Lecture 24: MAC for Arbitrary Length Messages
Recall

Previous lecture, we constructed MACs for fixed length messages

- The GGM Pseudo-random Function (PRF) Construction
  - **Given.** Pseudo-random Generator (PRG) 
    \[ G : \{0, 1\}^k \rightarrow \{0, 1\}^{2^k} \]
  - **We Constructed.** PRF \( F_{sk}(m) \) using the GGM construction from the domain \( \{0, 1\}^n \) to the range \( \{0, 1\}^k \)

- The MAC scheme was provided by \((\text{Gen}, \text{Mac}, \text{Ver})\)

  - \(\text{Gen}()\). Return \( sk \leftarrow \{0, 1\}^k \)
  - \(\text{Mac}_{sk}(m)\). Return \( \tau = F_{sk}(m) \)
  - \(\text{Ver}_{sk}(\tilde{m}, \tilde{\tau})\). Return \( \tilde{\tau} == F_{sk}(\tilde{m}) \)

- This construction is secure only for fixed-length messages
- We need different secret key for every length of the message. Otherwise, we can perform length-extension attacks
Goal of this Lecture

- Suppose we are given a MAC scheme \((Gen', Mac', Ver')\) that is secure only for \(B\)-bit messages and generates tags of length \(k\).
- We want to construct a MAC scheme \((Gen, Mac, Ver)\) that is secure for arbitrary length messages.
First Proposal I

The students proposed the following construction

\begin{align*}
\text{Gen}(). \text{ Return } \sk = \text{Gen}'() \\
\text{Mac}_\sk(m). \\
\quad 1 \quad \text{Pad the message } m \text{ so that the length of } m \text{ is a multiple of } B \\
\quad 2 \quad \text{Break } m \text{ into } n/B \text{ blocks } (m(1), m(2), \ldots, m(n/B)) \text{ such that each block } m(i) \text{ has size } B \\
\quad 3 \quad \text{For } i \in \{1, 2, \ldots, n/B\}: \text{ Compute } \tau(i) = \text{Mac}'_\sk(m(i)) \\
\quad 4 \quad \text{Return } \tau = (\tau(1), \tau(2), \ldots, \tau(n/B)) \\
\text{Ver}_\sk(\tilde{\tau}). \\
\quad 1 \quad \text{Let } \tilde{m} = (\tilde{m}(1), \tilde{m}(2), \ldots, \tilde{m}(t)) \text{ (each block of length } B) \\
\quad 2 \quad \text{Let } \tilde{\tau} = (\tilde{\tau}(1), \tilde{\tau}(2), \ldots, \tilde{\tau}(t)) \text{ (each block of length } k) \\
\quad 3 \quad \text{Return true if and only if for all } i \in \{1, \ldots, t\}, \text{ the expression } \text{Ver}'_{\sk}(\tilde{m}(i), \tilde{\tau}(i)) \text{ is true}
\end{align*}
Attacks on the Scheme. This scheme is insecure. The students proposed the following attacks. Suppose the adversary sees the message 
\[ m = (m^{(1)}, m^{(2)}, \ldots, m^{(n/B)}) \] and the tag 
\[ \tau = (\tau^{(1)}, \tau^{(2)}, \ldots, \tau^{(n/B)}) \].

1. Permuting.
The adversary can construct \( m' \) from \( m \) by arbitrarily permuting the blocks of the message \( m \). The adversary can construct the corresponding \( \tau' \) from \( \tau \) by performing the same permutation of blocks on the tag \( \tau \).
Shortening/Extending.
The adversary can construct \( m' \) from \( m \) by dropping blocks from \( m \). The adversary can construct the corresponding \( \tau' \) from \( \tau \) by dropping the tags for those blocks in the tag \( \tau \). The adversary can also construct \( m' \) from \( m \) by repeating some blocks of \( m \). The adversary can construct the corresponding \( \tau' \) from \( \tau \) by repeating the tags for those blocks in the tag \( \tau \).
Suppose the adversary sees another message
\( \hat{m} = (\hat{m}^{(1)}, \hat{m}^{(2)}, \ldots, \hat{m}^{(n/B)}) \) and its tag
\( \hat{\tau} = (\hat{\tau}^{(1)}, \hat{\tau}^{(2)}, \ldots, \hat{\tau}^{(n/B)}) \)

The adversary can construct \( m' \) from the messages \( m \) and \( \hat{m} \)
by splicing blocks. The adversary can construct the corresponding \( \tau' \) from the tags \( \tau \) and \( \hat{\tau} \) by splicing the corresponding blocks from the tags \( \tau \) and \( \hat{\tau} \).

For example, the tag of the message \( (m^{(1)}, \hat{m}^{(2)}, \ldots, \hat{m}^{(n/B)}) \)
is \( (\tau^{(1)}, \hat{\tau}^{(2)}, \ldots, \hat{\tau}^{(n/B)}) \)
Combination of the attacks mentioned above.
For example, the adversary can splice, extend, and permute!
Students proposed the following fixes.

- **Prevent Permutation.** We can include the information in each $\tau^{(i)}$ that it is the tag of the message-block at position $i$, then permutation attacks cannot be performed.

- **Prevent Shortening/Extension.** We can include the information in each $\tau^{(i)}$ that it is the tag of the message of total length $n$, then shortening/extension attacks cannot be performed.

- **Prevent Splicing.** We can include a time-stamp in each $\tau^{(i)}$ so that we cannot splice messages at two different time instances. (We shall use a *slightly different technique* but I wanted to summarize the proposals of the students)
Gen(). Return $sk = \text{Gen}'()$
Secure MAC Scheme for Arbitrary-length Messages II

- $\text{Mac}_{sk}(m)$.
  1. Pad $m$ to make its length a multiple of $B/4$.
  2. Break $m$ into $4n/B$ blocks of size $B/4$ each. We represent this as $m = (m^{(1)}, m^{(2)}, \ldots, m^{(4n/B)})$.
  3. Pick $r \leftarrow \{0, 1\}^{B/4}$.
  4. Let $[n]_2$ represent the $(B/4)$-bit representation of the number $n$ in binary.
  5. Let $[i]_2$ represent the $(B/4)$-bit representation of the number $i$ in binary.
  6. For each $i \in \{1, 2, \ldots, 4n/B\}$ create the following block

$$m^{+(i)} = (r, [n]_2, [i]_2, m^{(i)}).$$

  7. For each $i \in \{1, 2, \ldots, 4n/B\}$ create the tag

$$\tau^{(i)} = \text{Mac}'_{sk}(m^{+(i)}).$$

  8. Return $\tau = (r, \tau^{(1)}, \tau^{(2)}, \ldots, \tau^{(4n/B)})$. 

MAC Long Messages
Secure MAC Scheme for Arbitrary-length Messages III

Ver_{sk}(\tilde{m}, \tilde{\tau}).

1. Interpret \( \tilde{m} = (\tilde{m}^{(1)}, \tilde{m}^{(2)}, \ldots, \tilde{m}^{(t)}) \), where each block is of length \( B \).
2. Interpret \( \tilde{\tau} = (\tilde{r}, \tilde{\tau}^{(1)}, \tilde{\tau}^{(2)}, \ldots, \tilde{\tau}^{(t)}) \), where \( \tilde{r} \) is of length \( B/4 \) and each block \( \tilde{\tau}^{(i)} \) is of length \( k \).
3. Let \( \tilde{n} = t \times (B/4) \), the length of the message \( \tilde{m} \).
4. For each \( i \in \{1, 2, \ldots, t\} \) construct the following block

\[
\tilde{m}^{+ (i)} = (\tilde{r}, [\tilde{n}]_2, [i]_2, \tilde{m}^{(i)})
\]

5. Accept \( (\tilde{m}, \tilde{\tau}) \) if for each \( i \in \{1, 2, \ldots, t\} \) the test \( \text{Ver}_{sk}^{'}(\tilde{m}^{+ (i)}, \tilde{\tau}^{(i)}) \) accepts.
Note that we can assume that all random strings “\( r \)” are unique. Because, by birthday bound, we need to tag roughly \( \sqrt{2^B/4} = 2^{B/8} \) messages before we can expect one collision. If we choose \( B = 800 \), then we will (with high probability) see unique “\( r \)” strings. This, effectively, serves as a “time stamp”

Check that all the attacks that we discussed cannot be performed against this new MAC scheme \((\text{Gen}, \text{Mac}, \text{Ver})\)

**Very Important.** We emphasize that “protecting against known attacks” does not imply security of a MAC scheme. We can formally prove the security of the MAC scheme that we have described above. The proof is beyond the scope of this course.