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)
- $\Rightarrow$ CPA-secure $\Rightarrow$ 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: $A$ has access to encryption oracle
- CCA: $A$ 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**

<table>
<thead>
<tr>
<th>So far considered only <strong>passive (i.e. eavesdropping) attacks</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Attacker simply observes the channel</td>
</tr>
</tbody>
</table>

**Active Attacks**

<table>
<thead>
<tr>
<th>In the setting of integrity, explicitly consider <strong>active attacks</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Attacker has full control over the channel</td>
</tr>
</tbody>
</table>
### 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

- A and B share a key $k$
- A computes a tag $t = \text{Mac}_k(m)$
- A sends $(m, t)$ to B

$Vrfy_k(m', t') = 1?$
Message Integrity Using a MAC: Scenario 1

- $B$ receives $(m', t')$ and verifies the tag $\text{Vrfy}_k(m', t')$
- If $\text{Vrfy}_k(m', t') = 1 \implies m$ 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

- A shares key $k$ with his bank
- A transmits $m = “Send 100 \ GBP \ to \ C”$
- If $C$ modifies $m’ = “Send 1000 \ GBP \ to \ C”$ 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 (Gen, Mac, Vrfy):

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

\[ t \leftarrow \text{Mac}_k(m) \]

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

\[ \text{Vrfy}_k(m, \text{Mac}_k(m)) = 1 \]
### 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 $A$ induces the sender to authenticate messages $m_1, \ldots, m_i$ of his choice
MAC Security

\[ \begin{align*}
  & k \\
  & t_1 := \text{Mac}_k(m_1) \\
  & t_2 := \text{Mac}_k(m_2) \\
  & \vdots \\
  & t_i := \text{Mac}_k(m_i) \\
  & m_1, t_1 \\
  & m_2, t_2 \\
  & \vdots \\
  & m_i, t_i \\
  & m, t \\
  & \kappa \\
  & \text{Vrfy}_k(m, t) \ ??
\end{align*} \]

A stores the corresponding tags \( t_1, \ldots, t_i \)
MAC Security

It should be infeasible for $A$ to generate a new $(m, t)$:

\[ \forall i : m \neq m_i \text{ s.t. } \text{Vrfy}_k(m, t) = 1 \]
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!
- 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

**Forge**

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

- \( k \leftarrow \text{Gen}(1^n) \)
- \( A \) interacts with an oracle \( \text{Mac}_k(\cdot) \):
  - \( A \) submits \( m_1, \ldots, m_i \) to \( \text{Mac}_k(\cdot) \)
  - \( A \) collects back \( t_1, \ldots, t_i \) from \( \text{Mac}_k(\cdot) \)
  - 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 \( \text{Vrfy}_k(m, t) = 1 \) and \( m \notin M \)
Security for MACs

Π is secure if for all PPT attackers $A$, there is a negligible function $\epsilon$ such that:

$$\Pr[\text{Forge}_{A,\Pi}(n) = 1] \leq \epsilon(n)$$
Security for MACs

Π is secure if for all PPT attackers $A$, there is a negligible function $\epsilon$ such that:

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

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

$$\Pr[\text{PrivK}_{A,\Pi}^{\text{cpa}}(n) = 1] \leq \frac{1}{2} + \epsilon(n)$$
Security for MACs

Π is secure if for all PPT attackers A, there is a negligible function ε such that:

\[ \Pr[\text{Forge}_{A,\Pi}(n) = 1] \leq \epsilon(n) \]

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

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

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 \(\Rightarrow\) 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 $A$ replays ten times the message $m = \text{"Send 100 GBP to } A\text{"}$
- 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).