1 Plan for today's lecture

Last class, we discussed the motivation for studying message authentication codes. In today's lecture, we will define this primitive formally, study the relation between the different definitions, and finally see a construction.

2 Message authentication codes (MACs)

A message authentication code (MAC) for a message space $\mathcal{M}$ consists of three polynomial time algorithms (KeyGen, Sign, Verify) with the following syntax and correctness.

Syntax

- $\mathsf{KeyGen}(1^n)$: takes as input the security parameter and outputs a key k.
- $\mathsf{Sign}(m, k)$: takes as input a message $m \in \mathcal{M}$, a key k, and outputs a signature σ.
- $\mathsf{Verify}(m, \sigma, k)$: takes as input a message m, signature σ, key k, and outputs either 0 or 1.

Correctness : A MAC $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is said to be correct if for all keys k, messages $m \in \mathcal{M}$,

$$\mathsf{Verify}(m, \mathsf{Sign}(m, k), k) = 1.$$

If sign or verify are probabilistic, then the above should hold with probability 1.

3 Security games and corresponding security definitions

3.1 Strong Unforgeability under Chosen Message Attacks

The first one (defined in Figure 1) is a natural security definition: even after seeing many signatures, the adversary should not be able to produce a new signature. In this course, I will refer to this security game/definition as 'strong unforgeability under chosen message attack'. The 'chosen message' comes from the fact that the adversary gets to choose the messages for which it sees a signature. I am calling this 'strong unforgeability' because the adversary is allowed to send a new signature on one of the queried messages. [1] We can also consider weaker notions where the adversary is not allowed to submit a signature on one of the queried messages. [2]

Strong-UF-CMA

1. **Setup:** Challenger chooses a signing key k.
2. **Signature queries:** Adversary sends polynomially many signing queries (adaptively). The i^{th} query is a message m_i. The challenger sends $\sigma_i = \mathsf{Sign}(m_i, k)$ to the adversary.
   Note that the key k was chosen during setup, and the same key is used for all queries.
3. **Forgery:** The adversary sends a message m^* together with a signature σ^*, and wins if $(m^*, \sigma^*) \neq (m_i, \sigma_i)$ for all i, and $\mathsf{Verify}(m^*, \sigma^*, k) = 1$.

Figure 1: Security Game for MACs: Strong Unforgeability under Chosen Message Attack

[1] The textbook calls this just *unforgeability*.
[2] In the textbook, this is called *weak unforgeability*.

Definition 18.01. A MAC (KeyGen, Sign, Verify) is said to satisfy Strong-UF-CMA if, for any p.p.t. adversary $\mathcal{A}$, there exists a negligible function $\mu(\cdot)$ such that for all n,

$$\Pr\left[\mathcal{A} \text{ wins the strong unforgeability game}\right] \leq \mu(n).$$

3.2 Strong Unforgeability under Chosen Message Attacks, with Verification Queries

Next, we consider a strengthening of this definition, where the adversary also can also make 'verification queries'. The game is defined in Figure 2. This captures the real-world scenarios where the adversary may make some modifications to the signatures, and then 'check' if the modifications lead to an unforgeability attack. [3]

Strong-UF-CMA-Ver

1. **Setup:** Challenger chooses a signing key k.
2. **Queries:** Adversary can send either signing or verification queries adaptively.
   - **Signing queries:** The adversary sends message m_i and receives $\sigma_i = \mathsf{Sign}(m_i, k)$ from the challenger.
   - **Verification queries:** The adversary sends a message m'_i, a signature σ'_i such that $(m'_i, \sigma'_i) \neq (m_j, \sigma_j)$ for all j, and receives $\mathsf{Verify}(m'_i, \sigma'_i, k)$ from the challenger.

   Note that the key k was chosen during setup, and the same key is used for all queries. Also, the signing and verification queries can be interleaved.
3. **Winning condition:** The adversary wins if there exists a verification query (m'_i, σ'_i) such that $\mathsf{Verify}(m'_i, \sigma'_i, k) = 1$.

Figure 2: Security Game for MACs: Strong Unforgeability under Chosen Message Attacks, with verification queries

A few comments about this game:

- for the verification queries, the adversary must send (m'_i, σ'_i) such that it is not one of the (message, signature) pairs received from the challenger (via signing queries). Verification queries of the form (m'_i, σ'_i) where m'_i is one of the prior signing queries, and σ'_i is the received signature are not useful for the adversary (since the challenger is honest), and as a result, we assume for simplicity that the adversary does not make these queries.
- In Definition 18.01, the adversary submits a forgery at the end. Here, the adversary wins if one of the verification queries corresponds to a forgery (that is, the verification passes). Again, this is a simplification, and there is no loss of generality.

Definition 18.02. A MAC (KeyGen, Sign, Verify) is said to satisfy Strong-UF-CMA-Ver if, for any p.p.t. adversary $\mathcal{A}$, there exists a negligible function $\mu(\cdot)$ such that for all n,

$$\Pr\left[\mathcal{A} \text{ wins the strong unforgeability game with verification queries}\right] \leq \mu(n).$$

[3] Note: we are not adding verification queries because the challenger might be cheating. In all security games, the challenger represents the honest prty/parties in the system. In the case of signatures, the challenger represents the signing/signature-receiving parties, which are the honest parties in the system. The adversary is a 'man-in-the-middle' who sees a few signatures sent by the signer, and tries to fool the receiver by forging a new signature.

3.3 A few broken MAC schemes

Here are a few attempts at building a MAC scheme. The first one was suggested in one of the previous lectures, and is sometimes believed to offer message integrity. It is defined using an efficient, publicly known, deterministic function called $\mathsf{CRC32}$ ('cyclic redundancy check'). The scheme has no signing/verification key.

- $\mathsf{Sign}(m)$: Output $\mathsf{CRC32}(m)$ as the signature.
- $\mathsf{Verify}(m, \sigma)$: Output 1 iff $\sigma = \mathsf{CRC32}(m)$.

Note that this does not offer any security against unforgeability attacks. The attacker doesn't even need any queries; it can forge a signature on a desired message by simply sending $(m, \mathsf{CRC32}(m))$.

The next one is a bit more interesting, and a variant of it is also part of some cryptographic standards. It uses a PRF $F : \mathcal{X} \times \mathcal{K} \to \mathcal{Y}$, together with $\mathsf{CRC32}$ as follows.

- KeyGen : Pick a random PRF key $k \leftarrow \mathcal{K}$.
- $\mathsf{Sign}(m)$: Pick $x \leftarrow \mathcal{X}$, output $(x, F(x,k) \oplus \mathsf{CRC32}(m))$ as the signature.
- $\mathsf{Verify}(m, \sigma = (\sigma_1, \sigma_2), k)$: Output 1 iff $\sigma_2 = F(x,k) \oplus \mathsf{CRC32}(m)$.

It is easy to produce a forgery after seeing just one signature! Given a signature $\sigma = (\sigma_1, \sigma_2)$ on message m, we can compute a signature on m' as follows: set $\sigma'_1 = \sigma_1$, set $\sigma'_2 = \sigma_2 \oplus \mathsf{CRC32}(m) \oplus \mathsf{CRC32}(m')$.
Check that this is a valid forgery, and therefore the above scheme does not satisfy strong unforgeability.

4 MAC: Verification queries are useless

In this section, we will show that any MAC satisfying Definition 18.01 without verification queries also satisfies Definition 18.02. Such a result is helpful because it is usually easier to construct schemes satisfying Definition 18.01, but Definition 18.02 is more useful when we want to use MACs as a building block.

Intuition: Suppose there exists an adversary that can succeed with verification queries. We want to build a reduction algorithm that can win the unforgeability game without verification queries. Since the adversary gains some advantage via the verification queries, some verification queries are valid signatures. The reduction algorithm cannot make verification queries to the challenger, hence it does not know which ones are the valid ones. But it can pick a random index, and this guess is correct with probability at least $1/q$, where q is the number of queries.

4.1 The reduction algorithm

Suppose there exists an adversary that wins the unforgeability game with verification queries. We will make a few assumptions (which are made for ease of presentation). First, we assume that we know the number of verification queries made by the adversary. As discussed in previous lectures, if we don't know this number, then there are ways to 'guess' an upper bound on the number of verification queries. Second, we assume that the adversary always makes q queries (even if it has found a forgery after the i^{th} verification query).

The reduction algorithm picks an index i^* uniformly at random from $\{1, \ldots, q\}$. It does the following until it sees the $(i^*)^{\text{th}}$ verification query:

- if the adversary $\mathcal{A}$ requests for a signature on m, the reduction algorithm forwards it to the challenger. It receives σ, which it forwards to $\mathcal{A}$.
- for the j^{th} verification query (m'_j, σ'_j), if $j < i^*$, the reduction algorithm sends 0. Else, if $j = i^*$, the reduction algorithm sends (m'_j, σ'_j) to the challenger.

Claim 18.01. Suppose there exists a p.p.t. adversary $\mathcal{A}$ that wins the strong unforgeability game with probability p by making q verification queries (see Figure 2 for the security game). Then there exists a p.p.t. reduction algorithm $\mathcal{B}$ that wins the strong unforgeability game without verification queries (see Figure 1 for security game) with probability at least p/q.

A proof of this theorem can be found in Section 6.2 of the textbook.

Note: instead of strong unforgeability, suppose we consider just unforgeability (where the adversary must output a signature on a new message). Check that the above proof no longer works! In particular, unforgeability without verification queries DOES NOT IMPLY unforgeability with verification queries. See Question 18.01(*) below.

5 MACs with bounded message space, from PRFs

In this section, we present a MAC construction using PRFs. Let $F : \mathcal{X} \times \mathcal{K} \to \mathcal{Y}$ be a secure PRF. We will construct a MAC scheme with message space $\mathcal{M} = \mathcal{X}$, key space $\mathcal{K}$, and signature space $\mathcal{Y}$.

5.1 Construction

- KeyGen : choose a random key $k \leftarrow \mathcal{K}$.
- $\mathsf{Sign}(m, k)$: Given message $m \in \mathcal{M} = \mathcal{X}$, output $y = F(m, k)$ as the signature.
- $\mathsf{Verify}(m, \sigma, k)$: Output 1 iff $F(m, k) = \sigma$.

Correctness follows immediately from the construction.

5.2 Proof of security

In this section, we will show that the above scheme satisfies Definition 18.01.

Intuition: As a first step, can the adversary win the game without making any queries? In order to do so, the adversary must output $(m^*, F(m^*, k))$. But without making any queries, the adversary has no idea about k, and therefore, its best bet is to pick some m^*, pick a random $y \leftarrow \mathcal{Y}$, and output (m^*, y). The guess will be correct with probability $1/|\mathcal{Y}|$, and therefore, without any queries, the adversary can win with probability at most $1/|\mathcal{Y}|$.

Let us now consider general adversaries. The adversary is receiving some information about the key. But using PRF security, we can replace $F(\cdot, k)$ with a uniformly random function f. As a result, the adversary is playing a modified security game where it interacts with a random function, and must finally output m^* together with $f(m^*)$.

6 Lecture summary, plan for next lecture, additional resources

Summary: We first saw two different definitions of MAC security. In the first definition, the adversary receives a bunch of signatures, and must output a new (message, signature) pair. In the second definition, the adversary is also allowed to make verification queries. These two definitions are equivalent.

Next, we saw two broken MAC schemes.

Finally, we saw a construction based on PRFs.

Next lecture: We will prove security of the PRF-based construction in next class. After that, we will discuss how to sign long messages.

Relevant sections from textbook [Boneh-Shoup]: Section 6.2 contains a proof of the equivalence of the two definitions. The PRF construction is available in Section 6.3.

7 Questions

Question 18.01(*). In Lecture 17, we saw a weaker security game where the adversary must output a signature on a **new** message. We call this game *unforgeability under chosen message attacks*, and it is defined below. One can also consider a variant of this, where the adversary is also allowed to make verification queries. We saw that verification queries are useless for strong unforgeability (and as a result, Definitions 18.01 and 18.02 are equivalent).

Show that there exists a MAC that satisfies 'unforgeability under chosen message attacks', but does not satisfy 'unforgeability under chosen message attacks with verification queries'.

Hint:

- First, define your MAC such that given a signature σ on message m, it is possible to generate many valid signatures on m. Note that this will not break the unforgeability of your original MAC (although your scheme will not be strongly unforgeable).
- Next, note that the verification queries are allowed to query on any (message, signature) pair. Design your MAC such that each verification query reveals one bit of the signing/verification key.

Question 18.02. Let $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a MAC with key space $\mathcal{K}$, message space $\mathcal{M}$ and signature space $\mathcal{T}$. Additionally, the signing algorithm is randomized. It takes n bits of randomness to compute signatures. Construct a new MAC scheme $(\mathsf{KeyGen}_{\mathrm{det}}, \mathsf{Sign}_{\mathrm{det}}, \mathsf{Verify}_{\mathrm{det}})$ such that $\mathsf{Sign}_{\mathrm{det}}$ is deterministic. You can use additional cryptographic primitives as well.

Hint:

- Use a PRF with input space $\mathcal{M}$, key space $\mathcal{K}$ and output space $\{0,1\}^n$.
- The new signature scheme has two keys: a key k for Sign and Verify, and a PRF key k'. To sign a message m, compute $r = F(m, k')$, then use r as randomness for $\mathsf{Sign}(m, k)$.