Lecture 16: 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 various kinds of hash function families

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: $z = \mathsf{Ver}_{\mathsf{sk}}(m, \tau) \in \{0, 1\}$. 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 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 of new messages.
- Secrecy of 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 sign 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 is generated by the honest signer 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

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

  - 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
- This probability guarantee required above seems similar to the guarantee provided by Universal Hash-function Family

Unforgeability

- 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

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

  - Again, 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
- This probability guarantee required above seems similar to the guarantee provided by 2-wise Independent Hash-function Family

Food for thought

- Suppose we want to design a MAC that remains unforgeable even when the adversary has seen $(k-1)$ message-tag pairs. What probability guarantee will be needed?

Construction

- Let $\mathcal{H} = \{h_1, \ldots, h_K\}$ be a hash function family with domain $\mathcal{M}$ and range $\mathcal{T}$
- Construction

> - $\mathsf{Gen}()$ returns $\mathsf{sk} \xleftarrow{\$} \{1, \ldots, K\}$
> - $\mathsf{Sign}_{\mathsf{sk}}(m)$ returns $h_{\mathsf{sk}}(m)$
> - $\mathsf{Ver}_{\mathsf{sk}}(m, \tau)$ returns whether τ is identical to $h_{\mathsf{sk}}(m)$

- This scheme is correct
- If $\mathcal{H}$ is a universal hash-function family, then the MAC scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ ensures message integrity
- If $\mathcal{H}$ is a 2-wise independent hash-function family, then the MAC scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is unforgeable (since 2-wise independence implies universal, this will also ensure message integrity)

Food for thought

- Suppose we want to construct a MAC so that if t-parties among a set of n-parties decide to endorse a message m, then they can add a tag that the verifier can verify. How to construct such a scheme?