CS6222 Cryptography

Topic: Authentication
Date: Nov 12, 2024
Lecturer: Wei-Kai Lin (TA: Arup Sarker)
Scriber: Eric Xie, Adarsha Poudel

---

1 Real-World Authentication

Secure Hash Algorithm (SHA) is a widely used cryptographic hash function that maps inputs $x \in \{0,1\}^*$ to a fixed-length output of 512 bits, i.e., $\text{SHA} : \{0,1\}^* \to \{0,1\}^{512}$. By 2015-2017, major processor manufacturers (Intel, AMD, ARM) began integrating hardware support for SHA, improving its efficiency and performance.

Before SHA became the standard for authentication, particularly during the 2005-2010 period, Advanced Encryption Standard (AES) was often used instead. However, the choice between SHA and AES often depended on hardware constraints. For example, AES generally required more circuit area or volume, making it less efficient for certain applications. Additionally, during this period, authentication was often deprioritized compared to encryption, leading to less emphasis on optimizing authentication methods.

It remains uncertain whether Message Authentication Codes (MACs) were directly built into processors during this time. Instead, many systems likely relied on SHA as a wrapper to simulate MAC-like functionality. This practical choice was likely driven by ease of implementation and existing support for SHA-based operations.

Conceptually, encryption and authentication serve different purposes, yet encryption has been heavily prioritized in real-world applications. Despite this, SHA-based authentication schemes are often more efficient than constructing collision-resistant hash functions for similar purposes, making SHA a practical choice for many authentication scenarios.

2 Digital Signature Scheme

A **digital signature scheme** is a cryptographic construct that ensures the authenticity and integrity of a message while enabling non-repudiation. It consists of three algorithms:

2.1 Components

- **Key Generation (Gen)**: a PPT that produces a pair of keys:

$$(pk, sk) \leftarrow \text{Gen}(1^n),$$

  where pk is the public key, and sk is the secret key.

- **Signing (Sign)**: Takes the secret key sk and a message m as input to produce a signature σ:

$$\sigma \leftarrow \text{Sign}_{sk}(m), \quad m \in \mathcal{M}_n.$$

- **Verification (Ver)**: A deterministic algorithm that uses the public key pk to verify whether a signature σ is valid for a message m:

$$\text{Ver}_{pk}(m, \sigma) \in \{\text{Acc}, \text{Rej}\}.$$

2.2 Security Definition

A digital signature scheme is secure if no adversary, given polynomial-time access to the public key pk and a signing oracle, can forge a valid signature for any message m' that has not been queried to the signing oracle.

For all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$, there exists a negligible function $\epsilon(n)$ such that:

$$\Pr\begin{bmatrix}(pk, sk) \leftarrow \text{Gen}(1^n); \\ (m, \sigma) \leftarrow \mathcal{A}^{\text{Sign}_{sk}(\cdot)}(1^n, pk) : \\ \mathcal{A} \text{ did not query } m, \text{ and } \text{Ver}_{pk}(m, \sigma) = \text{Acc}\end{bmatrix} \leq \epsilon(n),$$

where n is the security parameter.

2.3 Comparison to MAC (Message Authentication Code)

- **Public Key vs. Secret Key**: Digital signatures utilize a *public key* pk for verification and a *secret key* sk for signing. In contrast, MAC schemes rely on a single shared *secret key* for both signing and verification.
- **Public Verification**: Digital signatures enable *public verification*, allowing anyone with the public key to validate a signature. MAC schemes require the shared secret key for verification.
- **Security Assumptions**: Digital signature security assumes the adversary knows pk but cannot forge σ for unqueried messages, whereas MAC schemes rely on the secret key's confidentiality.

3 Lamport One-Time Digital Signature Scheme

The **Lamport Signature Scheme** is a simple one-time signature scheme based on a one-way function (OWF).

Let f be a one-way function (OWF): $f : \{0,1\}^n \to \{0,1\}^n$.

3.1 Construction

- **Key Generation (Gen)**: Generate $2n$ strings chosen uniformly random of length n:

$$\text{sk} = \begin{bmatrix} x_{0,1} & x_{0,2} & \dots & x_{0,n} \\ x_{1,1} & x_{1,2} & \dots & x_{1,n} \end{bmatrix},$$

  where $x_{b,i} \in \{0,1\}^n$ for $b \in \{0,1\}$ and $i \in [1,n]$. Compute the public key using a one-way function $f : \{0,1\}^n \to \{0,1\}^n$:

$$\text{pk} = \begin{bmatrix} f(x_{0,1}) & f(x_{0,2}) & \dots & f(x_{0,n}) \\ f(x_{1,1}) & f(x_{1,2}) & \dots & f(x_{1,n}) \end{bmatrix}.$$

- **Signing (Sign)**: For a message $m \in \{0,1\}^n$, represented as $m = m_1 m_2 \dots m_n$:

$$\sigma = \{x_{m_1,1}, x_{m_2,2}, \dots, x_{m_n,n}\}.$$

  Each bit m_i determines whether to choose $x_{0,i}$ or $x_{1,i}$ from the secret key.

- **Verification (Ver)**: For a given (m, σ), check:

$$f(\sigma_i) = f(x_{m_i,i}) \quad \forall i \in [1, n].$$

If all checks pass, accept; otherwise, reject.

3.2 Claim: One-Time Security of Lamport Signature Scheme

The Lamport Signature Scheme is **one-time secure**. In the security game, the adversary is only allowed to see one valid message-signature pair by querying Sign_{sk} once, rather than querying the oracle a polynomial number of times.

Although this scheme is designed for one-time use, it is efficient and simple. While we might hope to extend its use to multiple messages for practical purposes, this "weak" one-time security can be strengthened using additional cryptographic techniques.

The Lamport Digital Signature Scheme (LDS) is proven to be **one-time secure** based on the hardness of inverting a one-way function f. The proof uses a reduction by contradiction: if there exists an adversary $\mathcal{A}$ that can forge a new message-signature pair (m', σ') such that $m' \neq m$ and the signature is valid, we construct another adversary $\mathcal{B}$ that inverts f, thereby violating its one-wayness.

The core intuition is that, for $m' \neq m$, there exists at least one bit position i where $m_i \neq m'_i$. At this position, the adversary $\mathcal{A}$ must produce a signature component σ'_i such that:

$$f(\sigma'_i) = \begin{cases} y_{1,i} & \text{if } m'_i = 1, \\ y_{0,i} & \text{if } m'_i = 0. \end{cases}$$

This effectively requires $\mathcal{A}$ to invert f, a task that is computationally infeasible.

Since the adversary $\mathcal{A}$ does not know which index i contains the difference $(m_i \neq m'_i)$, it cannot efficiently forge a valid signature. This reduction demonstrates that the security of LDS is rooted in the one-wayness of f, ensuring that the scheme is secure for one-time use.

4 Extending Lamport Signatures for Larger Message Spaces

4.1 Key Observations

For the Lamport Digital Signature Scheme (LDS):

$$|\text{sk}| = 2n^2, \quad |\text{pk}| = 2n^2, \quad |m| = n, \quad \text{where } n \in \mathbb{N}.$$

This construction limits the message length to n bits, making it challenging to extend the message length for larger message spaces M.

4.2 Solution: Composition with Hash Functions

To support larger message spaces or longer messages, we can compose the Lamport scheme with:

- **Collision-Resistant Hash Functions (CRHF)**, or
- **Universal One-Way Hash Functions (UOWHF)**.

The composition works as follows:

1. Instead of directly signing the message m, compute a hash $h(m)$ using a CRHF or UOWHF to reduce the effective message size.
2. Apply the Lamport scheme to sign $h(m)$.

Lemma: Using LDS with a CRHF or UOWHF produces a secure one-time signature for message spaces $M = \{0,1\}^*$, where messages can have arbitrary length.

4.3 Extending with Multiple Key Pairs

To extend the scheme for multiple messages:

1. Generate multiple key pairs:
$$(\text{pk}_0, \text{sk}_0) \leftarrow \text{Gen}(1^n), \quad (\text{pk}_1, \text{sk}_1) \leftarrow \text{Gen}(1^n).$$
2. Use a higher-level key pair:
$$(\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^n),$$
where $\text{pk} = (\text{pk}_0 \parallel \text{pk}_1)$.

4.4 Signing and Verifying

- **Signing:**
  1. Sign the concatenated public keys with the higher-level secret key:
  $$\sigma \leftarrow \text{Sign}_{\text{sk}}(\text{pk}_0 \parallel \text{pk}_1).$$
  2. For the message m_1, sign it using one of the lower-level secret keys:
  $$\sigma_1 \leftarrow \text{Sign}_{\text{sk}_0}(m_1).$$
- **Verification:**
  1. Verify the higher-level signature σ against the concatenated public keys.
  2. Verify σ_1 against m_1 using the corresponding lower-level public key.

4.5 Binary Tree Construction

To extend the scheme further for signing multiple messages, we can construct a binary tree where:

- Each node represents a public-private key pair.
- The root key pair is used to sign the keys of its child nodes.

When signing a message:

1. The message is associated with a leaf node.
2. To verify the signature, the public keys on the path from the root to the leaf must be transmitted alongside the message and its signature.

This structure allows a single one-time signature to support multiple messages while preserving security. The binary tree reduces key overhead and ensures scalability.