Lecture 8: Authenticated Encryption and CCA *Security*

Notes by Yael Kalai

MIT - 6.5620
Lecture 8 (September 29, 2025)

> ***Warning:*** This document is a rough draft, so it may contain bugs. Please feel free to email me with corrections.

Recap

- Last class we proved the security of the GGM construction, which constructs a PRF from any PRG.
- We defined the notion of a message authentication code (MAC), and constructed a MAC from any PRF.

  The construction is extremely simple: Given any PRF : $\mathcal{K}_\lambda \times \mathcal{M}_\lambda \to \{0,1\}^\lambda$, let

$$\mathsf{MAC}(k,m) := \mathsf{PRF}(k,m).$$

Today

- Prove the security of the MAC construction.
- Construct an Authenticated Encryption scheme.
- Define CCA-secure encryption.
- Prove that an authenticated encryption is CCA-secure.

Security of the MAC *Construction*

Theorem 1. *The above* MAC *is existentially unforgeable against adaptive chosen message attacks.*

Proof. Fix any poly-size $\mathcal{A}$. By the security of the PRF it holds that there exists a negligible μ such that for every $\lambda \in \mathbb{N}$,

$$\Pr[\mathcal{A}^{\mathsf{MAC}(k,\cdot)}(1^\lambda) = (m^*,\tau^*) : \ \tau^* = \mathsf{MAC}(k,m^*) \ \ \wedge \ \ m^* \notin Q] \le$$
$$\Pr[\mathcal{A}^{R_\lambda(\cdot)}(1^\lambda) = (m^*,\tau^*) : \ \tau^* = R_\lambda(m^*) \ \ \wedge \ \ m^* \notin Q] + \mu(\lambda),$$

where $R_\lambda : \mathcal{M}_\lambda \to \{0,1\}^\lambda$ is a truly random function. It is easy to see that

$$\Pr[\mathcal{A}^{R_\lambda(\cdot)}(1^\lambda) = (m^*, \tau^*) : \tau^* = R_\lambda(m^*) \quad \wedge \quad m^* \notin Q] = 2^{-\lambda}.$$

Thus, we conclude that for every $\lambda \in \mathbb{N}$,

$$\Pr[\mathcal{A}^{\mathsf{MAC}(k,\cdot)}(1^\lambda) = (m^*, \tau^*) : \tau^* = \mathsf{MAC}(k, m^*) \quad \wedge \quad m^* \notin Q] \le \mu(\lambda) + 2^{-\lambda}$$

which is negligible.

□

MAC*ing Long Messages*

Suppose we are given a PRF that takes as input messages in $\{0,1\}^n$ and we want to MAC messages in $\{0,1\}^{2n}$? In particular, suppose we want to MAC the message $m_1||m_2$ where $m_1, m_2 \in \{0,1\}^n$. It is tempting to let

$$\mathsf{MAC}(k, m_1||m_2) = \mathsf{MAC}(k, m_1)||\mathsf{MAC}(k, m_2).$$

However, this will not be secure, since the adversary can "mix-and-match". Specifically, given

$$\mathsf{MAC}(k, m_1||m_2) = \mathsf{MAC}(k, m_1)||\mathsf{MAC}(k, m_2)$$

the adversary can compute

$$\mathsf{MAC}(k, m_2||m_1) = \mathsf{MAC}(k, m_2)||\mathsf{MAC}(k, m_1).$$

Moreover, given

$$\mathsf{MAC}(k, m_1||m_2) = \mathsf{MAC}(k, m_1)||\mathsf{MAC}(k, m_2)$$

and

$$\mathsf{MAC}(k, m_1'||m_2') = \mathsf{MAC}(k, m_1')||\mathsf{MAC}(k, m_2'),$$

the adversary can generate

$$\mathsf{MAC}(k, m_1||m_2') = \mathsf{MAC}(k, m_1)||\mathsf{MAC}(k, m_2').$$

Remark. Note that this is in contrast to CPA secure encryption, where there concatenation preserves security!

So what if we want to MAC a long message? Well, there are several ways to solve this problem. One way is to construct a PRF for messages of arbitrary length by making the GGM tree deeper. Recall that the depth of the tree determines the length of the message. We will talk about other ways in the next lecture.

Authenticated Encryption

Often in application we want both secrecy and authenticity! So, what should we do? Thankfully, we already constructed a CPA-secure encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$, and we also constructed a MAC. But how do we put them together?

Take 1:

1. Alice and Bob first share a random secret key $k \leftarrow \{0,1\}^\lambda$.
2. Every time Alice sends a message $m \in \mathcal{M}_\lambda$ to Bob, she should encrypt her message
$$\mathsf{ct} \leftarrow \mathsf{Enc}(k, m).$$
3. Then, since she wishes to send this ciphertext in an authenticated way, she computes $\tau = \mathsf{MAC}(k, \mathsf{ct})$.
4. She sends (ct, τ).

Upon receiving (ct, τ), Bob first checks that indeed $\tau = \mathsf{MAC}(k, \mathsf{ct})$, and if this is not the case it discards this message! If the check passes, then he knows that this ciphertext was indeed sent by Alice, and then he decrypts it to retrieve the message $m = \mathsf{Dec}(k, \mathsf{ct})$. Is this secure? Not necessarily!

Lesson: Never use the same secret key for different applications! This can leak the secret key entirely! The issue is that both a MAC and a ciphertext may leak information about the secret key, and these leakages may not play nicely with each other!

The correct way to communicate securely, is to share *two* secret keys, one for the encryption and one for the MAC.

Our authenticated encryption scheme. Given a CPA-secure encryption $(\mathsf{Enc}, \mathsf{Dec})$ and given a MAC scheme MAC, we define an authenticated encryption scheme, denoted by $(\mathsf{Enc}', \mathsf{Dec}')$, as follows:

- First, Alice and Bob share *two* random secret key $k_{\mathsf{Enc}}, k_{\mathsf{MAC}} \leftarrow \{0,1\}^\lambda$.
- $\mathsf{Enc}'((k_{\mathsf{MAC}}, k_{\mathsf{Enc}}), m)$:
  1. Compute $c \leftarrow \mathsf{Enc}(k_{\mathsf{Enc}}, m)$.
  2. Compute $\tau = \mathsf{MAC}(k_{\mathsf{MAC}}, c)$.
  3. Output $\mathsf{ct} = (c, \tau)$.
- $\mathsf{Dec}((k_{\mathsf{MAC}}, k_{\mathsf{Enc}}), \mathsf{ct})$:
  1. Parse $\mathsf{ct} = (\mathsf{ct}, \tau)$.

2. Check if $\tau = \mathsf{MAC}(k_{\mathsf{MAC}}, c)$. If this is not the case then output $\perp$.
3. Output $m = \mathsf{Dec}(k_{\mathsf{Enc}}, c)$

Question: Why do we first encrypt and then MAC? Why don't we first MAC and then encrypt – namely, send $\mathsf{Enc}(k_{\mathsf{Enc}}, (m, \mathsf{MAC}(K_{\mathsf{MAC}}, m)))$?

The reason is that if we Encrypt-then-Mac (as is done above) then our resulting scheme is secure against *adaptive chosen ciphertext attacks* (CCA), defined below. CCA-security is the golden standard security notion.

Definition 2. An encryption scheme $(\mathsf{Enc}', \mathsf{Dec}')$ is CCA-secure if for every poly-size $\mathcal{A}$ there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$, $\mathcal{A}$ wins in the game below with probability at most $\frac{1}{2} + \mu(\lambda)$:

1. The challenger chooses a key $k \leftarrow \mathcal{K}_\lambda$.
2. The adversary $\mathcal{A}$ given 1^λ can choose a message $m_i \in \mathcal{M}_\lambda$ and receive $\mathsf{ct}_i \leftarrow \mathsf{Enc}'(k, m_i)$.

   Alternatively, he can choose a ciphertext ct_i and receive $m_i = \mathsf{Dec}'(k, \mathsf{ct}_i)$.

   This step can be repeated polynomially many times, where $\mathcal{A}$ can query $\mathsf{Enc}'(k, \cdot)$ and $\mathsf{Dec}'(k, \cdot)$ polynomially many times.
3. The adversary $\mathcal{A}$ chooses $m_0, m_1 \in \mathcal{M}_\lambda$.
4. The challenger chooses a random bit $b \leftarrow \{0, 1\}$, generates $\mathsf{ct}_b \leftarrow \mathsf{Enc}'(k, m_b)$, and sends the ciphertext ct_b to the adversary.
5. Again $\mathcal{A}$ is given oracle access to $\mathsf{Enc}'(k, \cdot)$ and $\mathsf{Dec}'(k, \cdot)$, but he is not allowed to send the query ct_b to the decryption oracle (but it can send any other ciphertext, even one obtained by changing a single bit of ct_b).
6. The adversary outputs a bit b'.

We say that $\mathcal{A}$ wins if $b' = b$.

Claim 1. Fix any MAC that is existentially unforgeable against adaptive chosen message attacks, and any CPA-secure encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$. Then the authenticated encryption scheme Enc-then-MAC, defined above, is CCA-secure.

Remark. Note that if we change the authenticated encryption scheme to be MAC-then-Enc the resulting scheme would not necessarily be CCA-secure.

Proof. Fix any MAC that is existentially unforgeable against adaptive chosen message attacks, and any CPA-secure encryption. Suppose there exists a poly-size adversary $\mathcal{A}$ that wins in the CCA game with non-negligible probability ϵ. Denote by $m_1, \ldots, m_\ell$ all the queries that $\mathcal{A}$ sent to its encryption oracle, and denote by $(c_1, \tau_1), \ldots, (c_\ell, \tau_\ell)$ the answers it obtains. Denote all the queries that $\mathcal{A}$ sends to the decryption oracle by $\mathsf{ct}'_1, \ldots, \mathsf{ct}'_\ell$, and for every $i \in [\ell]$ parse $\mathsf{ct}'_i = (c'_i, \tau'_i)$.

For every λ denote by

$$\delta(\lambda) := \Pr[\exists i \in [\ell] : \ (\mathsf{MAC}(k_{\mathsf{MAC}}, c'_i) = \tau'_i) \ \wedge \ (c'_i \notin \{c_1, \ldots, c_\ell\})], \tag{1}$$

where the probability above is over the randomness of the CCA game.

We distinguish between two cases:

- **Case 1:** δ is a non-negligible function. In this case we can use $\mathcal{A}$ to break the security of the MAC scheme.

  Specifically, we construct a poly-size adversary $\mathcal{B}$ and prove that it wins in the MAC security game with probability $\geq \frac{\delta}{\ell}$, which is non-negligible. $\mathcal{B}$ does the following:

  1. Sample a random key $k_{\mathsf{Enc}} \leftarrow \{0,1\}^\lambda$ for the CPA-secure encryption scheme.
  2. Emulate $\mathcal{A}$'s oracles, i.e., the encryption and decryption oracles corresponding to the authenticated encryption, using k_{Enc} and the oracle to $\mathsf{MAC}(k_{\mathsf{MAC}}, \cdot)$.
  3. Denote by $\mathsf{ct}'_1, \ldots, \mathsf{ct}'_\ell$ the oracle calls that $\mathcal{A}$ makes to its decryption oracle.
  4. Choose at random $i \leftarrow [\ell]$, and parse $\mathsf{ct}'_i = (c'_i, \tau'_i)$.
  5. Output (c'_i, τ'_i)

  By Equation (1) it holds that with probability $\geq \frac{\delta}{\ell}$ both τ'_i is a valid tag of c'_i and $\mathcal{B}$ didn't send c'_i as an oracle query.

- **Case 2:** δ is a negligible function. In this case, we argue that the decryption algorithm is useless, in the sense that we can simulate the decryption algorithm without knowing the secret key. Formally, we show how to convert $\mathcal{A}$ into a poly-size adversary $\mathcal{B}$ that breaks the CPA security of the underlying encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$.

  $\mathcal{B}$ emulates $\mathcal{A}$, as follows:

  1. a set $S = \emptyset$.
  2. Sample $k_{\mathsf{MAC}} \leftarrow \{0,1\}^\lambda$.

3. Every time that $\mathcal{A}$ queries its encryption oracle with a message $m \in \mathcal{M}_\lambda$, $\mathcal{B}$ does the following:
   (a) Send m to the encryption oracle to obtain $c \leftarrow \mathsf{Enc}(k_{\mathsf{Enc}}, m)$.
   (b) Compute $\tau = \mathsf{MAC}(k_{\mathsf{MAC}}, c)$.
   (c) Let $S = S \cup \{(m, c, \tau)\}$.
4. Every time that $\mathcal{A}$ queries its decryption oracle with a ciphertext $\mathsf{ct} = (c, \tau)$, $\mathcal{B}$ does the following:
   (a) Check if there exists m such that $(m, c, \tau) \in S$. If so, emulate the decryption oracle by outputting m.
   (b) Otherwise, output $\perp$.
5. Once $\mathcal{A}$ chooses the two challenge messages $m_0, m_1 \in \mathcal{M}_\lambda$, $\mathcal{B}$ forwards these challenge messages m_0, m_1 to the challenger.
6. Upon receiving $c_b \leftarrow \mathsf{Enc}(k_{\mathsf{Enc}}, m_b)$ for a random $b \leftarrow \{0, 1\}$, $\mathcal{B}$ proceeds to simulate the encryption and decryption oracles of $\mathcal{A}$ as above, and finally it outputs the guess b' that $\mathcal{A}$ outputs.

The fact that we are in Case 2 implies that there exists a negligible function δ such that for every $\lambda \in \mathbb{N}$,

$$\Pr[b' = b] \geq \epsilon(\lambda) - \delta(\lambda),$$

contradicting the assumption that $(\mathsf{Enc}, \mathsf{Dec})$ is CPA-secure. □

References