# Introduction to Modern Cryptography

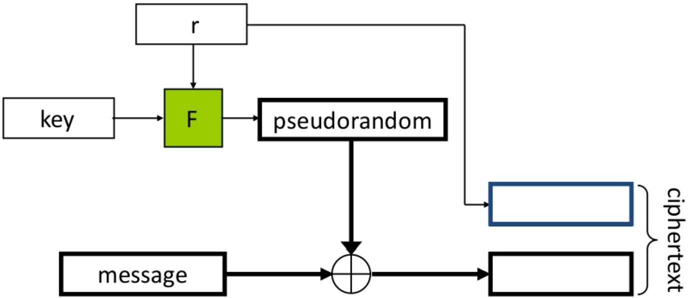
Michele Ciampi

(Slides courtesy of Prof. Jonathan Katz)

Lecture 10, Part 1

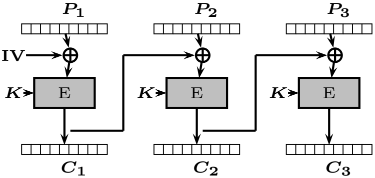
# Message Integrity

# CPA-secure Encryption for Short Messages (Recall)

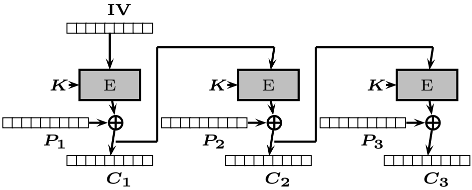



- Not solve OTP limitation 1 (key as long as the message)
- Solves OTP limitation 2 (key used only once)
- $\implies$ CPA-secure $\implies$ EAV-secure

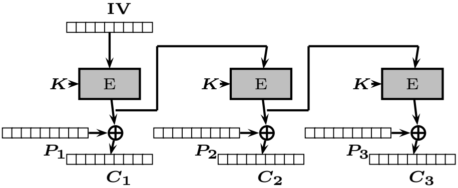
CBC



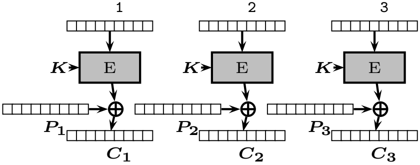

CFB




OFB




CTR



# So Far

The described scheme based on **PRF/block cipher** in a given **mode of operation**:

- Solves OTP limitation 1 (key as long as the message)
- Solves OTP limitation 2 (key used only once)
- EAV-secure (single-message secrecy)
- CPA-secure (multiple message secrecy)
- **Not CCA-secure**

# CCA vs. CPA (Recall)

- CPA: $\boldsymbol{A}$ has access to **encryption oracle**
- CCA: $\boldsymbol{A}$ has access to **decryption oracle**
  - in addition to access to an **encryption oracle**

- CCA attacks are a real problem: **Padding-Oracle Attack**
- None of the schemes we have seen so far is CCA-secure

CCA related to the ability of the attacker to make **undetected (predictable) changes to the ciphertext** (cf. malleability)

# Secrecy vs. Integrity

- So far concerned with **secrecy of communication**
- What about **integrity**?
- Integrity ensures that a received message:
  1. originated from **the intended sender**, and
  2. was **not modified**
- Standard error-correction not enough:
  - Not concerned with random errors
  - Concerned with **malicious, intended "errors"**

# Passive Attacks vs. Active Attacks

### Passive Attacks

So far considered only **passive (i.e. eavesdropping) attacks**

- Attacker simply observes the channel

### Active Attacks

In the setting of integrity, explicitly consider **active attacks**

- Attacker has full control over the channel

# Passive Attacks vs. Active Attacks

## Passive Attacks

So far considered only **passive (i.e. eavesdropping) attacks**

- Attacker simply observes the channel

## Active Attacks

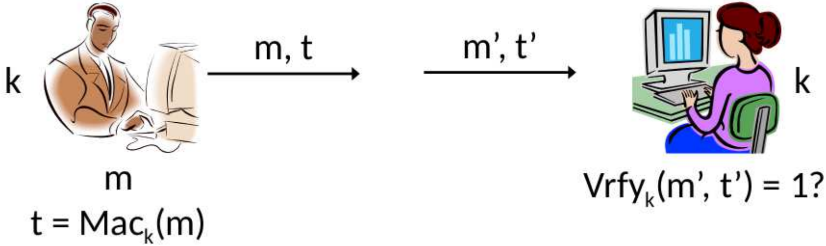
In the setting of integrity, explicitly consider **active attacks**

- Attacker has full control over the channel

## MAC

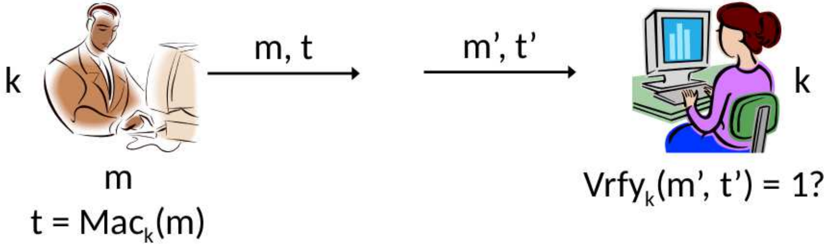
The right tool for integrity protection against active attacks is a **message authentication code (MAC)**

# Message Integrity Using a MAC: Scenario 1



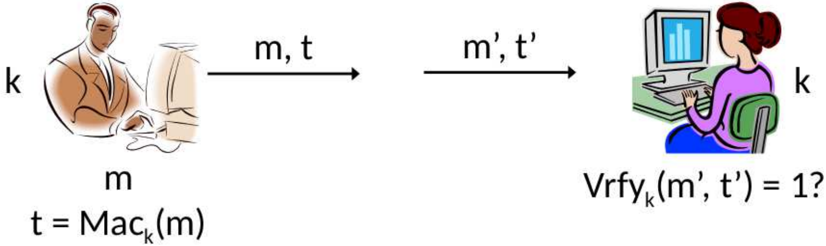

- $\boldsymbol{A}$ and $\boldsymbol{B}$ share a key $\boldsymbol{k}$
- $\boldsymbol{A}$ computes a **tag** $\boldsymbol{t} = \mathsf{Mac}_{\boldsymbol{k}}(\boldsymbol{m})$
- $\boldsymbol{A}$ sends $(\boldsymbol{m}, \boldsymbol{t})$ to $\boldsymbol{B}$

# Message Integrity Using a MAC: Scenario 1




- $B$ receives $(m', t')$ and **verifies the tag** $\mathsf{Vrfy}_k(m', t')$
- If $\mathsf{Vrfy}_k(m', t') = 1 \implies m$ was not modified

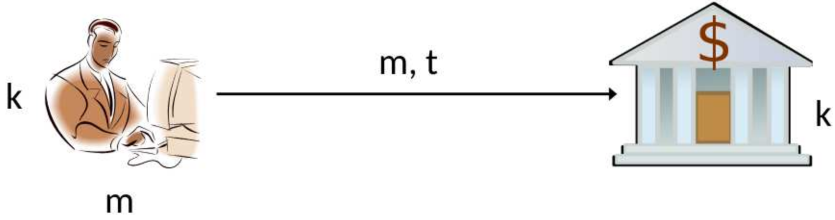
# Message Integrity Using a MAC: Scenario 1




**Observe**

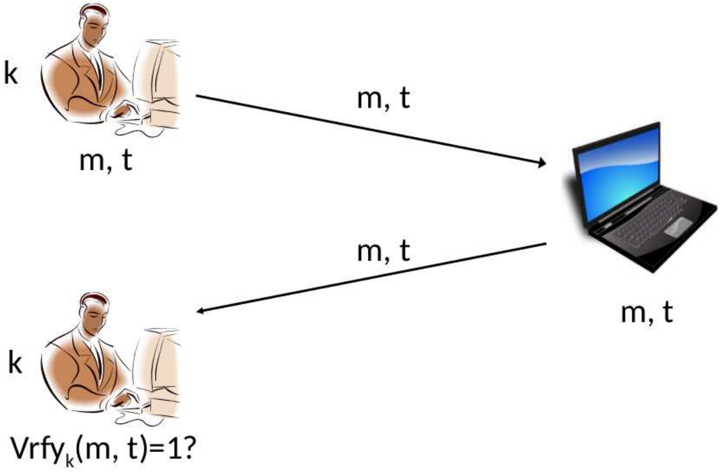
- **Not concerned with secrecy**
- Message $m$ transmitted in the clear

# Message Integrity Using a MAC: Scenario 2




- $\boldsymbol{A}$ shares key $\boldsymbol{k}$ with his bank
- $\boldsymbol{A}$ transmits $\boldsymbol{m} =$"*Send* **100** *GBP to* $\boldsymbol{C}$"
- If $\boldsymbol{C}$ modifies $\boldsymbol{m'} =$"*Send* **1000** *GBP to* $\boldsymbol{C}$" the bank will detect the modification due to the MAC

# Message Integrity Using a MAC: Scenario 3




***A*** authenticates his own ***m*** to himself at different points in time

# Secrecy vs. Integrity

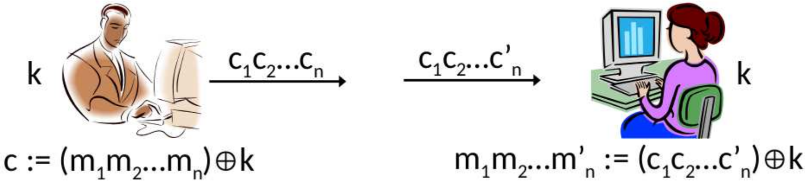
**Secrecy and integrity are orthogonal concerns**

- Possible to have either one without the other
- Sometimes you might want one without the other
- Most often, both are needed

**Encryption alone does not provide integrity**

- Related to the property of **malleability**
- None of the schemes so far provide any integrity

# Malleability (Recall)




- The OTP is perfectly secret, but is still malleable
- Encryption under OTP does not imply integrity
- **Encryption does not provide message auth.**

# Message Authentication Code (MAC)

## MAC

A message authentication code is defined by three PPT algorithms $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$:

- $\mathsf{Gen}$: takes as input $\mathbf{1}^{\boldsymbol{n}}$; outputs $\boldsymbol{k}$. (Assume $|\boldsymbol{k}| \geq \boldsymbol{n}$)
- $\mathsf{Mac}$: takes as input key $\boldsymbol{k}$ and message; outputs a tag $\boldsymbol{t}$

$$\boldsymbol{t} \leftarrow \mathsf{Mac}_{\boldsymbol{k}}(\boldsymbol{m})$$

- $\mathsf{Vrfy}$: takes key $\boldsymbol{k}$, message $\boldsymbol{m}$, and tag $\boldsymbol{t}$; outputs $\mathbf{1}$ (*accept*) or $\mathbf{0}$ (*reject*)
- Correctness: $\forall \boldsymbol{m}$ and $\forall \boldsymbol{k}$ output by $\mathsf{Gen}$:

$$\mathsf{Vrfy}_{\boldsymbol{k}}(\boldsymbol{m}, \mathsf{Mac}_{\boldsymbol{k}}(\boldsymbol{m})) = \mathbf{1}$$

# MAC Security

## Threat model

**Adaptive chosen-message attack**

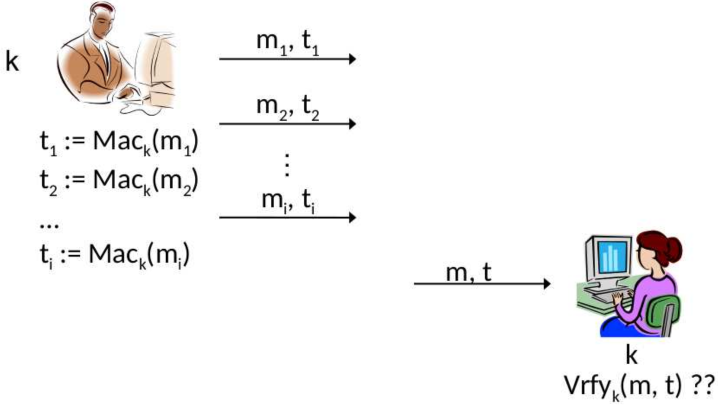
- Assume the attacker can induce the sender to authenticate messages of the attacker's choice

## Security goal

**Existential unforgeability**

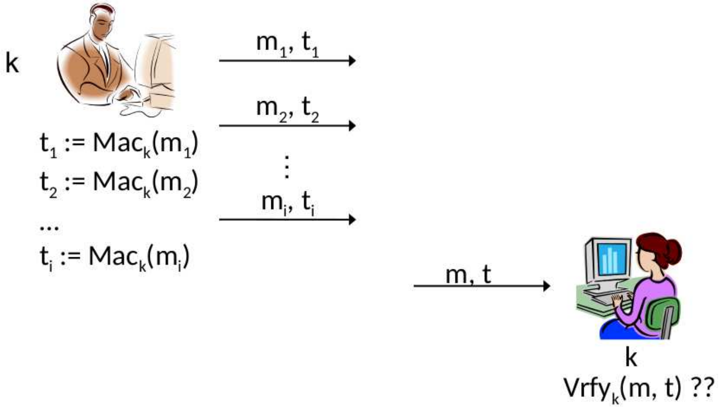
- Attacker should not be able to forge a valid tag on any message not previously authenticated by the sender

# MAC Security



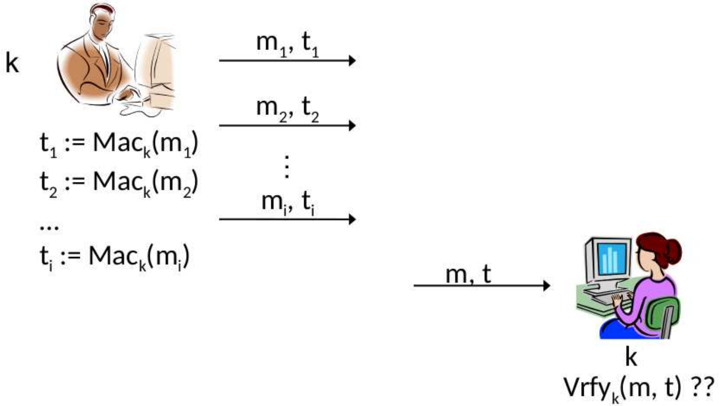

Attacker $\boldsymbol{A}$ induces the sender to authenticate messages $\boldsymbol{m_1, \ldots, m_i}$ of his choice

# MAC Security




$\boldsymbol{A}$ stores the corresponding tags $\boldsymbol{t_1, \ldots, t_i}$

# MAC Security




It should be infeasible for $\boldsymbol{A}$ to generate a new $\boldsymbol{(m, t)}$ **:**
$\forall \boldsymbol{i}$ **:** $\boldsymbol{m \neq m_i}$ s.t. $\mathsf{Vrfy}_{\boldsymbol{k}}\boldsymbol{(m, t) = 1}$

# Is the Definition too Strong?

## MAC Security

- We don't want to make any assumptions about what the sender might authenticate
- We don't want to make any assumptions about what forgeries are **meaningful**
  - What is *meaningful* is application dependent!
- $\Longrightarrow$ enough if a forgery exists i.e. **existential** as opposed to **meaningful** forgery

A MAC satisfying this definition can be used in **any context** where integrity is needed

# MAC Security: Formal Definition

**$\mathsf{Forge}_{A,\Pi}(n)$**

Fix $A, \Pi$. Define randomized experiment $\mathsf{Forge}_{A,\Pi}(n)$:

- $k \leftarrow \mathsf{Gen}(1^n)$
- $A$ interacts with an oracle $\mathsf{Mac}_k(\cdot)$:
  - $A$ submits $m_1, \ldots, m_i$ to $\mathsf{Mac}_k(\cdot)$
  - $A$ collects back $t_1, \ldots, t_i$ from $\mathsf{Mac}_k(\cdot)$
  - Let $M = \{m_1, \ldots, m_i\}$ be the set of messages submitted to the oracle
- $A$ outputs $(m, t)$
- $A$ succeeds, and the experiment evaluates to $1$, if $\mathsf{Vrfy}_k(m, t) = 1$ and $m \notin M$

# Security for MACs

$\mathbf{\Pi}$ is secure if for all PPT attackers $\boldsymbol{A}$, there is a negligible function $\boldsymbol{\epsilon}$ such that:

$$\mathbf{Pr}[\mathsf{Forge}_{\boldsymbol{A},\mathbf{\Pi}}(\boldsymbol{n}) = \mathbf{1}] \leq \boldsymbol{\epsilon}(\boldsymbol{n})$$

# Security for MACs

$\mathbf{\Pi}$ is secure if for all PPT attackers $\boldsymbol{A}$, there is a negligible function $\boldsymbol{\epsilon}$ such that:

$$\mathbf{Pr}[\mathsf{Forge}_{\boldsymbol{A},\mathbf{\Pi}}(\boldsymbol{n}) = \mathbf{1}] \leq \boldsymbol{\epsilon}(\boldsymbol{n})$$

Compare to definitions of secure encryption e.g. CPA:

$$\mathbf{Pr}[\mathsf{PrivK}^{\mathbf{cpa}}_{\boldsymbol{A},\mathbf{\Pi}}(\boldsymbol{n}) = \mathbf{1}] \leq \frac{\mathbf{1}}{\mathbf{2}} + \boldsymbol{\epsilon}(\boldsymbol{n})$$

# Security for MACs

$\mathbf{\Pi}$ is secure if for all PPT attackers $\boldsymbol{A}$, there is a negligible function $\boldsymbol{\epsilon}$ such that:

$$\mathbf{Pr}[\mathsf{Forge}_{\boldsymbol{A},\mathbf{\Pi}}(\boldsymbol{n}) = \mathbf{1}] \leq \boldsymbol{\epsilon}(\boldsymbol{n})$$

Compare to definitions of secure encryption e.g. CPA:

$$\mathbf{Pr}[\mathsf{PrivK}^{\mathbf{cpa}}_{\boldsymbol{A},\mathbf{\Pi}}(\boldsymbol{n}) = \mathbf{1}] \leq \frac{\mathbf{1}}{\mathbf{2}} + \boldsymbol{\epsilon}(\boldsymbol{n})$$

Secure MAC $\implies$ infeasible to forge **even a single message**

# Replay Attacks

## Replay Attack

A message from previous communication is captured and re-transmitted (replayed) at a later point in time

## Warning!

- MACs do not prevent **replay attacks**
- The tag on the original message is valid $\implies$ the tag on the replayed message is also valid
- **No stateless mechanism can prevent replay attacks**

# Replay Attacks

- Replay attacks are often a significant real-world concern
- e.g. Attacker $\boldsymbol{A}$ replays ten times the message $\boldsymbol{m} =$ *"Send* **100** *GBP to* $\boldsymbol{A}$*"*
- Need to protect against replay attacks at a higher level
- Decision about what to do with a replayed message is application-dependent

# End

References: Sec. 4.1, 4.2 (up to replay attacks).