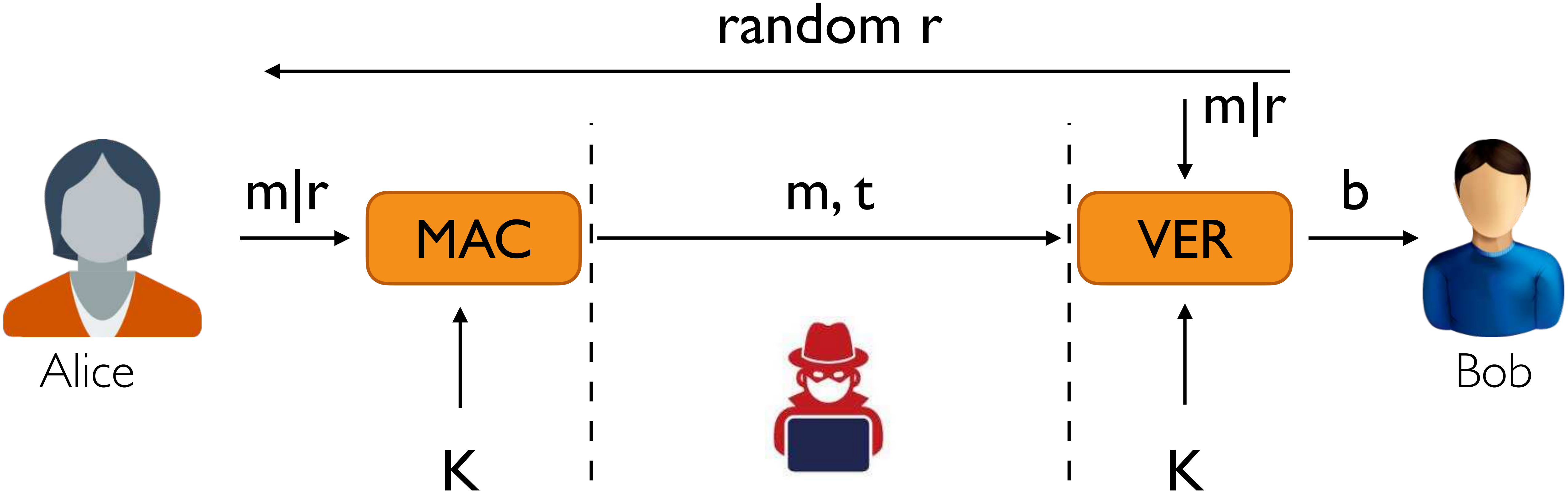
Replay attacks

Attacker can just replay a message-tag pair he has seen

This attack is inherent

Mitigating replay attacks: If replays are really an issue in an application, prevent them using a higher level protocol

Example (to make sure t is generated **now** w.h.p)



Hash Functions

Another very important cryptographic tool

$$H:\{0,1\}^* \longrightarrow \{0,1\}^L$$

map arbitrary-length inputs to fixed-length output

What security property? **Collision resistance**

It must be “hard” to find x≠x’ such that H(x)=H(x’)

Def. A function H is *collision resistant* if every adversary running for some time T (e.g., T=100 years) find x≠x’ such that H(x)=H(x’) with probability at most ε (e.g., $\varepsilon=2^{-80}$).

By the birthday paradox*, a brute-force attack succeeds with probability $\approx 2^{-L/2}$ ⇒ output length is critical

*Birthday paradox: if you sample *n* items from a set of size *T*, if $n \approx \sqrt{T}$ you have probability about 1/2 of finding at least one repeated pair.

Hash Functions

Other security properties

Second-preimage resistance

Given x, it must be "hard" to find x'≠x such that H(x)=H(x')

Weaker than collision resistance (the adversary can't choose x)

Random oracle. H has a "random-looking" output.

Strong heuristic assumption

MACs in practice: HMAC

A direct MAC construction from hash functions

directly handles long messages (no need of modular hash-and-mac)

$$\text{HMAC}(K, m) = H(\ (K' \oplus opad)\ \|\ H(\ (K' \oplus ipad)\ \|\ m)\ )$$

K' key derived from K, ipad/opad specific constants

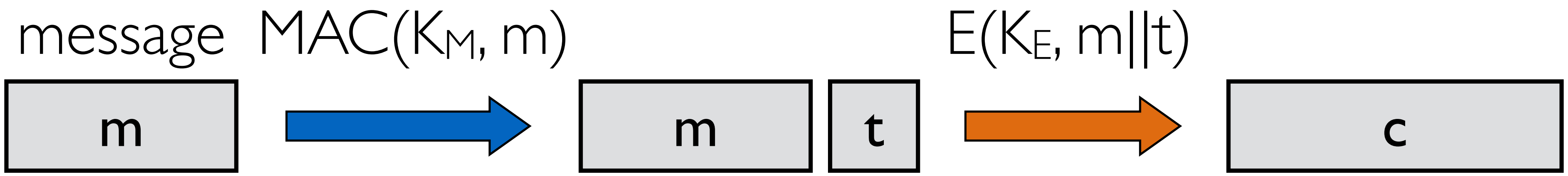
H can be SHA-2 or SHA-3 for example

HMAC is a secure MAC for **variable-length** messages

Authenticated Encryption

Three natural options

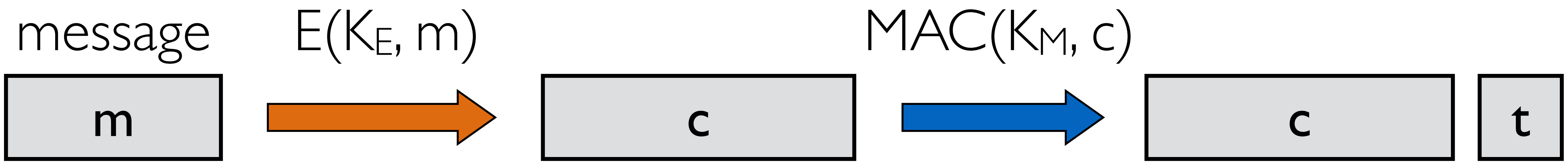
1) MAC-then-Encrypt (used in SSL)



message $\text{MAC}(K_M, m)$ $E(K_E, m||t)$

m m t c

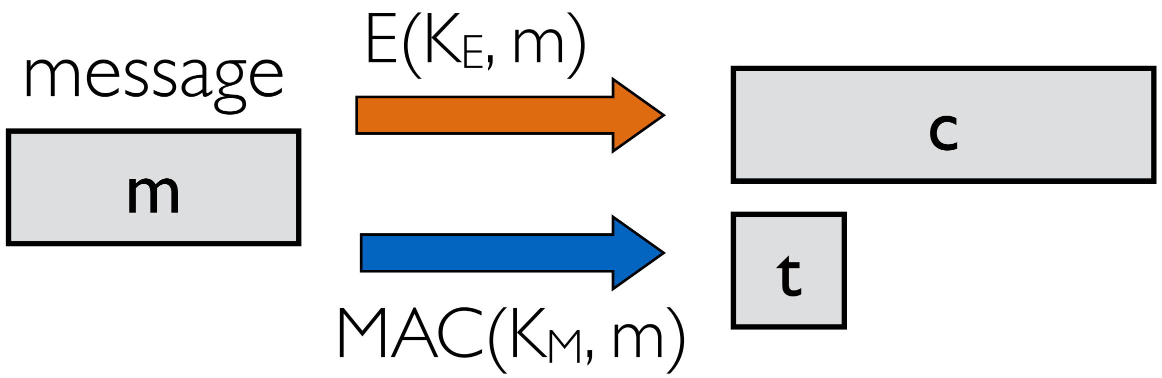
2) Encrypt-then-MAC (used in IPsec)



message $E(K_E, m)$ $\text{MAC}(K_M, c)$

m c c t

3) Encrypt-and-MAC (used in SSH)



message $E(K_E, m)$ $\text{MAC}(K_M, m)$

m c t

Authenticated Encryption in practice

Use recommendations:

- Encrypt-then-MAC
- dedicated modes of operations, e.g., AES-GCM