CS276: Cryptography October 1, 2015

Message Authentication Codes and CCA2

Instructor: Alessandro Chiesa *Scribe: David Fifield*

1 Previous lecture

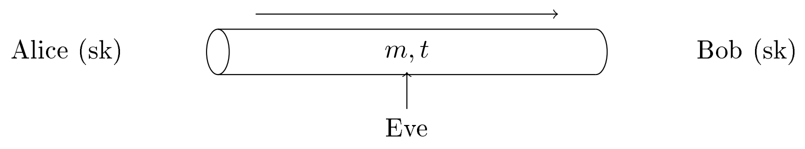
Last time we:

- Constructed a CPA-secure encryption scheme from PRFs.
- Introduced cipher modes of operation and found quibbles (malleability attacks) that caused us seek more than "just" CPA security.
- Defined a security hierarchy for encryption schemes: CPA, CCA1, and CCA2.
- Wanted to patch our CPA-secure scheme to also be secure against an active attacker: prevent the attacker from monkeying with ciphertext.

This last desire takes us on a detour through message authentication codes (MACs).

2 Message authentication codes (MACs)

In brief, a message authentication code guarantees both the *integrity of a message* and the *identity of the sender*. We are in this scenario: Alice and Bob share a secret key sk. Eve is an active attacker capable not only of eavesdropping, but also of modifying messages.



The idea is to attach a *tag* to the message with the property that Eve cannot generate a tag for any message that Alice did not previously send. The tagging scheme needs to satisfy two properties:

Completeness Alice can tag any message m to get a tag t that Bob can verify (together with m and the shared secret key).

Security Eve cannot forge tags for messages she has not already seen tagged.

Informally, a MAC is a pair of algorithms T (for "tag") and V (for "verify"). T outputs a tag given a secret key and a message. V outputs a bit indicating whether a given tag is valid for a secret key and message.

$$T(1^k, \text{sk}, m) \to t$$
$$V(1^k, \text{sk}, m, t) \to \{0, 1\}$$

Note that Alice and Bob share a secret key sk. This is different from the asymmetric (digital signature) setting.

Formally, a MAC is defined as follows:

Definition 1 *A* message authentication code *(MAC) is a pair* (T, V) *of efficient algorithms such that:*

1. *(Completeness)* $\forall k \in \mathbb{N}, \forall \mathrm{sk} \in \{0,1\}^{r(k)}, \forall m \in \{0,1\}^{n(k)}$,
$$\Pr\left[V(1^k, \mathrm{sk}, m, T(1^k, \mathrm{sk}, m)) = 1\right] = 1$$
2. *(Security)* $\forall$ ppt $\mathcal{A}$,
$$\Pr_{\mathrm{sk}}\left[\mathcal{A}^{T(1^k,\mathrm{sk},\cdot)}(1^k) \to (m,t) \text{ s.t. } \begin{cases} V(1^k, \mathrm{sk}, m, t) = 1 \\ T \textit{ was never queried on } m \end{cases}\right]$$
is negligible in k*.*

The "completeness" property says that a tag produced by T must be verifiable by V. The equality holds with probability 1 to allow for some nondeterminism in the algorithms.

The "security" property means that the adversary $\mathcal{A}$ gets oracle access to $T(1^k, \mathrm{sk}, \cdot)$; i.e., it can get a tag for any message it likes but does not know sk directly. The adversary wins if it can produce a message and a valid tag for the message, as long as the message is not one that it previously asked the tagging oracle to tag. Note that m is unquantified. $\mathcal{A}$ wins if it can produce a tag for *any* message; the message could even depend on the results to its queries to the tagging oracle. This differs from the notion of CPA-security; here we are not looking for indistinguishability, but unforgeability. This is a strong notion of security: the adversary only has to produce a tag for any message at all, not necessarily a meaningful one. This definition is known as "existential unforgeability under a chosen message attack (CMA)."

$\mathcal{A}$ cannot simply hard-code a message and tag because the probability is taken over the secret key as well as any random coins used by the algorithms.

Remark 2 *The adversary wins this game if it somehow learns a valid tag for a message without having to consult the oracle. Imagine, for example, that the adversary observes and records a tagged message from Alice to Bob "before" undergoing the oracle experiment. This kind of* replay attack *is out of scope for this definition. In a real system, ideas of message freshness or non-repetition need to be handled by some other layer.*

Remark 3 *One could also imagine giving the adversary oracle access to a verifier, in addition to a tagger. Is this definition equivalent? We were not sure in class. After class, we consulted section 6.1.5.1 of Goldreich's book, which says that any MAC can be transformed into one where tagging is deterministic and there is only one valid tag per message; with such a MAC a verification oracle does not give any extra power.*

3 MACs for short messages

We want to construct a MAC for messages whose length is fixed. Some ideas thaat were suggested in class and turn out not to work:

- Run m through a one-way function with a hardcore predicate. But the tag would be only a single bit and would be forgeable half the time just by guessing.
- Let the tag be the output of a PRG seeded with $\mathrm{sk}|m$: $G(\mathrm{sk}|m)$. Doesn't work because a PRG doesn't work if its input is not random.
- Seed a PRG with sk and XOR it with m: $G(\mathrm{sk}) \oplus m$. This is malleable: one could flip corresponding bits in (m, t) to produce another valid (m', t').

We will use a PRF-based construction. Think of it this way: A PRF is a large table of pseudorandomness. The message m is an index into this table. The adversary cannot predict the table because it does not known the secret key sk.

Theorem 4 *Let f be a PRF. Define algorithms T and V as:*

$$T(1^k, \mathrm{sk}, m) = f_{\mathrm{sk}}(m)$$
$$V(1^k, \mathrm{sk}, m, t) = t \overset{?}{=} f_{\mathrm{sk}}(m)$$

Then (T, V) is a message authentication code.

Proof: Consider algorithms $(\overline{T}, \overline{V})$ that use a true random function $\mathcal{U}$ in place of the PRF f:

$$\overline{T}(1^k, \mathcal{U}, m) = \mathcal{U}(m)$$
$$\overline{V}(1^k, \mathcal{U}, m, t) = \mathcal{U}(m) \overset{?}{=} t$$

We need to show:

(1) $(\overline{T}, \overline{V})$ is a MAC.

(2) $(\overline{T}, \overline{V})$ looks like (T, V).

For part (1), the completeness property of $(\overline{T}, \overline{V})$ is clear. It remains to show the security property. First note that giving an adversary oracle access to $\overline{T}(1^k, \mathcal{U}, \cdot)$ is the same as giving it oracle access to $\mathcal{U}(\cdot)$ because $\overline{T}$ in the construction simply calls $\mathcal{U}$. Its success probability is simply the probability of guessing a random string:

$$\Pr_{\mathcal{U}}\left[\mathcal{A}^{\overline{T}(1^k, \mathcal{U}, \cdot)}(1^k) \text{ forges}\right] = \Pr_{\mathcal{U}}\left[\mathcal{A}^{\mathcal{U}(\cdot)}(1^k) \text{ forges}\right] = 2^{-k} \le \mathrm{negl}(k)$$

$\mathcal{A}$ can forge only by guessing the output of $\mathcal{U}$, which it can do with negligible probability.

For part (2), we need to show that

$$\left|\Pr_{\mathcal{U}}\left[\mathcal{A}^{\overline{T}(1^k, \mathcal{U}, \cdot)}(1^k) \text{ forges}\right] - \Pr_{\mathrm{sk}}\left[\mathcal{A}^{T(1^k, \mathrm{sk}, \cdot)}(1^k) \text{ forges}\right]\right| \qquad (*)$$

is negligible. We will use $\mathcal{A}$ to build an adversary $\mathcal{B}$ that attacks the PRF f. $\mathcal{B}$ will be successful whenever $\mathcal{A}$ is successful. Recall that the PRF adversary is trying to distinguish between a PRF oracle and a truly random oracle.

```
B^O(1^k) = 1. $(m, t) \leftarrow \mathcal{A}^{\mathcal{O}}(1^k)$
           2. Check if $(m, t)$ is a forgery; output 1 if so, 0 if not.
```

Use a hybrid argument to show that the quantity in (*) is negligible. Add and subtract quantities involving $\mathcal{B}^{\mathcal{U}(\cdot)}$ and $\mathcal{B}^{T(1^k,\mathrm{sk},\cdot)}$. Details are omitted. $\square$

A technical note about part (2) of the proof: In $\mathcal{B}^{\mathcal{O}}$, how do you check whether (m, t) is a forgery? In the PRF-based MAC construction, you can do it by querying $\mathcal{O}$ and seeing that its output is the same as t. In fact, you can do that in general using reasoning along the lines of Remark 3.

4 MACs for long messages

Now we look at the construction of a MAC for long messages. "Long" means that it scales to a message length chosen *after* the PRF is already fixed. (The previous construction requires an $n(k)$-bit PRF for an $n(k)$-bit message.)

In class, several ideas were suggested that do not work:

- Split the message into pieces $m_1, \ldots, m_\ell$; MAC each piece individually; and concatenate: $T = f_{\mathrm{sk}}(m_1), \ldots, f_{\mathrm{sk}}(m_\ell)$. Doesn't work because an adversary can omit, repeat, or rearrange pieces and still have a valid tag.
- Do the same, but include the message index in each MAC: $T = f_{\mathrm{sk}}(1, m_1), \ldots, f_{\mathrm{sk}}(\ell, m_\ell)$. Doesn't work because you can swap corresponding pieces of two different messages.
- XOR all tags together: $T = f_{\mathrm{sk}}(m_1) \oplus \cdots \oplus f_{\mathrm{sk}}(m_\ell)$. Doesn't work because you could concatenate two messages and XOR their tags to get a new valid message–tag pair.
- Do something chained like the CBC mode of operation? We'll get to that idea later.

We will use a construction that is along the lines of the first two ideas. The problem with them is that they allow the adversary to rearrange pieces of a single message, mix and match pieces of two separate messages, or extend messages. To thwart that, we will include the message length ℓ and a random number r in each call to the PRF. Think of r as a "session ID" that is generated fresh on each call to T; it prevents the outputs of different calls to T from being swapped around.

Theorem 5 *Let f be a PRF, $\vec{m} = m_1, \ldots, m_\ell$ be a message vector, and $\vec{t} = t_1, \ldots, t_\ell$ be a tag vector. Then*

$$T(1^k, \mathrm{sk}, \vec{m}) = r, f_{\mathrm{sk}}(1, \ell, r, m_1), \ldots, f_{\mathrm{sk}}(\ell, \ell, r, m_\ell)$$
$$V(1^k, \mathrm{sk}, \vec{m}, \vec{t}) = \bigwedge_{i=1}^{\ell} f_{\mathrm{sk}}(i, \ell, r, m_i) \stackrel{?}{=} t_i$$

where r is chosen randomly, is a message authentication code.

Proof: Consider algorithms that use a random function $\mathcal{U}$ in place of the PRF f:

$$\overline{T}(1^k, \mathcal{U}, \vec{m}) = r, \mathcal{U}(1, \ell, r, m_1), \ldots, \mathcal{U}(\ell, \ell, r, m_\ell)$$
$$\overline{V}(1^k, \mathcal{U}, \vec{m}, \vec{t}) = \bigwedge_{i=1}^{\ell} \mathcal{U}(i, \ell, r, m_i) \stackrel{?}{=} t_i$$

We need to show:

(1) $(\overline{T},\overline{V})$ is a MAC.

(2) $(\overline{T},\overline{V})$ looks like (T,V).

For part (1) ($(\overline{T},\overline{V})$ is a MAC), the completeness of the construction is clear. For security, there is now a new degree of freedom, the random value r; we want to live in a world where all r have been picked and none collide. Define the events:

$$\begin{aligned}\mathcal{E}_{\text{FORGE}} &= \mathcal{A}^{\overline{T}(1^k,\mathcal{U},\cdot)}(1^k) \text{ forges}\\ \mathcal{E}_{\text{REP}} &= \mathcal{A}^{\overline{T}(1^k,\mathcal{U},\cdot)} \text{ receives some } r \text{ more than once}\end{aligned}$$

We want to show that $\Pr[\mathcal{E}_{\text{FORGE}}]$ is negligible. We start by bounding $\Pr[\mathcal{E}_{\text{REP}}]$:

$$\Pr[\mathcal{E}_{\text{REP}}] \le \frac{\text{time}(\mathcal{A})^2}{2^k}$$

(The probability that any single pair of r's collides is $\frac{1}{2^k}$; $\mathcal{A}$ can see no more than $\text{time}(\mathcal{A})^2$ pairs; use the union bound.)

We use this to bound $\Pr[\mathcal{E}_{\text{FORGE}} \wedge \mathcal{E}_{\text{REP}}]$.

$$\Pr[\mathcal{E}_{\text{FORGE}} \wedge \mathcal{E}_{\text{REP}}] \le \Pr[\mathcal{E}_{\text{REP}}] \le \frac{\text{time}(\mathcal{A})^2}{2^k} \le \text{negl}(k)$$

We will now bound $\Pr[\mathcal{E}_{\text{FORGE}} \wedge \overline{\mathcal{E}_{\text{REP}}}]$. We have this scenario:

$$\begin{array}{rl} q \text{ queries} \begin{cases} \vec{m}_1 & \to (r_1,\ldots \\ \vec{m}_2 & \to (r_2,\ldots \\ & \vdots \\ \vec{m}_q & \to (r_q,\ldots \end{cases} \\ \text{output} \begin{cases} \vec{m} & \to (r,\ldots \end{cases} \end{array}$$

Because we have that $\overline{\mathcal{E}_{\text{REP}}}$, we know that $i \ne j \implies r_i \ne r_j$. There are three cases to consider:

(i) $\forall i, r_i \ne r$ (r not seen). Then the probability of forgery is 2^{-k}, "just luck."

(ii) $\exists i, r_i = r$ and $\ell_i \ne \ell$ (r seen, different lengths). Then the probability of forgery is 2^{-k}.

(iii) $\exists i, r_i = r$ and $\ell_i = \ell$ (r seen, same lengths). Then there must exist a j such that $m_{i,j} \ne m_j$ (there is some component where $\vec{m}_i$ and $\vec{m}$ differ). Then the probability of forgery is 2^{-k}.

In all cases the probability of forgery is 2^{-k}, so $\Pr[\mathcal{E}_{\text{FORGE}} \wedge \overline{\mathcal{E}_{\text{REP}}}] = 2^{-k}$.

Now, by the law of total probability,

$$\Pr_{\mathcal{U}}[\mathcal{E}_{\text{FORGE}}] = \Pr_{\mathcal{U}}[\mathcal{E}_{\text{FORGE}} \wedge \mathcal{E}_{\text{REP}}] + \Pr_{\mathcal{U}}[\mathcal{E}_{\text{FORGE}} \wedge \overline{\mathcal{E}_{\text{REP}}}].$$

Both terms in the right-hand side are negligible, so their sum is negligible.

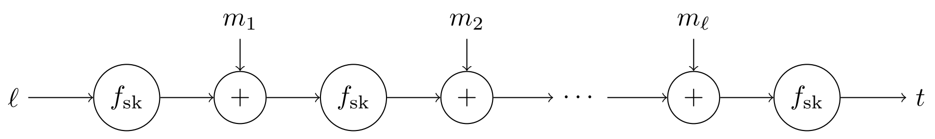
For part (2) ($(\overline{T},\overline{V})$ looks like (T,V)), use a hybrid argument as in the previous section. □

4.1 CBC-MAC

A disadvantage of the previous construction is that the tag vector is as long as the input message vector. One way to achieve shorter tags is to use a construction similar to the CBC mode of operation, called CBC-MAC:

$$t \leftarrow f_{\mathrm{sk}}(m_\ell \oplus f_{\mathrm{sk}}(\cdots \oplus f_{\mathrm{sk}}(m_2 \oplus f_{\mathrm{sk}}(m_1 \oplus f_{\mathrm{sk}}(\ell)))))$$

Perhaps better explained in a diagram:



5 From CPA to CCA2

We detoured into MACs in order to strengthen CPA security against active attackers. The intuition is that a MAC prevents an attacker from creating its own ciphertexts. It can only get decryptions of messages it has asked the oracle to encrypt.

CCA2 has both E and D oracles. (CPA has only E.) We would like to make the D oracle useless, so that the CCA2 adversary has no advantage over the CPA adversary. The intuition is that when the adversary queries the D oracle on a ciphertext protected by a MAC, either:

(i) the ciphertext is something it already got from the E oracle; or

(ii) it can predict the output of D (i.e., $\perp$) with high probability.

We started, but did not finish, the proof of the next theorem.

Theorem 6 *Let (E, D) be a CPA-secure encryption scheme and (T, V) be a message authentication code. Then there is a construction of algorithms (E', D') that is a CCA2-secure encryption scheme.*

Proof: Break sk into two pieces, sk_1 and sk_2. Let

$$\begin{aligned} E'(1^k, \mathrm{sk}, m) = {} & 1.\ c \leftarrow E(1^k, \mathrm{sk}_1, m) \\ & 2.\ t \leftarrow T(1^k, \mathrm{sk}_2, c) \\ & 3.\ \text{output } (c, t) \\ D'(1^k, \mathrm{sk}, (c, t)) = {} & 1.\ \text{check if } V(1^k, \mathrm{sk}_2, c, t) = 1 \\ & 2.\ \text{if so output } D(1^k, \mathrm{sk}_1, c);\ \text{else output } \perp \end{aligned}$$

Suppose for contradiction that (E', D') is not CCA2-secure. Then there exists a ppt $\mathcal{A}$ and $\{m_k^{(0)}\}_k, \{m_k^{(1)}\}_k$ such that

$$\left|\Pr\left[\mathcal{A}^{E'(1^k,\mathrm{sk},\cdot),D'(1^k,\mathrm{sk},\cdot)}(E'(1^k, \mathrm{sk}, m_k^{(0)})) = 1\right] - \Pr\left[\mathcal{A}^{E'(1^k,\mathrm{sk},\cdot),D'(1^k,\mathrm{sk},\cdot)}(E'(1^k, \mathrm{sk}, m_k^{(1)})) = 1\right]\right|$$

(where in both cases the D' oracle is forbidden from answering the specific challenge given to the adversary) is not negligible.

We are going to have to construct *two* adversaries, one for CPA and one for MAC. We begin with the CPA adversary. What's different about this reduction is there's this other oracle, the decryption oracle, that we must somehow simulate.

$$\begin{aligned}\mathcal{B}^{E(sk_1,\cdot)}(c) = \ & 1.\ \text{sample sk}_2 \text{ randomly} \\ & 2.\ t \leftarrow T(1^k, \text{sk}_2, c) \\ & 3.\ \text{output } \mathcal{A}^{\mathcal{O}_1,\mathcal{O}_2}(c,t)\text{, with } \mathcal{O}_1 \text{ and } \mathcal{O}_2 \text{ as defined below}\end{aligned}$$

$\mathcal{O}_1$ is the encryption oracle that we will simulate by calling the E oracle and then manually computing a tag using our own sk$_2$. $\mathcal{O}_2$ is the decryption oracle that we will simulate by always outputting $\perp$. We are assuming that $\mathcal{A}$ never queries $\mathcal{O}_2$ on encryptions that it previously received from $\mathcal{O}_1$; this is without loss of generality because we can always wrap the oracles in a cache that remembers the answers to previous queries.

$$\begin{aligned}\mathcal{O}_1(m_i) \rightarrow c_i &= E(1^k, \text{sk}_1, m_i) \text{ (query the oracle)} \\ t_i &= T(1^k, \text{sk}_2, c_i) \text{ (compute yourself)} \\ \mathcal{O}_2(\cdot) \rightarrow \perp &\end{aligned}$$

We want to show that

$$\left|\Pr\left[\mathcal{A}^{E'(1^k,\text{sk},\cdot),D'(1^k,\text{sk},\cdot)}(E'(1^k,\text{sk},m_k^{(0)}))=1\right] - \Pr\left[\mathcal{A}^{E'(1^k,\text{sk},\cdot),D'(1^k,\text{sk},\cdot)}(E'(1^k,\text{sk},m_k^{(1)}))=1\right]\right|$$

is negligible. We will use a hybrid argument involving the $\mathcal{B}$ adversary. Below we abbreviate the oracles and challenges:

$$\begin{aligned}\left|\Pr\left[\mathcal{A}^{E',D'}(0)=1\right] - \Pr\left[\mathcal{A}^{E',D'}(1)=1\right]\right| \leq & \left|\Pr\left[\mathcal{A}^{E',D'}(0)=1\right] - \Pr\left[\mathcal{B}^{E}(0)=1\right]\right| \\ & + \left|\Pr\left[\mathcal{B}^{E}(0)=1\right] - \Pr\left[\mathcal{B}^{E}(1)=1\right]\right| \\ & + \left|\Pr\left[\mathcal{B}^{E}(1)=1\right] - \Pr\left[\mathcal{A}^{E',D'}(1)=1\right]\right|\end{aligned}$$

We need to show that all three terms on the right-hand side are negligible. The second term, if negligible, contradicts the CPA-security of (E,D). Proofs for the first and third terms will come later. □