CS6222 Cryptography

Topic: Authentication
Date: Oct 29, 2024
Lecturer: Wei-Kai Lin (TA: Arup Sarker)
Scriber: Eric Xie, Alex Fetea

---

1 Authentication

To ensure message integrity and defend against attacks where a message may have been corrupted, we introduce a mechanism for message authentication. Message authentications serves two main purposes: confirming the sender of the message, and verifying the integrity/contents of the message.

1.1 Tagging and Verification

A simple apporach is to add a unique tag to each message. This tag functions as a message signature. We define this as:

$$\sigma \leftarrow \text{Tag}(m)$$

where σ is the tag generated for a message m.

When a receiver receives the tagged message (m, σ), even if the message or signature is modified, a verification function Ver can be used to validate the message-signature pair, preserving message integrity. This Ver function will work as follows:

$$\text{Ver}(m, \sigma) = \begin{cases} \text{Acc} & \text{if } (m, \sigma) \text{ is valid} \\ \text{Rej} & \text{otherwise} \end{cases}$$

1.2 Attacker Knowledge and Key-Based Security

We assume that an adversary has knowledge of all components of the system. Our initial version is insufficient to defend against attacks, as a known Tag algorithm allows an adversary to verify their own message.

To combat this vulnerability, we introduce a key generation function, making our functions as follows:

$$\sigma \leftarrow \text{Tag}_k(m) \quad \text{and} \quad \{\text{acc}, \text{rej}\} \leftarrow \text{Ver}_k(m', \sigma')$$

where $k \leftarrow \text{Gen}(1^n)$. This key k is a secret key known only by the sender and the intended reciever. This approach makes it computationaly infeasible for an attacker to create (m, σ) pairs without knowing k.

2 Message Authentication Code (MAC)

A Message Authentication Code (MAC) is used to validate the authenticity and integrity of a message. A MAC binds a key to the message, which allows other particpants who hold the key to verify the message. The scheme consists of three functions: `Gen`, `Tag`, and `Ver`.

2.1 Definition

- **Key Generation:** $k \leftarrow \text{Gen}(1^n)$ - A key k is generated based on the security parameter n. This key is used by both the sender and reciever, and must be secret to ensure security.
- **Tagging:** $\sigma \leftarrow \text{Tag}_k(m)$ - A tag σ is produced for a message m using the key k. This servers as the fingerprints of the message.
- **Verification:** $\{\text{Acc, rej}\} \leftarrow \text{Ver}_k(m, \sigma)$ - The verification function checks the validity of the message-tag pair (m, σ) and returns either acceptance (Acc) or rejection (rej).

2.2 Correctness

The MAC scheme should satisfy correctness, meaning that for any valid (m, σ) generated with a key k will be able to be verified. Formally, any message $m \in M$ and any security parameter $n \in \mathbb{N}$,

$$\Pr_{k \leftarrow \text{Gen}(1^n)} [\text{Ver}_k(m, \text{Tag}_k(m)) = \text{Acc}] = 1.$$

2.3 Security

We consider three versions of security definitions to outline the scheme's robustness against forgery attempts by an adversary A. The definitions are shown below, each showing an increase resistance to forgery.

Basic Security Definition - Version 1: An adversary is given the security parameter n and outputs a pair (m', σ). The probability that the verification function accepts this forged pair is bounded as follows:

$$\Pr\left[(m', \sigma) \leftarrow A(1^n) : \text{Ver}_k(m', \sigma) = \text{Acc}\right] \leq \epsilon(n).$$

Security with Known Message-Tag Pair -Version 1.1: The adversary receives an additional message-tag pair (m, σ) and attempts to forge a new valid pair (m', σ):

$$\Pr\left[(m', \sigma) \leftarrow A(m, \sigma) : \text{Ver}_k(m', \sigma) = \text{Acc}\right] \leq \epsilon(n).$$

While this version provides the adversary with one extra pair, it does not reveal any information about k, keeping the system secure.

Version 1.X: For any message $m \in \mathbb{M}$, we generate an honest tag $\sigma \leftarrow \text{Tag}_k(m)$ for $m \neq m'$ that can be verified. The adversary now must output a different valid pair (m', σ') under the assumption:

$$\Pr\left[(m', \sigma') \leftarrow A(m, \sigma) : \text{Ver}_k(m', \sigma') = \text{Acc}\right] \leq \epsilon(n).$$

In this scenario, the scheme allows for the signing of multiple messages by either using distinct halves of the key for different messages or using separate keys entirely, ensuring that each message is independently secure.

2.4 Full Security Definition

For a Message Authentication Code (MAC) scheme to be secure, it must withstand attacks from any Non-Uniform Probabilistic Polynomial-Time (NUPPT) adversary. Formally, we require that:

$$\forall \text{ NUPPT } A, \exists \text{ negligible function } \epsilon(\cdot) \text{ such that for all } n \in \mathbb{N},$$
$$\Pr\left[(m', \sigma') \leftarrow A^{\text{Tag}_k(\cdot)}(1^n) : \text{Ver}_k(m', \sigma') = \text{Acc}\right] \leq \epsilon(n).$$

The only constraint imposed on the adversary A is that it must output a message m' that has never been queried to the tagging oracle $\text{Tag}_k(\cdot)$. This ensures that m' is a fresh message, and the adversary cannot rely on previously seen message-tag pairs to forge a valid signature.

2.5 Security Game

The security of the Message Authentication Code (MAC) scheme can be evaluated by using a security game between an adversary A and a challenger Chal. This game models an interaction where the adversary attempts to forge a valid tag for a message they have not queried to the tagging oracle.

1. **Setup:**
   - Both A and Chal receive a security parameter 1^n, where n is the level of security.
   - The challenger generates a secret key k by using the key generation function:
   $$k \leftarrow \text{Gen}(1^n)$$
   - This key k remains hidden from A, functioning as the secret of the legitmate users of the MAC scheme.
2. **Tagging Oracle:**
   - The challenger provides the adversary A with access to a tagging oracle, $\text{Tag}_k(x)$, allowing A to submit messages of its choice
   - The adversary can query this oracle multiple times with chosen messages m and receive the corresponding tag $\sigma = \text{Tag}_k(m)$.
   - This phase is supposed to mimic scenarios where an adversary intercepts network communcation between parties, and can attempt to analyze the tags to try and detect patterns.
3. **Adversary's Goal:**
   - The adversary A aims to produce a new, unqueried message-tag pair (m', σ') such that:
   $$\text{Ver}_k(m', \sigma') = \text{Acc}$$
   - Here, m' is a message that has never been queried to the oracle Tag_k. This is to make sure the adversary is forging a new valid tag, rather than repeating one seen before.
4. **Winning Condition:**

- The adversary A wins the game if they can produce a valid tag for any message m' that was not previously queried to the tagging oracle, i.e., if:

$$\text{Ver}_k(m', \sigma') = \text{Acc} \quad \text{and} \quad m' \neq m \quad \text{for all queries } m \text{ to } \text{Tag}_k.$$

- If the adversary meets this win condition, it has successfully forged a valid tag without having the key and is not relying on previously seen pairs.

This security game simulates a scenario in which an adversary tries to deceive honest users in a network by having them sign or tag specific messages. This demonstrates how a secure MAC scheme ensures that the adversary's probability of winning this game is negligible, indicating resistance to forgery attacks.

3 Construction: Information-Theoretic MAC

To construct a Message Authentication Code (MAC) with information-theoretic security, we define the following components:

3.1 Key Generation

$$k \leftarrow \{0,1\}^n$$

where k is a randomly generated bit string of length n.

3.2 Tagging

The tagging function $\text{Tag}_k(m)$ produces a tag σ by taking the bitwise XOR of the message m and the key k:

$$\sigma \leftarrow m \oplus k$$

where:

- $m \in \{0,1\}^n = \mathbb{M}$
- k is the secret key of length n
- $\oplus$ is the bitwise XOR operator

The tag generated by XORing each bit of m with the corresponding bit in k, will act as an authentication code. This is useful because its simple, and has an inverse operation that we can use for verification.

3.3 Verification

The verification function $\text{Ver}_k(m, \sigma)$ accepts if the following condition holds:

$$\text{Acc} \quad \text{if } m = \sigma \oplus k.$$

This requires that the original message m can be retrieved by XORing the tag σ with the key k.

3.4 Security Discussion

In this construction, the adversary must guess the correct k, but since no information about k is leaked in Definition 1, the adversary's only viable approach is a random guess, no matter the computation time.

- Since k is chosen uniformly at random, any tag could correspond to many different possible keys k. This heavily spends on value m, and make its impossible for an adversary to guess k by observing a message and tag pair.
- The use of a random key k ensures that each tag σ is effectively random from the adversary's perspective.

4 Defining a Digital Signature Scheme

A digital signature scheme can be defined similarly, with a pair of public and private keys (pk, sk) generated as follows:

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

where:

- pk is the public key, which is shared openly and can be used to verify signatures
- sk is the private key, which is only known by the sender and used to generate signatures

In this setting, the sender A transmits the public key pk to the receiver B, while an adversary can observe the public key, but they cannot derive the private key, as generating pk from sk is designed to be computationally infeasible.

4.1 Signing and Verification

The signing function produces a signature σ:

$$\sigma \leftarrow \text{Sign}_{\text{sk}}(m)$$

The verification function accepts or rejects based on whether the signature is valid for a given message m under the public key pk:

$$\text{Ver}_{\text{pk}}(m, \sigma) = \{\text{acc}, \text{rej}\}$$

4.2 Security of the Digital Signature

The security of this scheme is defined similarly to the MAC:

$$\Pr_{(\text{pk},\text{sk})\leftarrow\text{Gen}(1^n)}\left[(m', \sigma') \leftarrow A^{\text{Sign}_{\text{sk}}(\cdot)}(1^n, \text{pk}) : \text{Ver}_{\text{pk}}(m', \sigma') = \text{acc}\right] \leq \epsilon(n)$$

where A is the adversary with access to the $\text{Sign}_{\text{sk}}(\cdot)$, allowing them to recieve valid signatures on messages.

The adversary has access to some message-signature pairs and observes the public key, but is assumed unable to modify m', as doing so would invalidate the scheme's objective.

4.3 Security Implications

The digital signature's security model ensures that even if an adversary sees multiple message signature pairs, they cannot forge a valid signature because:

- Any modification to m would invalidate teh signature.
- The scheme validates sender's identity, as only someone with the private key can create valid signatures.
- Since only the person with the private key can make a valid signature, they cant deny that a message came from them if the signature is valid.

5 Constructing a MAC and Digital Signature Using a Pseudorandom Function (PRF)

We can construct both a MAC and a digital signature scheme using a pseudorandom function (PRF). Assume we have a family of PRFs $F = \{f_k : \{0,1\}^n \to \{0,1\}^n \mid |k| = n, n \in \mathbb{N}\}$. The ideas is the use the randomness of PRF's to create a secure MAC scheme by linking each message with a unique, seemingly random tag.

5.1 MAC Construction Using PRFs

- **Key Generation:**
$$k \leftarrow \{0,1\}^n$$
- **Tagging:**
$$\text{Tag}_k(m) : m \in \mathbb{M} = \{0,1\}^n$$
Output $\sigma \leftarrow f_k(m)$.
- **Verification:**
$$\text{Ver}_k(m, \sigma) : \quad \text{Acc if and only if } \sigma = f_k(m)$$

The construction $(\text{Gen}, \text{Tag}, \text{Ver})$ defines a MAC.

5.2 Correctness

The MAC scheme satisfies correctness because:

$$\text{Ver}_k(m, \sigma = \text{Tag}_k(m)) \Rightarrow f_k(m) = \sigma$$

indicating that the generated tag σ matches $f_k(m)$, confirming the message's authenticity.

5.3 Security Proof

The security of this MAC scheme can be verified through a reduction from the original MAC security definition, ensuring that any successful forgery attempt would contradict the security of the underlying PRF. Heres how it works:

- Assume adversary A successful creates a valid message tag pair m', σ' with non-negligble probability. Here m' was no queried to the tagging oracle, but $\sigma' = f_k(m')$

- If A can produce a valid pair, this implies that A can compute $f_k(m')$ without querying it. This would contradict the security of PRF which should be indistinguishable from random.
- Since the security of PRF ensures that probability of generating a valid pair without the key is negligible, the same must also hold true for a MAC scheme based on f_k

References