Lecture 3: Message Authentication Codes

Last time: CPA secure encryption

Today:
1. Recap
2. Motivate and define the notion of message authentication codes (MACs)
3. Construct MACs

Recap:

Definition (informal): A symmetric encryption scheme \((Enc, Dec)\) is said to be secure against adaptive chosen message attack if for any PPT adv \(A\), any messages \(m_1, \ldots, m_t \in M\) and any \(m'_1, \ldots, m'_t \in M\) chosen adaptively by \(A\)

\[ Enc(k, m_1), \ldots, Enc(k, m_t) \approx Enc(k, m'_1), \ldots, Enc(k, m'_t) \]

Construction: Using PRF and one-time pad.

In practice: Using AES and one-time pad with counter mode:

\[ Enc(k, m_1 || \ldots || m_n) = r_r, \{AES(k, r + i) \oplus m_i\}_{i \in [n]} \]
This definition does not provide any form of authentication!
Namely, an adversary may chance the message and the parties may not be able to detect it. This is a security breach!

**Authentication:**

![Diagram showing authentication process between User and Server]

How does the server know that it is Alice who is sending the instruction?

**Message Authentication Codes (MACs)**

Assumes the communicating parties share a secret key $k$.

![Diagram showing message authentication process between Alice and Bob]

Ensures the authenticity of $m$
**Attacker goal:** Existential forgery; i.e., forge a MAC for any message.

Is this goal too strong? Why do we care if the attacker MACs gibberish?

Parties can MAC their secret key (which is gibberish)

**Attacker power:** See MACs for messages of its choice.

**Definition:** A message authentication code consists of a (signing) function $MAC: K \times M \rightarrow \{0,1\}^n$ with the following security guarantee:

For any $PPT$ adversary $A$, it wins in the following game with only negligible probability.

<table>
<thead>
<tr>
<th>Challenger</th>
<th>$A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_1$</td>
<td>$MAC(k, m_1)$</td>
</tr>
<tr>
<td>$m_2$</td>
<td>$MAC(k, m_2)$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$m_k$</td>
<td>$MAC(k, m_k)$</td>
</tr>
</tbody>
</table>

$A$ outputs $(m^*, t^*)$
Remark: More generally, one can define a MAC as two algorithms: a signing algorithm $\text{Sig}: K \times M \rightarrow \{0,1\}^n$ and a verification algorithm $\text{Ver}: K \times M \times \{0,1\}^n \rightarrow \{0,1\}$, such that for every $k \in K$ and $m \in M$,

$$\text{Ver}(k, m, \text{Sig}(k, m)) = 1$$

We chose to define a single algorithm ($\text{MAC}$) since that is the case in practice. In the public key setting we will define it via two algorithms as above (stay tuned!).

**Definition:** A $\text{MAC}: K \times M \rightarrow \{0,1\}^n$ is secure against adaptive chosen message attacks if any $PPT$ adversary wins in the above game with only negligible probability.

**Constructions:** Use a PRF!

$$\text{MAC}(k, m) = \text{PRF}(m)$$

**Note:** The value of a PRF is required to look random, whereas the value of a MAC is required to be unpredictable (both given oracle access to the function). The latter is weaker if the output size is large.
**Theorem:** Every PRF with domain $D$ and range $R$, such that $1/|R|$ is negligible is a $MAC$ with message space $D$.

**Corollary:** $AES$ is a secure $MAC$ for messages in $\{0,1\}^{128}$.

**Authenticating arbitrarily long messages:**

**Attempt 1:** Partition the message into smaller blocks and $MAC$ each block separately.

\[ m = (m_1, \ldots, m_n) \in D^n \quad \Rightarrow \quad MAC(k, m_1), \ldots, MAC(k, m_n) \]

**Insecure!** Can execute a mix and match attack.

**Attempt 2:** Partition the message into smaller blocks and $MAC$ each block using a chaining:

\[
\begin{align*}
    m_1 &\quad \rightarrow \quad f_k \\
    m_2 &\quad \rightarrow \quad m_3 \oplus f_k \\
    m_3 &\quad \rightarrow \quad m_4 \oplus f_k \\
    \vdots &\quad \vdots \\
    m_n &\quad \rightarrow \quad tag
\end{align*}
\]
Insecure! Can execute an extension attack.

Given a tag for $m$, denoted by $tag$, and given a tag for $m'$, denoted by $tag'$, one can generate a tag for $m'||tag' \oplus m$. The tag is $tag$.

**Final fix:** Choose two independent and random keys $k, k^* \in K$.

This $MAC$ is known as **Cipher Block Changing (CBC) MAC**, and secure against adaptive chosen message attack if $f$ is a **PRF**.
MAC with improved efficiency: Galois MAC (GMAC)

Basic idea: Use the same chaining structure as above, but instead of using a PRF (i.e., AES), use a one-time secure MAC, and encrypt the tag using AES.

Namely: Instead of using AES, use the multiplication function

\[ M_H : \{0,1\}^{128} \rightarrow \{0,1\}^{128} \]
defined by \( M_H(x) = H \cdot x \) where \( H \in \{0,1\}^{128} \) and multiplication is in the Galois field \( GF[2^{128}] \)

\[
H = AES(k, 0)
\]

\[
tag = (iv, AES(k, iv) \oplus tag_1)
\]
Note that $MAC(k, m)$ is computed as follows:

1. Parse $m = m_1 || ... m_n$.
2. Compute $v = \sum_i m_i H^i$ over $GF[2^{128}]$
3. Output an encryption of $H \cdot (v \oplus n)$

**Efficiency gain:** $H^i$ can be precomputed, and multiplication is more efficient than AES.