

# 2550 Intro to cybersecurity

L11: MACs, Public-Key Cryptography

abhi shelat/Ran Cohen

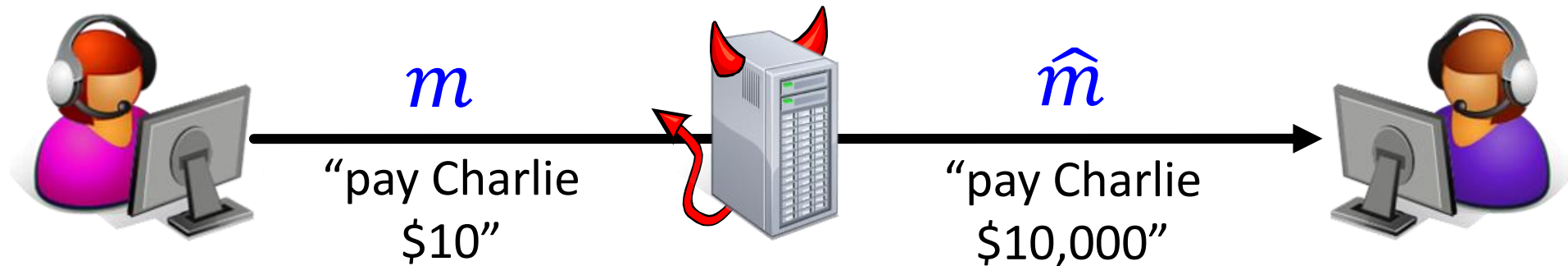
# Outline

- Message Authentication Code (MAC)
  - Authenticating fixed-length messages
  - Authenticating arbitrary-length messages
- Authenticated Encryption
- Public-Key Cryptography

# Message Authentication

## Alice and Bob wish to communicate

- Eve completely controls the channel
- Would like to assure the receiver of a message that it has not been modified



## Encryption vs. authentication

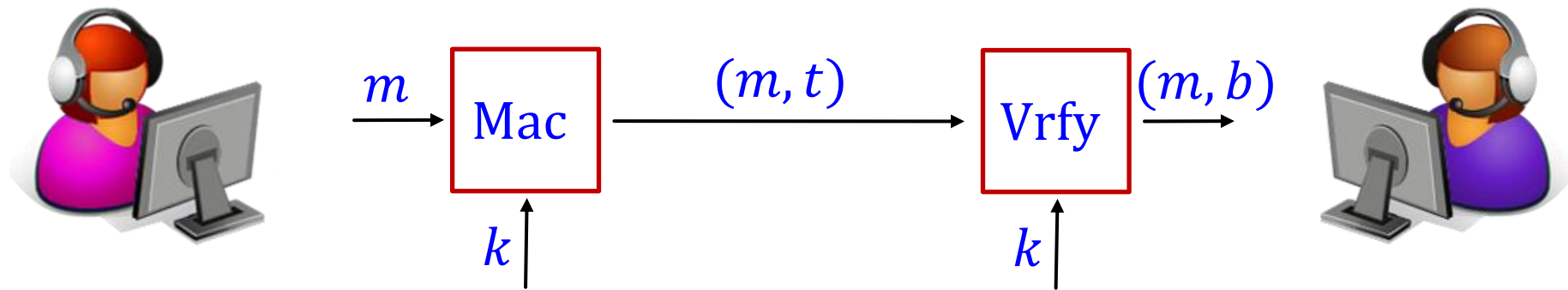
- Encryption = data secrecy
- Authentication = data integrity

Orthogonal aspects: In general, one does not guarantee the other

# Message Authentication Code (MAC)

**Syntax:**  $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$

- Key-generation algorithm  $\text{Gen}$  on input  $1^n$  outputs a key  $k$
- Tag-generation algorithm  $\text{Mac}$  takes a key  $k$  and a message  $m \in \{0,1\}^*$  and outputs a tag  $t \in \{0,1\}^*$
- Verification algorithm  $\text{Vrfy}$  takes a key  $k$ , a message  $m$ , a tag  $t$ , and outputs a bit  $b$



**Correctness:**  $\forall k, m$

$$\text{Vrfy}_k(m, \text{Mac}_k(m)) = 1$$

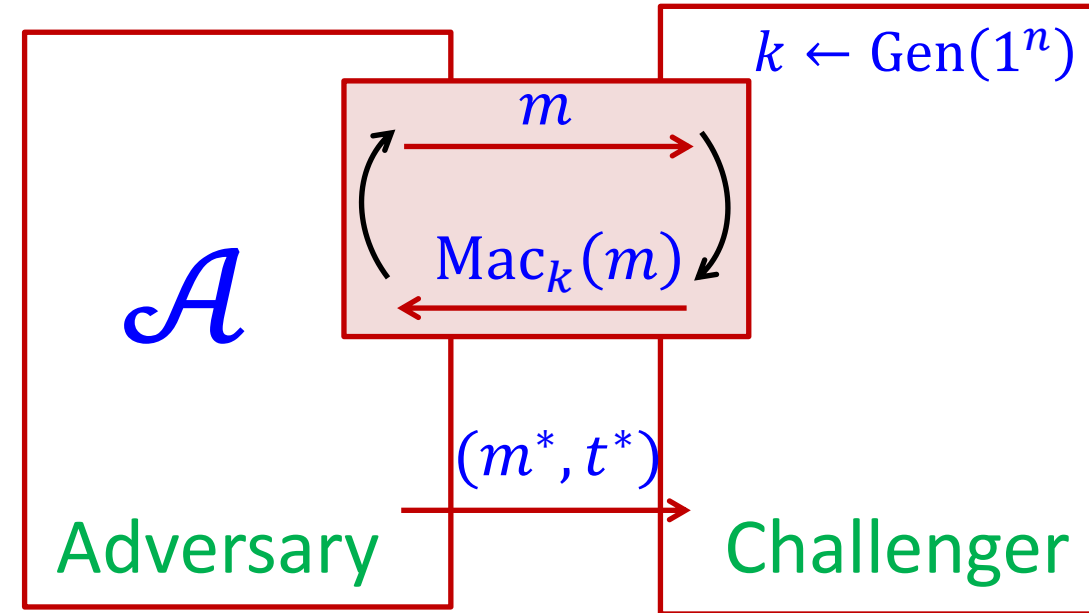
# The Security of MACs

- $\mathcal{A}$  can adaptively ask for tags of messages of its choice
- $\mathcal{A}$  tries to forge a **valid tag** on a **new message**

## Definition:

A MAC scheme  $\Pi$  is secure if for every PPT adversary  $\mathcal{A}$  there exists a negligible function  $\nu(\cdot)$  such that

$$\Pr[\text{MacForge}_{\Pi, \mathcal{A}}(n) = 1] \leq \nu(n)$$



$Q$  = Set of all queries asked by  $\mathcal{A}$

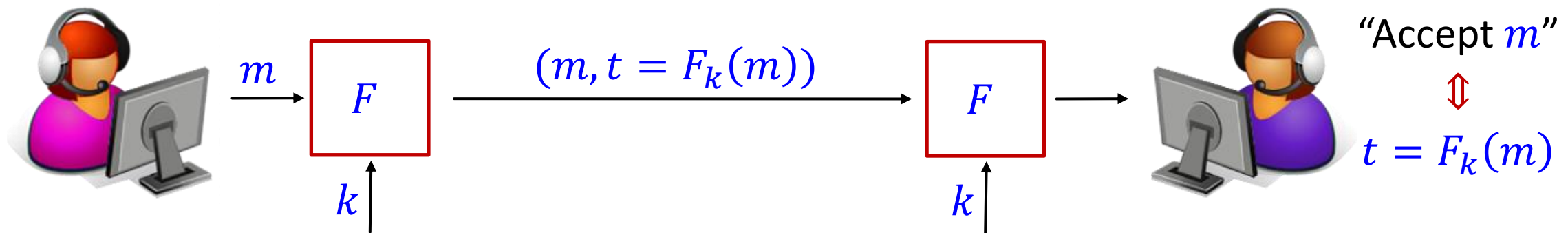
$$\text{MacForge}_{\Pi, \mathcal{A}}(n) = \begin{cases} 1, & \text{if } \text{Vrfy}_k(m^*, t^*) = 1 \\ & \text{and } m^* \notin Q \\ 0, & \text{otherwise} \end{cases}$$

- Does not prevent “replay attacks”!

# A Fixed-Length MAC

Let  $F: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$  be a PRF

- **Key generation:** Sample  $k \leftarrow \{0,1\}^n$
- **Tag generation:** On input  $k \in \{0,1\}^n$  and  $m \in \{0,1\}^n$  output  $t = F_k(m)$
- **Verification:** On input  $k \in \{0,1\}^n$ ,  $m \in \{0,1\}^n$ , and  $t \in \{0,1\}^n$ , output **1** if  $t = F_k(m)$  and **0** otherwise



# A Fixed-Length MAC

Let  $F: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$  be a PRF

- **Key generation:** Sample  $k \leftarrow \{0,1\}^n$
- **Tag generation:** On input  $k \in \{0,1\}^n$  and  $m \in \{0,1\}^n$  output  $t = F_k(m)$
- **Verification:** On input  $k \in \{0,1\}^n$ ,  $m \in \{0,1\}^n$ , and  $t \in \{0,1\}^n$ ,  
output  $1$  if  $t = F_k(m)$  and  $0$  otherwise

## Theorem:

If  $F$  is a PRF then the above MAC scheme is secure

# Authenticating Arbitrary-Length Messages

$$m = \boxed{m_1 \quad m_2 \quad \quad \dots \quad \quad \dots \quad \quad m_d}$$

Given  $\widehat{\Pi} = (\widehat{\text{Gen}}, \widehat{\text{Mac}}, \widehat{\text{Vrfy}})$  for fixed-length messages,  
define  $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$  for arbitrary-length messages as follows:

**Attempt 1:** authenticate each block on its own

- $\text{Gen} = \widehat{\text{Gen}}$
- $\text{Mac}_k((m_1, \dots, m_d)) = (t_1, \dots, t_d)$  where  $t_i = \widehat{\text{Mac}}_k(m_i)$
- $\text{Vrfy}_k((m_1, \dots, m_d), (t_1, \dots, t_d)) = 1 \iff \widehat{\text{Vrfy}}_k(m_i, t_i) = 1$   
for every  $i \in \{1, \dots, d\}$

**Completely insecure...**

- If  $t = (t_1, t_2)$  is a valid tag for  $m = (m_1, m_2)$   
then  $t^* = t_1$  is a valid tag for  $m^* = m_1$



# Authenticating Arbitrary-Length Messages

$$m = \boxed{m_1 \quad m_2 \quad \quad \dots \quad \quad \dots \quad \quad m_d}$$

Given  $\widehat{\Pi} = (\widehat{\text{Gen}}, \widehat{\text{Mac}}, \widehat{\text{Vrfy}})$  for fixed-length messages,  
define  $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$  for arbitrary-length messages as follows:

**Attempt 2:** authenticate the length  $d$  as well

- $\text{Gen} = \widehat{\text{Gen}}$
- $\text{Mac}_k((m_1, \dots, m_d)) = (t_1, \dots, t_d)$  where  $t_i = \widehat{\text{Mac}}_k(d, m_i)$
- $\text{Vrfy}_k((m_1, \dots, m_d), (t_1, \dots, t_d)) = 1 \iff \widehat{\text{Vrfy}}_k((d, m_i), t_i) = 1$   
for every  $i \in \{1, \dots, d\}$

**Still completely insecure...**

- If  $t = (t_1, t_2)$  is a valid tag for  $m = (m_1, m_2)$   
then  $t^* = (t_2, t_1)$  is a valid tag for  $m^* = (m_2, m_1)$

# Authenticating Arbitrary-Length Messages

$$m = \boxed{m_1 \quad m_2 \quad \quad \dots \quad \quad \dots \quad \quad m_d}$$

Given  $\widehat{\Pi} = (\widehat{\text{Gen}}, \widehat{\text{Mac}}, \widehat{\text{Vrfy}})$  for fixed-length messages,  
define  $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$  for arbitrary-length messages as follows:

**Attempt 3:** authenticate the index of each block as well

- $\text{Gen} = \widehat{\text{Gen}}$
- $\text{Mac}_k((m_1, \dots, m_d)) = (t_1, \dots, t_d)$  where  $t_i = \widehat{\text{Mac}}_k(d, i, m_i)$
- $\text{Vrfy}_k((m_1, \dots, m_d), (t_1, \dots, t_d)) = 1 \iff \widehat{\text{Vrfy}}_k((d, i, m_i), t_i) = 1$   
for every  $i \in \{1, \dots, d\}$

**Still completely insecure...**

- If  $t = (t_1, t_2)$  is a valid tag for  $m = (m_1, m_2)$   
and  $t' = (t'_1, t'_2)$  is a valid tag for  $m' = (m'_1, m'_2)$   
then  $t^* = (t_1, t'_2)$  is a valid tag for  $m^* = (m_1, m'_2)$

# Authenticating Arbitrary-Length Messages

$$m = \boxed{m_1 \quad m_2 \quad \dots \quad m_d}$$

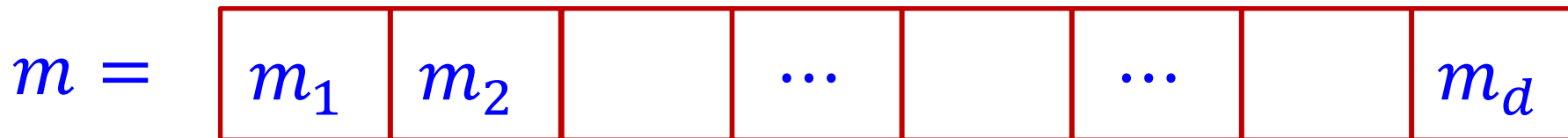
Given  $\widehat{\Pi} = (\widehat{\text{Gen}}, \widehat{\text{Mac}}, \widehat{\text{Vrfy}})$  for fixed-length messages,  
define  $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$  for arbitrary-length messages as follows:

**Solution 1:** sample a random  $r$  for each message  $m$

- $\text{Gen} = \widehat{\text{Gen}}$
- $\text{Mac}_k(m) = (r, t_1, \dots, t_d)$  where  $t_i = \widehat{\text{Mac}}_k(r, d, i, m_i)$  and  $r \leftarrow \{0, 1\}^n$
- $\text{Vrfy}_k((m_1, \dots, m_d), (r, t_1, \dots, t_d)) = 1 \iff \widehat{\text{Vrfy}}_k((r, d, i, m_i), t_i) = 1$   
for every  $i \in \{1, \dots, d\}$

Drawback: Long tags

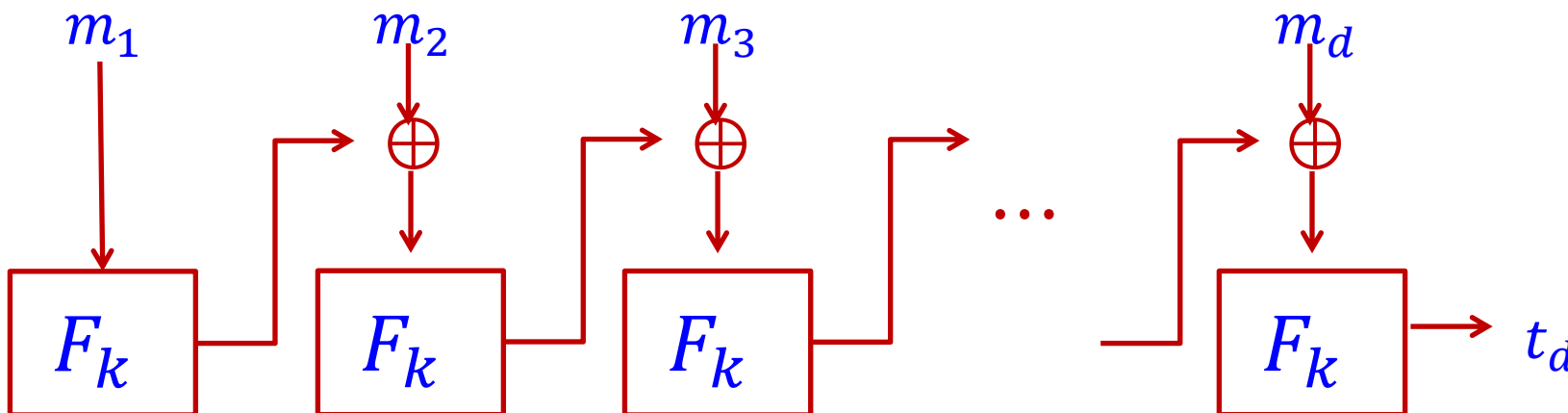
# Authenticating Arbitrary-Length Messages



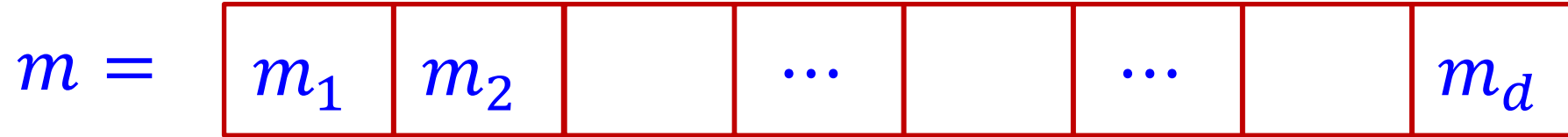
**Solution 2 (CBC-MAC):**  $\text{Mac}_k(m) = t_d$  where

- $t_0 = 0^n$  and  $t_i = F_k(t_{i-1} \oplus m_i)$  for  $i = 1, \dots, d$
- $F_k$  can be any PRF
- $d$  must be fixed ahead of time

In practice: AES as a PRF  
(block length is 128 bits)



# Authenticating Arbitrary-Length Messages



Use a collision-resistant hash function to compress  $m$  into a short “fingerprint”  $H(m)$   
Authenticate  $H(m)$  instead of  $m$

## Solution 3 (“Hash-and-Authenticate”):

- $\text{Gen} = \widehat{\text{Gen}}$
- $\text{Mac}_k(m) = \widehat{\text{Mac}}_k(H(m))$
- $\text{Vrfy}_k(m, t) = 1 \iff \widehat{\text{Vrfy}}_k(H(m), t) = 1$

Must be hard to find  $m \neq m'$   
such that  $H(m) = H(m')$

## HMAC:

- Designed in 1996
- Relies only on a hash function (not on PRF)
- Widely used in standards: TLS, IPSec, SSH

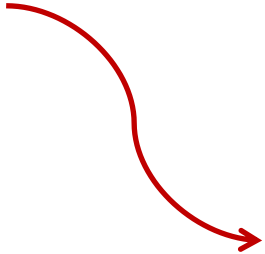
# Outline

- Message Authentication Code (MAC)
  - Authenticating fixed-length messages
  - Authenticating arbitrary-length messages
- Authenticated Encryption
- Public-Key Cryptography

# Encryption vs. Authentication?

**Recall:** CPA-secure encryption from any PRF

$$\text{Enc}_k(m; r) = (r, F_k(r) \oplus m)$$


$$\text{Enc}_k(m \oplus 1^n; r) = (r, F_k(r) \oplus m \oplus 1^n)$$

**An adversary can modify the encrypted message  $m$  without anyone even noticing...**

# Encryption vs. Authentication?

**We want to achieve both:**

**Encryption + Message Authentication  
= Authenticated Encryption**

**...but how?**



# First Attempt

“Encrypt-and-Authenticate”:



$(c, t)$



$$k = (k_E, k_M)$$

$$c \leftarrow \text{Enc}_{k_E}(m)$$

$$t \leftarrow \text{Mac}_{k_M}(m)$$

$$k = (k_E, k_M)$$

$$m \leftarrow \text{Dec}_{k_E}(c)$$

$$? \leftarrow \text{Vrfy}_{k_M}(m, t)$$

**May be completely insecure!**

- $t \leftarrow \text{Mac}_{k_M}(m)$  may completely reveal  $m$ ...
- In general, MACs have no secrecy guarantees

# Second Attempt

“Encrypt-then-Authenticate”:



$(c, t)$



$k = (k_E, k_M)$   
 $c \leftarrow \text{Enc}_{k_E}(m)$   
 $t \leftarrow \text{Mac}_{k_M}(c)$

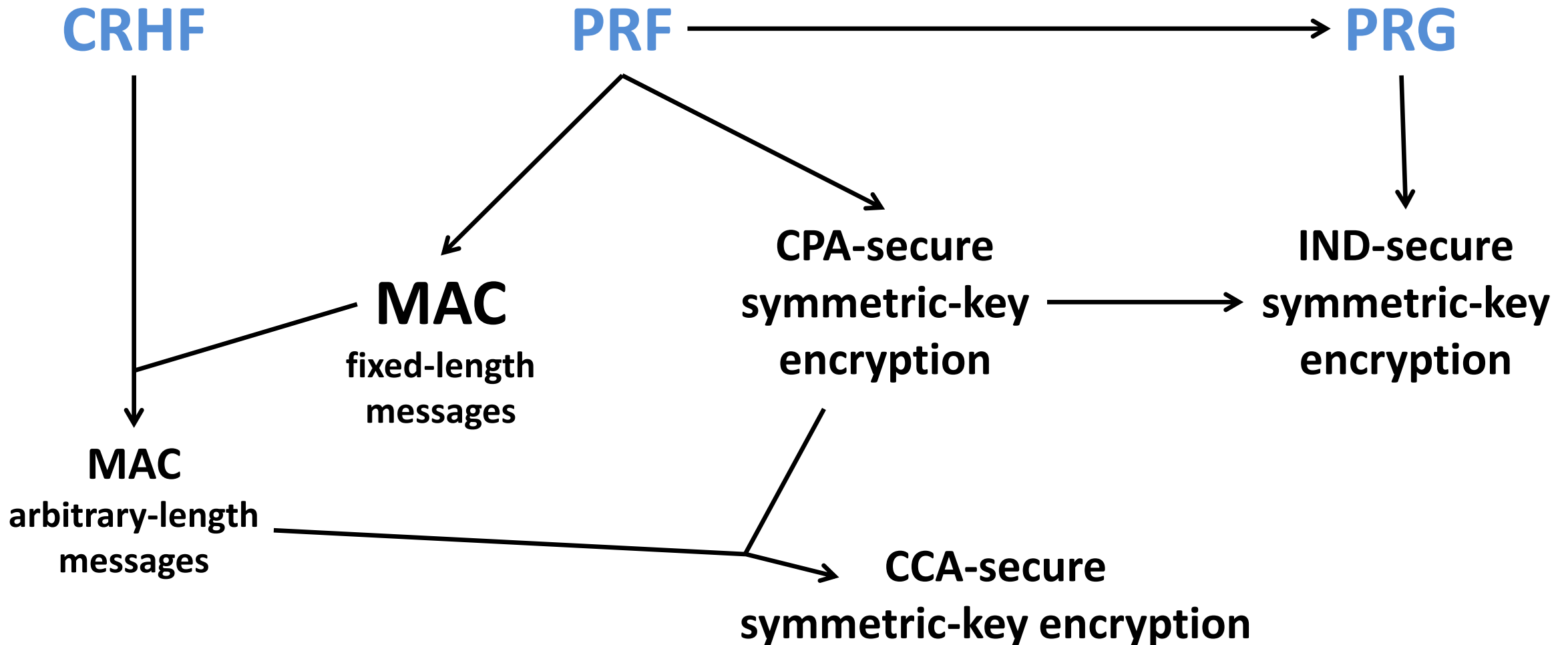
Even better: Infeasible to  
generate any new valid  
ciphertext!

$k = (k_E, k_M)$   
 $m \leftarrow \text{Dec}_{k_E}(c)$   
 $? \leftarrow \text{Vrfy}_{k_M}(c, t)$

**If the encryption is CPA-secure and the MAC is secure, then:**

- The construction is a CPA-secure encryption scheme
- The construction is a secure MAC

# The World of Crypto Primitives (so far)

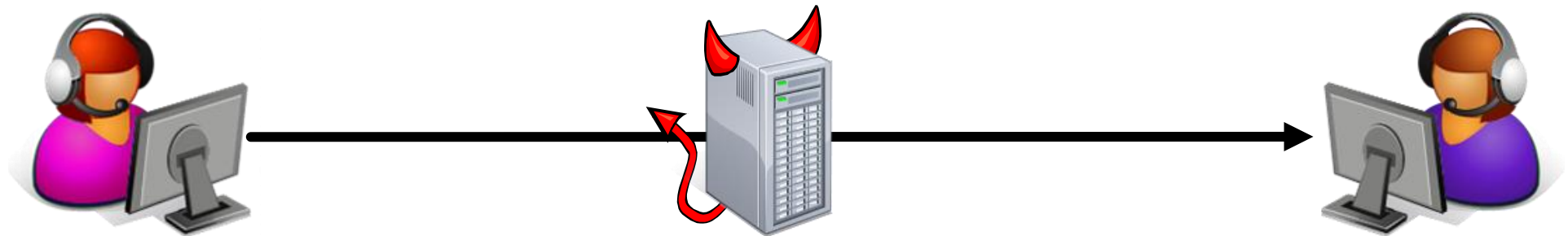


# Outline

- Message Authentication Code (MAC)
  - Authenticating fixed-length messages
  - Authenticating arbitrary-length messages
- Authenticated Encryption
- Public-Key Cryptography

# Public-Key Cryptography

- If Alice and Bob want to use symmetric-key encryption/MACs they must agree on the secret key
- How can they do that??



# Public-Key Cryptography

- If Alice and Bob want to use symmetric-key encryption/MAC they must agree on the secret key
- How can they do that??

644

IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-22, NO. 6, NOVEMBER 1976

## New Directions in Cryptography

*Invited Paper*

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

**Abstract**—Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.

The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel. In order to use cryptography to insure privacy, however, it is currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such as private courier or registered mail. A private conversation



Turing Award 2015

# Public-Key Cryptography

Diffie and Hellman discussed 3 problems:

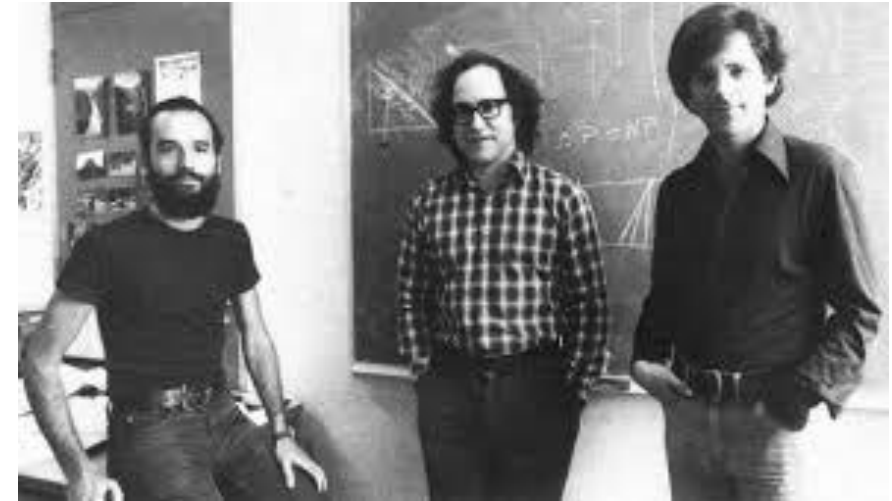
- **Key agreement:**
  - Alice and Bob agree on a secret key over an insecure channel
- **Public-key encryption:**
  - Everybody can encrypt using a public encryption key
  - Decryption requires a secret decryption key
- **Digital signatures:**
  - Alice can sign a message using a secret signing key
  - Everybody can verify using a public verification key



	Secrecy	Integrity
<b>Symmetric cryptography (private-key schemes)</b>	Private-key encryption	MAC
<b>Asymmetric cryptography (public-key schemes)</b>	Public-key encryption	Digital signatures

# Public-Key Cryptography

- Private-key cryptography is extremely efficient in practice, relies on heuristics
- Public-key cryptography is less efficient, relies on hardness of mathematical problems
- 1976 Diffie and Helman constructed a key-agreement protocol relying on the hardness of the discrete-log problem
- 1977 Rivest, Shamir, and Adleman constructed public-key encryption & digital signatures relying on the hardness of factoring
- 1984 ElGamal noticed that the KA protocol of DH essentially gives public-key encryption



Turing Award 2002



# Public-Key Encryption

**Syntax:**  $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$

- Key-generation algorithm  $\text{Gen}$  on input  $1^n$  outputs a public encryption key  $pk$  and a secret decryption key  $sk$
- $\text{Enc}$  takes the public key  $pk$  and a message  $m$  and outputs a ciphertext  $c$
- $\text{Dec}$  takes the secret key  $sk$  and a ciphertext  $c$ , and outputs a plaintext  $m$

**Correctness:**  $\forall m$  and  $(pk, sk) \leftarrow \text{Gen}(1^n)$   
$$\text{Dec}_{sk}(\text{Enc}_{pk}(m)) = m$$

# Public-Key Encryption

## Syntax: $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$

- Key-generation algorithm  $\text{Gen}$  on input  $1^n$  outputs a public encryption key  $pk$  and a secret decryption key  $sk$
- $\text{Enc}$  takes the public key  $pk$  and a message  $m$  and outputs a ciphertext  $c$
- $\text{Dec}$  takes the secret key  $sk$  and a ciphertext  $c$ , and outputs a plaintext  $m$

## Security:

defined via the following experiment

- A public-key encryption scheme must be CPA-secure
- A public-key encryption scheme immediately gives a key-agreement protocol

