Lecture 21: Message Authentication Codes

Outline

- In today's lecture, we will learn about Message Authentication Codes (MACs)
- We shall define security notions that we expect from such a primitive
- Finally, we shall construct MACs using random functions that are secure against adversaries with unbounded computational power

Introduction: MAC

- A Message Authentication Scheme (MAC) is a private-key version of signatures involving two parties, the Signer and the Verifier
  - Private-key: This means that the signer and the verifier met yesterday and established a secret-key
  - Signature: This means that the verifier can verify that the signer endorses a particular message, and an eavesdropper cannot forge such endorsements
- Defined by three algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$
  - Secret-key Generation: $\mathsf{sk} = \mathsf{Gen}()$
  - Signing Messages: Compute the tag $\tau = \mathsf{Sign}_{\mathsf{sk}}(m)$
  - The Signer sends (m, τ) to the verifier
  - Verifying Message-tag pairs: $z = \mathsf{Ver}_{\mathsf{sk}}(\widetilde{m}, \widetilde{\tau}) \in \{0, 1\}$. Output $z = 1$ indicates that the message-tag pair is accepted, while output $z = 0$ indicates that the message-tag pair is not accepted.

Pictorial Summary

Signer | Verifier

$\mathsf{sk} = \mathsf{Gen}()$

Yesterday

$\tau = \mathsf{Sign}_{\mathsf{sk}}(m)$

Send (m, τ)

Today

$z = \mathsf{Ver}_{\mathsf{sk}}(m, \tau)$

Comments

- No Message Secrecy: Previously, we saw that primitives like encryption and secret sharing require hiding some information from the adversary. In MACs, the message m is in the clear! We want to ensure that an adversary should not be able to generate tags that verify for new messages.
- Secrecy of the secret key sk: The secrecy of sk is paramount. If the secret-key sk is obtained by an adversary, then the adversary can use the signing algorithm to easily sign arbitrary messages!
  Intuitively, MACs ensure that the "holder of the secret key sk" endorses the message. If the eavesdropper obtains the secret-key, then the eavesdropper can endorse arbitrary messages.

Correctness

- Let the message space be $\mathcal{M}$
- Intuition: We want to ensure that the tag for any message $m \in \mathcal{M}$ that the honest signer generates should always verify
- Mathematically, we can write this as: For every message $m \in \mathcal{M}$, we have

$$\mathbb{P}\left[z = 1 \colon \mathsf{sk} = \mathsf{Gen}(), \tau = \mathsf{Sign}_{\mathsf{sk}}(m), z = \mathsf{Ver}_{\mathsf{sk}}(m, \tau)\right] = 1$$

- English Translation: The probability that $z = 1$ is 1, where the secret-key $\mathsf{sk} = \mathsf{Gen}()$, the tag $\tau = \mathsf{Sign}_{\mathsf{sk}}(m)$, and the output $z = \mathsf{Ver}_{\mathsf{sk}}(m, \tau)$.
- Note that this guarantee is for every message m. We do not want the signing algorithm to create verifiable tags *only* for a subset of messages
- The probability is over the choice of sk output by the generation algorithm Gen()

Message Integrity

- We want to ensure that an adversary cannot tamper the message m into a different message m' such that the original tag τ is also a valid tag for the adversarial message m'
- Let $\mathcal{T}$ be the range of the signing algorithm (i.e., the set of all possible tags)
- Message Integrity can be ensured if the following property holds. For all distinct $m, m' \in \mathcal{M}$, we have

$$\mathbb{P}\left[\mathsf{Sign}_{\mathsf{sk}}(m') = \tau | \mathsf{Sign}_{\mathsf{sk}}(m) = \tau\right] \leqslant \frac{1}{|\mathcal{T}|}$$

  - Note that we cannot insist on the above probability to be 0 when the set of all possible tags is smaller than the set of all messages
    Constructing an adversary who can violate message integrity with probability $\frac{1}{|\mathcal{T}|}$ is an interesting exercise.

- We want to ensure that an adversary cannot forge the tag of a new message m' by observing one message-tag pair (m, τ)
- Unforgeability can be ensured if the following property holds. For all distinct $m, m' \in \mathcal{M}$, we have

$$\mathbb{P}\left[\mathsf{Sign}_{\mathsf{sk}}(m') = \tau' | \mathsf{Sign}_{\mathsf{sk}}(m) = \tau\right] = \frac{1}{|\mathcal{T}|}$$

  - First, note that we insist that the forged message is different from the original message. That is, we have $m' \neq m$. We do not insist on $\tau' \neq \tau$.
  - Next, note that unforgeability is a stronger requirement than message integrity. For example, "unforgeability restricted to $\tau = \tau'$" is identical to "message integrity." Therefore, again, the forging probability above cannot be 0.

Food for thought

- Suppose we want to design a MAC that remains unforgeable even when the adversary has seen t message-tag pairs. How to define t-unforgeability?

Construction using Random Functions

- In the following slides, we will construct a MAC using Random Functions
- Understand its properties and its shortcomings
- Then, we shall replace the random function using a pseudorandom function family in the next lecture

Goal.

- Suppose we have n-bit messages, i.e., the message space is $\mathcal{M} = \{0,1\}^n$
- We will generate $n/100$-bit tags, i.e., the space of tags is $\mathcal{T} = \{0,1\}^{n/100}$

Scheme.

- Secret-key Generation Algorithm.
  - Let f be a random function from the domain $\{0,1\}^n$ to the range $\{0,1\}^{n/100}$
  - Let the secret key sk be the function table of f
  - Both the sender and the verifier will obtain the secret key $\mathsf{sk} = f$
- Tagging Algorithm.
  - The tag $\tau\{0,1\}^{n/100}$ for a message $m \in \{0,1\}^n$ using the secret key $\mathsf{sk} = f$ is computed by: $\tau = f(m)$
  - To endorse the message m, the sender will send the pair (m, τ)
- Verification Algorithm.
  - The verifier will receive a pair $(\widetilde{m}, \widetilde{\tau})$
  - The verifier will check whether $\widetilde{\tau} = f(\widetilde{m})$ or not, where the secret-key $\mathsf{sk} = f$

Analysis of Adversarial Forging Attack.

- Suppose the adversary sees a pair (m, τ)
- The adversary does not know the secret-key $\mathsf{sk} = f$, but it knows that $f(m) = \tau$
- Now, the adversary has to generate a different message $m' \in \{0,1\}^n$ and a tag τ' such that the pair (m', τ') verifies
- The adversarial pair (m', τ') will verify if an only if $f(m') = \tau'$
- Let us look at this probability

$$\mathbb{P}\left[f(m') = \tau' | f(m) = \tau\right]$$

The probability is over the choice of f from the set of all possible functions $\mathcal{M} \to \mathcal{T}$.

- Let us parse this mathematical expression. The adversary already knows the fact that "$f(m) = \tau$." So, we are conditioning on that fact in the probability expression. And, conditioned on this fact, we are interested in finding the probability that $f(m') = \tau'$, where $m' \neq m$

- First observation. Given the fact that $f(m) = \tau$ (i.e., evaluation of a function at one input) the evaluation of $f(m')$ is uniformly random over the range $\mathcal{T} = \{0,1\}^{n/100}$. Because, for a random function, given the evaluation of a function f at one input, the evaluation of the function f at any other input is uniformly random over the range.
- So, conditioned on the knowledge of the adversary that $f(m) = \tau$, the probability that $f(m') = \tau'$, where $m' \neq m$, is "1 divided by the size of the range." In our case, that is

$$\frac{1}{2^{n/100}}$$

- Therefore, we conclude

$$\mathbb{P}\left[f(m') = \tau' | f(m) = \tau\right] = \frac{1}{2^{n/100}}$$

- Note that the entire analysis remains identical if we fix $\tau' = \tau$. Therefore, our construction preserves message integrity.

Conclusion.

- It is highly unlikely that an adversary will be able to forge a tag given one (m, τ) pair

Extension of the Unforgeability Attack.

- In fact, this scheme has an even more interesting property
- Suppose the sender has sent several message-tag pairs. That is, the sender has sent (m_1, τ_1), (m_2, τ_2), ..., (m_t, τ_t). Note that they satisfy the following relation $\tau_1 = f(m_1)$, $\tau_2 = f(m_2)$, ..., $\tau_t = f(m_t)$.
- The adversary has seen all these message-tag pairs. Can the adversary forge a new message-tag pair? Let us see.

Analysis of the Probability of Forging in the Extension.

- Let us write down what the adversary has seen. The adversary knows that

$$f(m_1) = \tau_1, f(m_2) = \tau_2, \ldots, f(m_t) = \tau_t$$

- Conditioned on this information, we are interested in the probability that $f(m') = \tau'$, where m' is different from all the messages $m_1, m_2, \ldots, m_t$
- So, we are interested in the probability

$$\mathbb{P}\left[f(m') = \tau' | f(m_1) = \tau_1, f(m_2) = \tau_2, \ldots, f(m_t) = \tau_t\right],$$

where $m' \notin \{m_1, m_2, \ldots, m_t\}$.

- Main Observation. Even if we know the evaluation of the function f at inputs m_1, m_2, ..., m_t, the evaluation of f at a new input m' is uniformly random over the range. So, we can conclude that the probability of forging is

$$\mathbb{P}\left[f(m') = \tau' | f(m_1) = \tau_1, f(m_2) = \tau_2, \ldots, f(m_t) = \tau_t\right] = \frac{1}{2^{n/100}}$$

Conclusion.

- The MAC using a random function to generate tags is secure even when the adversary sees t message-tag pairs

Positive Features.

- Even if the adversary has unbounded computational power, the probability arguments bounding its probability to forge still holds
- The scheme is secure even when the adversary has seen t message-tag pairs

Primary Shortcoming.

- Let us compute the size of the function table for the function f. Recall that f is from the domain $\{0,1\}^n$ to the range $\{0,1\}^{n/100}$. So, there are a total of $\left(2^{n/100}\right)^{2^n} = 2^{(n/100)2^n}$ different functions. This implies that we need $(n/100)2^n$ (exponential in n) bits to represent this function! Even for $n = 512$, this number is larger than the number of atoms (which is $< 2^{273}$) in the entire universe.

What Next?

To fix the shortcomings mentioned above, we set forth the following goals for ourselves

- We will construct functions that use a smaller key, i.e., length is polynomial in n

However, our security will hold only for computationally bounded adversaries (instead of adversaries with unbounded computational power) In the previous lecture, we have constructed pseudorandom functions, which shall serve this exact purpose!