COS 533: Advanced Cryptography
Lecture 4 (25 September 2017)
Lecturer: Mark Zhandry

Princeton University
Scribe: Chirag Bharadwaj

Lecture 4: Authentication

Message Authentication Codes

Previously, we saw how to construct CPA-secure secret-key encryption schemes using OWFs[1]. Now, we examine how to construct *CCA*-secure secret-key encryption schemes using more fundamental cryptographic primitives. One way to stop such attacks is to include a "tag" with each encrypted message sent between two parties.

Informally, one might describe the original situation as follows:

1. Alice sends some message m on a channel to Bob.
2. However, Eve, an adversary, intercepts m and instead sends some new m' to Bob.

Now, with the addition of the "tag", the adversary's position might become more challenging:

1. Alice sends some message/tag pair (m, σ) to Bob.
2. Eve intercepts (m, σ) and instead forges some new (m', σ'), which she sends to Bob.

A reasonable goal would be to make forging such a "tag" σ' impossible for the adversary with high probability. We can formalize this notion with the following definitions.

Definition. A *message authentication code (MAC)* is a function $\mathbf{Mac} : \{0,1\}^\lambda \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}$.

Definition. The value $\sigma = \mathbf{Mac}(k, m)$ is a *signature* for the plain-text m under a key $k \xleftarrow{\$} \{0,1\}^\lambda$.

Definition. A *verifier* for a MAC is a function $\mathbf{Ver} : \{0,1\}^\lambda \times \{0,1\}^{m(\lambda)} \times \{0,1\}^{n(\lambda)} \to \{0,1\}$. The value of $\mathbf{Ver}(k, m, \sigma)$ in binary corresponds to whether or not σ is a valid signature for m under k.

So the notion of a signature has formalized the aforementioned "tag", replete with a way to verify whether such a signature is valid. We must still argue for a few things, namely (i) what it means for such a verifier to be "correct" and (ii) what it means for $(\mathbf{Mac}, \mathbf{Ver})$ to be considered secure. We describe these below.

Correctness. For all m, it holds that $\Pr_{k \xleftarrow{\$} \{0,1\}^\lambda}\left[\mathbf{Ver}(k, m, \mathbf{Mac}(k, m)) = 1\right] = 1$.

That is, a verifier is considered *correct* for a MAC if it can almost surely validate a generated signature when given as input the same key and plain-text that the signature was generated with. Before defining security, it could help to design the following experiment.

Experiment. Let A be a PPT adversary with security parameter λ. Then, $\mathbf{EUF\text{-}CMA\text{-}Exp}(A, \lambda)$ is given by:

1. A interacts with a challenger, denoted Ch.
2. At first, Ch chooses a random key $k \xleftarrow{\$} \{0,1\}^\lambda$.
3. Next, A sends Ch a message m. Ch then signs m, producing a signature $\sigma = \mathbf{Mac}(k, m)$, which it sends back.
4. A can repeat Step 3 as many times as it wishes. We will charge A one unit of time for each iteration i.

[1] Actually, we cheated and used OWPs. But it is possible to relax the construction to OWFs.

5. A forges a new message/signature pair (m', σ') and sends it to Ch. Then, Ch performs the following check:

   - $m' \neq m$ (More generally, Ch can check that $m' \notin \{m_i\}$[2].)
   - $\mathbf{Ver}(k, m', \sigma') = 1$.

   If *both* of the above hold, the output of the experiment is 1. Otherwise, it is 0.

Security. The pair of functions $(\mathbf{Mac}, \mathbf{Ver})$ is *EUF-CMA-secure*[3] if for all PPT adversaries A, there exists a negligible function ε such that

$$\Pr[\mathbf{EUF\text{-}CMA\text{-}Exp}(A, \lambda) = 1] < \varepsilon(\lambda).$$

If we examine the definition of the EUF-CMA experiment, we see that the argument to the probability is 1 only when both checks pass Ch in Step 5. This means that A has somehow successfully forged some *new* message/signature pair (m', σ') that passes the verifier on key k. Then, the pair is *secure* if the probability that A does this successfully is sufficiently negligible.

We can also define a stronger notion of security:

Security (Strong)**.** The pair of functions $(\mathbf{Mac}, \mathbf{Ver})$ is *strongly EUF-CMA-secure* if it is EUF-CMA-secure but Ch checks $(m', \sigma') \notin \{(m_i, \sigma_i)\}$ rather than just $m \notin \{m_i\}$ in Step 5(i) of the experiment.

Note that strong EUF-CMA security only differs from the weaker version if there are multiple possible MACs for m.

Constructing MACs from PRFs

To continue the motif from previous lectures, we can also demonstrate how MACs can be built from more primitive cryptographic objects; in this case, we choose to build them out of PRFs.

Construction. Let $F : \{0,1\}^\lambda \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$ be a secure PRF. Then, we can define $(\mathbf{Mac}, \mathbf{Ver})$ as follows:

$$\mathbf{Mac}(k, m) = F(k, m)$$

$$\mathbf{Ver}(k, m, \sigma) = \Big(F(k, m) = \sigma\Big).$$

We can show that this construction satisfies correctness quite trivially from the definitions:

$$\Pr\Big[\mathbf{Ver}(k, m, \mathbf{Mac}(k, m)) = 1\Big] = \Pr\Big[F(k, m) = \mathbf{Mac}(k, m)\Big] = 1.$$

Proving that the construction satisfies strong EUF-CMA security is a little harder. We can do it more easily by introducing additional assumptions. For example, let us assume that an adversary cannot choose a random **Mac** and succeed with high probability, i.e. the function is NOT polynomially-bounded. Then, we can prove the following:

Lemma. If $\dfrac{1}{2^{m(\lambda)}}$ is negligible and F is a secure PRF, then $(\mathbf{Mac}, \mathbf{Ver})$ is strongly EUF-CMA-secure.

Proof. Assume for the sake of contradiction that there exists a PPT adversary $\tilde{A}$ and a non-negligible function ε such that

$$\Pr[\mathbf{EUF\text{-}CMA\text{-}Exp}(\tilde{A}, \lambda) = 1] \geq \varepsilon(\lambda).$$

We will denote the above adversary-challenger apparatus as $\mathbf{Hyb}_0$, i.e. our zeroth hybrid experiment. We will also define a first hybrid experiment $\mathbf{Hyb}_1$ as follows.

[2] We use the notation $\{m_i\}$ to mean the set of all previously-MACed messages m_i over all iterations i of Step 3.
[3] That is, it is existentially unforgeable under chosen-message attacks.

Experiment. Let A be a PPT adversary with security parameter λ. Then, $\mathbf{Hyb}_1(A, \lambda)$ is given by:

1. A interacts with a challenger, denoted Ch.
2. Ch generates an oracle $\mathcal{O} : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$, which has a random output for each input.
3. Next, A sends Ch a message m. Ch then runs $\mathcal{O}$ on m, producing a signature $\sigma = \mathcal{O}(m)$, which it sends back.
4. A can repeat Step 3 as many times as it wishes. We will charge A one unit of time for each iteration i.
5. A forges a new message/signature pair (m', σ') and sends it to Ch. Then, Ch performs the following check:
   - $(m', \sigma') \notin \{(m_i, \sigma_i)\}$
   - $\mathcal{O}(m') = \sigma'$.

   If *both* of the above hold, the output of the experiment is 1. Otherwise, it is 0.

Then, since $\mathcal{O}$ just guesses randomly between 0 and 1 for $m(\lambda)$ bits on each of its outputs, the probability that Step 5(ii) succeeds in the experiment above is just $\frac{1}{2^{m(\lambda)}}$, as the outputs are chosen uniformly at random, independently of the inputs. Formally:

$$\Pr[\mathbf{Hyb}_1(A, \lambda) = 1] \leq \frac{1}{2^{m(\lambda)}}.$$

Note that this result is just an upper bound, as even if the second condition in Step 5 holds serendipitously, the first condition still need not hold—hence the inequality. Since we assumed at the start that $\frac{1}{2^{m(\lambda)}}$ is negligible, this adversary A succeeds with negligible probability.

We now return our attention to $\tilde{A}$ on both hybrid experiments. We claim that it is possible to construct a second adversary B that could successfully distinguish between $\mathbf{PRF\text{-}Exp}_0$ and $\mathbf{PRF\text{-}Exp}_1$. We present such a B below:

Experiment. Let B be a PPT adversary, also with parameter λ. B simulates $\tilde{A}$, playing the role of a challenger:

1. B interacts with a PRF challenger, denoted Ch. B also acts as the EUF-CMA challenger for $\tilde{A}$.
2. Whenever $\tilde{A}$ makes a query m_i on iteration i, B forwards it to its own challenger Ch, obtaining the signature σ_i. B forwards σ_i back to $\tilde{A}$.
3. Next, $\tilde{A}$ responds by forging (m', σ') and sending it to challenger B. Then, B forwards m' to its own challenger Ch, obtaining the signature σ''.
4. B performs the following check:
   - $(m', \sigma') \notin \{(m_i, \sigma_i)\}$
   - $\sigma' \neq \sigma''$.

   If *both* of the above hold, the output of the experiment is 1. Otherwise, it is 0.

Note that here, $\tilde{A}$ is basically in $\mathbf{Hyb}_0$ from the perspective of B, which is running $\mathbf{PRF\text{-}Exp}_0$. Similarly, $\tilde{A}$ would be in $\mathbf{Hyb}_1$ from the perspective of B if it instead ran $\mathbf{PRF\text{-}Exp}_1$ with its own challenger, due to the presence of the oracle $\mathcal{O}$ in both[4]. Thus, we can reduce the problem of distinguishing the PRF experiments to that of distinguishing the hybrid experiments:

$$\Pr\left[\mathbf{PRF\text{-}Exp}_0(B, \lambda) = 1\right] = \Pr\left[\mathbf{Hyb}_0(\tilde{A}, \lambda) = 1\right] \geq \varepsilon(\lambda)$$

and

$$\Pr\left[\mathbf{PRF\text{-}Exp}_1(B, \lambda) = 1\right] = \Pr\left[\mathbf{Hyb}_1(\tilde{A}, \lambda) = 1\right] \leq \frac{1}{2^{m(\lambda)}},$$

[4]Proof omitted.

where both inequalities are from our assumptions above. However, $\varepsilon(\lambda)$ is non-negligible, while $1/2^{m(\lambda)}$ is negligible. Thus, B can somehow distinguish between $\mathbf{PRF\text{-}Exp}_0$ and $\mathbf{PRF\text{-}Exp}_1$ with non-negligible probability, which we know to be impossible. We have reached a contradiction, so it is not possible for $\tilde{A}$ to exist.

It follows that $(\mathbf{Mac}, \mathbf{Ver})$ is indeed EUF-CMA-secure.

□

CCA-secure Secret-Key Encryption

There are at least a few good ways that one could combine the concept of MACs with the CMA-secure secret key encryption schemes that we have seen in the past to synthesize a CCA-secure secret key encryption scheme. Suppose that we already have a MAC pair $(\mathbf{Mac}, \mathbf{Ver})$ and a CMA-secure secret key encryption scheme $(\mathbf{Enc}, \mathbf{Dec})$. We present three plausible ways to construct a CCA-secure secret key encryption scheme $(\mathbf{Enc}', \mathbf{Dec}')$ below.

1. **Encrypt-and-MAC.** In this scheme,
$$\mathbf{Enc}'((k_0, k_1), m) = (\mathbf{Enc}(k_0, m), \mathbf{Mac}(k_1, m)).$$
2. **Encrypt-then-MAC.** In this scheme,
$$\mathbf{Enc}'((k_0, k_1), m) = (\underbrace{\mathbf{Enc}(k_0, m)}_{c}, \mathbf{Mac}(k_1, c)).$$
3. **MAC-then-encrypt.** In this scheme,
$$\mathbf{Enc}'((k_0, k_1), m) = \mathbf{Enc}(k_0, (m, \mathbf{Mac}(k_1, m))).$$

Unfortunately, only one of these will work *all* the time. We examine each one to see what is wrong with two of them:

1. **Encrypt-and-MAC.** This scheme does not work. Recall that we can allow adversaries to repeatedly query the challenger. Since MACs are deterministic, revealing it in the second half of the returned pair means that an adversary can easily learn and forge signatures by repeatedly querying to gather information. No secrecy is guaranteed, so an adversary can easily learn a plain-text message. As a result, this scheme is not even *CPA*-secure, never mind CCA-secure.
3. **MAC-then-encrypt.** This scheme also does not work. We can provide a counterexample to demonstrate this. Suppose F is a secure PRF. Then, we define the following CPA-secure encryption scheme:
$$r \xleftarrow{\$} \{0,1\}^\lambda$$
$$\mathbf{Enc}(k, m) = (r, F(k, r) \oplus m, 0)$$
$$\mathbf{Dec}(k, (r, c, b)) = F(k, r) \oplus c$$
We can define the behavior of an adversary and challenger via the following family of experiments.
Experiment. Let A be a PPT adversary with security parameter λ. Then, $\mathbf{Exp}_b(A, \lambda)$ is given by:
   1. A interacts with a challenger Ch.
   2. At first, Ch chooses a random key and a random string $k, r \xleftarrow{\$} \{0,1\}^\lambda$.
   3. Next, A can make one of two kinds of queries:
      - **Challenge queries.** A sends Ch two messages m_0 and m_1. Ch selects and encrypts m_b, producing a triple $(r, c, 0) = \mathbf{Enc}(k, m_b)$. Then, Ch sends this triple back to A.
      - **CCA queries.** A chooses a random string r, arbitrary cipher-text c, and bit b of its choice, and sends the triple to Ch. First, Ch checks if (r, c, b) was the response to a challenge query. Then, Ch ignores the last bit b and decrypts the triple to produce a plain-text m. It sends this back to A.

This may be CPA-secure, since A cannot learn m from challenge queries alone, but it can learn a plain-text very easily with CCA queries. All A has to do is run a challenge query to obtain the triple $(r, c, 0)$ and then change the last bit to 1 to obtain $(r, c, 1)$. If it passes this to Ch in a CCA query, the challenger will believe it has not seen this triple previously and decrypt the message to m_b:

$$(r, c, 0) = \mathbf{Enc}(k, m_b)$$

$$\begin{aligned}\mathbf{Dec}(k, (r, c, 1)) &= \mathbf{Dec}(k, (r, F(k, r) \oplus m_b, 1)) \\ &= F(k, r) \oplus [F(k, r) \oplus m_b] \\ &= [F(k, r) \oplus F(k, r)] \oplus m_b \\ &= 0 \oplus m_b \\ &= m_b\end{aligned}$$

It seems silly, but this scheme, while secure against CPA attacks, is *not* secure against CCA attacks, since the adversary A can learn which of the two experiments it is in. Then, since $\mathbf{Enc}'$ uses $\mathbf{Enc}$ as the outer-most function call, an adversary for $\mathbf{Enc}'$ can similarly learn plain-texts.

2. **Encrypt-then-MAC.** This scheme works. To prove this, we first set up a family of experiments:

   Experiment. Let A be a PPT adversary with security parameter λ. Then, $\mathbf{IND\text{-}CCA\text{-}Exp}_b(A, \lambda)$ is given by:

   1. A interacts with a challenger Ch.
   2. At first, Ch chooses two random keys $k_0, k_1 \xleftarrow{\$} \{0,1\}^\lambda$.
   3. Next, A can make one of three kinds of queries:

      - **Challenge queries.** A sends Ch two messages m_0 and m_1. Ch selects and encrypts m_b, producing a cipher-text and signature $(c, \sigma) = \mathbf{Enc}'((k_0, k_1), m_b)$, i.e. $c = \mathbf{Enc}(k_0, m_b)$ and $\sigma = \mathbf{Mac}(k_1, c)$. Then, Ch sends this pair back to A.
      - **CPA queries.** A simply sends a message m. Ch encrypts m, producing a cipher-text and signature $(c, \sigma) = \mathbf{Enc}'((k_0, k_1), m)$, i.e. $c = \mathbf{Enc}(k_0, m)$ and $\sigma = \mathbf{Mac}(k_1, c)$. Then, Ch sends this pair back.
      - **CCA queries.** A chooses an arbitrary cipher-text c and signature σ of its choice, and sends the pair to Ch. Then, Ch performs the following check:

        – (c, σ) was not the result of a challenge query.
        – $\mathbf{Ver}(k, c, \sigma) = 1$.

        If *both* of the above hold, then Ch sends $m = \mathbf{Dec}'((k_0, k_1), c) = \mathbf{Dec}(k_0, c)$ back to A. Otherwise, Ch sends a special failure token $\perp$ to A instead.

   4. A can repeat Step 3 as many times as it wishes. We will charge A one unit of time for each iteration i.
   5. Finally, A outputs a guess b' for b. The output of $\mathbf{IND\text{-}CCA\text{-}Exp}_b(A, \lambda)$ is b'.

   We can prove that $(\mathbf{Enc}', \mathbf{Dec}')$ is IND-CCA-secure (under the usual notion of security) by setting up some hybrid experiments, as usual. We do so below:

   - $\mathbf{Hyb}_0(A, \lambda)$ is identical to $\mathbf{IND\text{-}CCA\text{-}Exp}_0(A, \lambda)$.
   - $\mathbf{Hyb}_1(A, \lambda)$ is the following modification to $\mathbf{IND\text{-}CCA\text{-}Exp}_0(A, \lambda)$: for CCA queries, Ch instead performs the following check:
     – (c, σ) was not the result of a challenge query.
     – (c, σ) *was* the result of a CPA query.

- $\mathbf{Hyb}_2(A, \lambda)$ is the following modification to $\mathbf{IND\text{-}CCA\text{-}Exp}_1(A, \lambda)$: for CCA queries, Ch instead performs the following check:
  - (c, σ) was not the result of a challenge query.
  - (c, σ) *was* the result of a CPA query.
- $\mathbf{Hyb}_3(A, \lambda)$ is identical to $\mathbf{IND\text{-}CCA\text{-}Exp}_1(A, \lambda)$.

We can then proceed as in an earlier lecture, showing that there must be some pair of adjacent hybrid experiments that can be distinguished with non-negligible probability, yet failing to come up with such a pair. Such a contradiction would show that the adversary A cannot exist, so (Enc', Dec') is indeed IND-CCA-secure.[5]

Digital Signatures

So far, we have focused on authentication for secret-key encryption schemes. We can actually extend the concept to public-key encryption schemes as well via the notion of a *digital signature*. We formalize this notion below.

Definition. A *signer* is a function $\mathbf{Sig} : \{0,1\}^\lambda \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}$.

Definition. Let **Gen** be a PRG that generates a secret key **sk** and a public key **pk** via $(\mathbf{sk}, \mathbf{pk}) = \mathbf{Gen}(1^\lambda)$. Then, the value $\sigma = \mathbf{Sig}(\mathbf{sk}, m)$ is a *digital signature* for the plain-text m.

Definition. A *public verifier* for a digital signature is a function $\mathbf{Ver} : \{0,1\}^\lambda \times \{0,1\}^{m(\lambda)} \times \{0,1\}^{n(\lambda)} \to \{0,1\}$. The value of $\mathbf{Ver}(\mathbf{pk}, m, \sigma)$ in binary corresponds to whether or not σ is a valid signature for m.

Notice how digital signatures are very similar to MACs, but with public verifiers (since **pk** is public, we can make **Ver** public as well). The notion of correctness and EUF-CMA-security for digital signatures is also virtually identical to that of MACs, with some slight twists due to the presence of **sk** and **pk**. We present these below.

Correctness. For all m, it holds that $\Pr_{(\mathbf{sk},\mathbf{pk}) \leftarrow \mathbf{Gen}(1^\lambda)} \left[\mathbf{Ver}(\mathbf{pk}, m, \mathbf{Sig}(\mathbf{sk}, m)) = 1 \right] = 1.$

That is, a public verifier is considered *correct* for a digital signature if it can almost surely validate a generated signature when given as input the public key and plain-text that the signature was generated with. Before defining security, it could help to design the following experiment.

Experiment. Let A be a PPT adversary with security parameter λ. Then, $\mathbf{EUF\text{-}CMA\text{-}Exp}(A, \lambda)$ is given by:

1. A interacts with a challenger, denoted Ch.
2. At first, Ch runs $(\mathbf{sk}, \mathbf{pk}) = \mathbf{Gen}(1^\lambda)$. It then broadcasts **pk** to A.
3. Next, A sends Ch a message m. Ch then signs m, producing a signature $\sigma = \mathbf{Sig}(\mathbf{sk}, m)$, which it sends back.
4. A can repeat Step 3 as many times as it wishes. We will charge A one unit of time for each iteration i.
5. A forges a new message/signature pair (m', σ') and sends it to Ch. Then, Ch performs the following check:
   - $m' \notin \{m_i\}$.
   - $\mathbf{Ver}(\mathbf{pk}, m', \sigma') = 1$.

   If *both* of the above hold, the output of the experiment is 1. Otherwise, it is 0.

Security. The triple of functions $(\mathbf{Gen}, \mathbf{Sig}, \mathbf{Ver})$ is *EUF-CMA-secure* if for all PPT adversaries A, there exists a negligible function ε such that

$$\Pr[\mathbf{EUF\text{-}CMA\text{-}Exp}(A, \lambda) = 1] < \varepsilon(\lambda).$$

Security (Strong)**.** The triple of functions $(\mathbf{Gen}, \mathbf{Sig}, \mathbf{Ver})$ is *strongly EUF-CMA-secure* if it is EUF-CMA-secure but Ch checks $(m', \sigma') \notin \{(m_i, \sigma_i)\}$ rather than just $m \notin \{m_i\}$ in Step 5(i) of the experiment.

[5] Proof omitted. For a very similar proof, see Lecture 2.

q-time Digital Signatures

We now turn our attention to some variations on the digital signatures that we studied above. For example, would it be possible to tighten the security by changing Step 4 of **EUF-CMA-Exp** so that A can only run the loop for q iterations? This would bound the number of signatures that A could learn.

We denote the above restriction as a *q-time digital signature*. Such signatures can also be strongly- or weakly-CMA-secure. Let us demonstrate how such a device can be built from more primitive PRFs.

Construction. Let $f : \{0,1\}^\lambda \to \{0,1\}^{m(\lambda)}$ be a OWF. Then, we can create a *1-time weakly-secure signature*[6]:

- To create **sk**, we use **Gen** to produce n pairs of random numbers., each of length λ. This might look something like the table shown below (where each $|x_{ib}| = \lambda$):

x_{10}	x_{20}	$\cdot$	$\cdot$	$\cdot$	x_{n0}
x_{11}	x_{21}	$\cdot$	$\cdot$	$\cdot$	x_{n1}

  To create **pk**, we simply "hash" each of these $2n$ numbers using the OWF f. This might look something like the table shown below:

$f(x_{10})$	$f(x_{20})$	$\cdot$	$\cdot$	$\cdot$	$f(x_{n0})$
$f(x_{11})$	$f(x_{21})$	$\cdot$	$\cdot$	$\cdot$	$f(x_{n1})$

  This **pk** is then broadcast to all parties.

- **Sig**$(\mathbf{sk}, m)$ takes a message m, which has length $n(\lambda)$, and "hashes" it bit-by-bit as follows: if bit i is 0, then **Sig** reads off $f(x_{i0})$ from **pk** as y_i. Otherwise, it reads off $f(x_{i1})$ as y_i. After going through all n bits, **Sig** reads off the sequence of hashed results in order to produce $\sigma = y_1 y_2 \cdots y_n$, which gets returned.

- **Ver**$(\mathbf{pk}, m, \sigma)$ "hashes" m using the same process as **Sig** above to produce a sequence $y'_1 y'_2 \cdots y'_n$. It then computes $y'_1 y'_2 \cdots y'_n \oplus \sigma$; if the result is 0, then the verifier has validated the signature, and if the result is non-zero, then the verifier has shown that the signature is invalid. Essentially, the verifier compares the sequence of hashed results bit-by-by with the provided signature to ensure they agree.

Why is this only a 1-time scheme? A second plain-text message must differ in at least one bit from the first, so an adversary can only determine which x_{ib} that it is if it compares the two hashed results and inverts $f(x_{ib})$, which we know occurs with negligible probability. That is, creating additional signatures while reusing the same key tables will cause the security to drop by a factor of 2 as additional bits in **sk** are revealed.

We can extend this construction to *many-time weakly-secure signatures* (i.e. $q > 1$), which we will see later on.

[6] These are sometimes called *Lamport signatures*, after their creator Leslie Lamport.