CSA E0 235: Cryptography 30 Sept 2019

Scribe for Lecture 11

Instructor: Arpita Patra *Submitted by: Nikita Yadav*

1 Contents

In this lecture, we covered the following topics:

- We defined message integrity and message authentication. We motivated message authentication codes (MAC) by showing that secret key encryption is not enough to achieve secure communication. We defined chosen message attack (CMA) and strong chosen message attack (SCMA).
- We constructed secure fixed-length MAC using PRF.
- We looked at domain extension to find MAC tag for arbitrary length message using the fixed-length MAC. We learnt cipher block chaining message authentication code (CBC-MAC) which is a secure and efficient MAC for arbitrary length messages.
- In the last section, we explored information-theoretic (IT) message authentication codes. We argued that perfect security is not achievable for MACs, then we defined and constructed one-time IT-secure MAC using strongly universal functions. We also looked at a more efficient one-time IT-secure MAC - Carter-Wegman MAC.

2 Message Integrity and Authentication

So far in the course we have learnt how to achieve *secret* communication over an open channel. We showed that we can prevent an eavesdropper and even an active adversary from learning the content of message sent over an open channel. But this is not always enough for secure communication. Consider the case of a bank which is providing services over the internet. When the bank receives a request for money transfer from user A to user B, the bank has to consider the following:

- Is the request truly coming from user A? *– issue of message authentication (MA)*
- Are the details of the request as received, such as recipient, amount, etc. exactly those that user A intended to send? *– issue of message integrity (MI)*

If user A had decided to use encryption scheme in this case, then she will encrypt the message and send it over the channel. An adversary can tap the channel and can simply replace the ciphertext or modify the ciphertext in transit, thereby changing the underlying message. This can have serious consequences even though it does not leak any information about the hidden message. Thus, encryption schemes (unless designed for specific purpose of MI and MA) do not provide message integrity and message authentication.

- **Message Authentication:** The receiver should be able to identify whether a message it received was sent by the party claiming to sent it or not.
- **Message Integrity:** The receiver should be able to identify whether a message was modified in transit or not.

Thus message integrity and authentication are also part of *secure* communication. These are important even when privacy is not a concern.

3 Message Authentication Codes

Message authentication code (MAC) is the tool to achieve message integrity and authentication. We will continue to consider the private key setting where the parties share the same secret key. We now define what a MAC is and how it is used. Consider that Alice and Bob wish to communicate in an authenticated manner and they follow the following steps:

- They share a secret key k using some mechanism.
- Alice wants to send a message m to Bob, she computes the MAC tag t based on the message m and key k using **tag-generation algorithm Mac**, t $\leftarrow Mac_k(m)$. She sends (m,t) to Bob.
- Bob verifies whether t is a valid tag on the message m (with respect to the shared key) or not. This is done by running the **verification algorithm Vrfy** that takes as input key k, message m and tag t. It outputs 1 if t is a valid tag of m, otherwise 0.

Note that same key is used for tag generation and verification and hence it is symmetric MAC.

3.1 Syntax of Message Authentication Code (MAC)

A MAC consists of three probabilistic polynomial-time algorithms **(Gen, Mac, Vrfy)** with the following syntax:

- **Key-generation algorithm Gen** takes as input the security parameter 1^n and outputs a key chosen uniformly at random from $\{0,1\}^n$. Running time of this algorithm is O(Poly(n)).
- **Tag-generation algorithm Mac** takes as input a key k, a message m $\in \{0,1\}^*$, and outputs a tag t. This algorithm can be randomized or deterministic, we write it as t $\leftarrow Mac_k(m)$. Running time of this algorithm is O(Poly(n)).
- **Verification algorithm Vrfy** takes as input a key k, a message m and a tag t. It outputs a bit b, with b=1 meaning valid and b=0 meaning invalid. This is usually deterministic and we write it as b:=$Vrfy_k(m,t)$. Running time of this algorithm is O(Poly(n)).

Correctness: It is required that for every n, every key output by Gen(1^n) and every message m $\in \{0,1\}^*$, it holds that $Vrfy_k(m, Mac_k(m)) = 1$.
Any MAC defines the following three spaces (sets):

- **Key space (K)** is the set of all possible keys output by Gen algorithm.
- **Plain-text space (M)** is the set of all possible messages supported by Mac algorithm.
- **Tag space (T)** is the set of all tags generated by Mac algorithm. The sets M and K together define the set T.

Canonical verification: The verification algorithm re-computes the tag t' and check for equality. It outputs 1 if and only if t'=t. Canonical verification can be done only for deterministic MACs (tag-generation algorithm is deterministic).

3.2 Security of Message Authentication Codes

We now define the notion of security for MAC. We consider randomized probabilistic polynomial-time adversary. The idea is that no efficient adversary should be able to generate a valid tag on any new message that was not seen previously.

3.2.1 Chosen Message Attack (CMA)

In Chosen Message Attack (CMA), the adversary is given access to a MAC oracle $Mac_k(.)$; the adversary can adaptively submit any message m of its choice to this oracle, and is given in return a tag t $\leftarrow Mac_k(m)$. We will consider it a break of the scheme if the adversary is able to come up with a valid message and tag pair (m,t), such that it has not seen any MAC tag on message m before.
Let's define it more formally with the following experiment. Consider a message authentication code Π = (Gen, Mac, Vrfy), a PPT adversary A and a value n for the security parameter. We will call this experiment as $Mac - forge^{cma}_{A,\Pi}(n)$.

- The challenger runs the Gen(1^n) algorithm to get a key k.
- The adversary A is given input 1^n and oracle access to $Mac_k(.)$. The adversary obtains MAC tags on arbitrary messages chosen adaptively. We will call this phase as **training phase.** The challenger keeps record of the queried messages in a set Q.
- In the **challenge phase**, the adversary sends a forged tag and message pair (m,t).
- Adversary succeeds if and only if (1) $Vrfy_k(m,t)$=1 and (2) m $\notin$ Q. In this case the output of the experiment is defined to be 1.

Π is CMA-secure if for every PPT adversary A, there is a negligible function negl, such that:

$$Pr(Mac - forge^{cma}_{A,\Pi}(n) = 1) \leq negl \tag{1}$$

3.2.2 Strong CMA (SCMA) security for MAC

CMA does not consider the possibility that an adversary might be able to generate a new tag on a message it queried in training phase. It may be possible for an adversary to forge a different valid tag on some already queried message. We define strong CMA, which captures this possibility. Consider a modified experiment Mac-**s**forge.

- The challenger runs the Gen(1^n) algorithm to get a key k.
- The adversary A is given input 1^n and oracle access to $Mac_k(.)$. The adversary obtains MAC tags on arbitrary messages chosen adaptively. We will call this phase as **training phase. The challenger keeps pairs of queried messages and their associated tag responses in some set Q.**
- In the **challenge phase**, the adversary sends a forged tag and message pair (m,t).
- Adversary succeeds if and only if (1) $Vrfy_k(m,t)$=1 and **(2) (m,t)** $\notin$ **Q**. In this case the output of the experiment is defined to be 1.

Π is **strong** CMA-secure if for every PPT adversary A, there is a negligible function negl, such that:

$$Pr(Mac - \mathbf{s}forge^{cma}_{A,\Pi}(n) = 1) \leq negl \tag{2}$$

If a MAC is deterministic (canonical verification) and cma-secure then it is also scma-secure. This is because for every message only one valid tag exists and if the adversary has queried a message m once, it is not possible for it to come up with a different valid tag for the message m. This also shows that randomization is not necessary for constructing a scma-secure MAC.

3.3 Replay Attacks - Not Captured in MAC Security Definition

MACs do not offer protection against replay attacks. In replay attack, an attacker can re-send a previously sent message and MAC tag to an honest party. Consider again the scenario where Alice sends a request to bank to transfer \$1000 from her account to Bob's account. For doing so, Alice computes a MAC tag and append it to the request so that the bank knows that the request is authentic. If MAC is secure, then an attacker (maybe Bob) will not be able to modify the request (hence the amount) because this will require the attacker to come up with a valid tag for the new message, which is hard for secure-MAC. But nothing prevents the attacker from replaying Alice's request multiple times to the bank and each request will be authentic. The bank servers the requests and it can have severe consequences.

This is not taken care in MAC definition because whether this attack is of concern or not depends on the application scenario and can be handled by higher-level applications. Common techniques for preventing replay attacks are to use sequence numbers or time-stamps.

4 Constructing Secure Message Authentication Codes

4.1 A Fixed-Length MAC

If there is a function ℓ such that for every key k output by Gen(1^n), algorithm Mac_k is only defined for messages m $\in \{0,1\}^{\ell(n)}$, then we call the scheme a fixed-length MAC for messages of length ℓ(n). Now we show a construction of secure fixed-length MAC by using pseudorandom functions. Intuitively, if our tag-generation algorithm is just a pseudorandom function, then given a message m, tag t := $F_k(m)$. Then forging a tag on a new message is equivalent to guessing the output of pseudorandom function at a new input point. The probability of guessing the output of a *random* function at a new input point is 2^{-n}. The probability of guessing such a value for a pseudorandom function can be only negligibly greater.
Let F: $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a pseudorandom function. We define a fixed-length MAC Π = (Gen, Mac, Vrfy) for messages of length n as follows:

- **Gen(1^n)** takes as input the security parameter and outputs a key chosen uniformly at random from $\{0,1\}^n$.
- **Mac** takes as input k $\in \{0,1\}^n$ and a message m $\in \{0,1\}^n$, outputs the tag := $F_k(m)$. If $|m| \neq |k|$ then output nothing. This is a **deterministic Mac** algorithm.
- **Vrfy** takes as input k $\in \{0,1\}^n$, a message m $\in \{0,1\}^n$, and a tag t $\in \{0,1\}^n$, outputs 1 if and only if t $= F_k(m)$. If $|m| \neq |k|$ then output 0. This is a **canonical verification** algorithm.

Theorem 1. If F is a PRF then Π is a cma-secure MAC.

Proof. Let A be a PPT adversary. Consider the message authentication code $\Pi' = (Gen'$, Mac', $Vrfy')$ which is same as $\Pi = (Gen, Mac, Vrfy)$ in above construction except that a truly random function f is used instead of the pseudorandom function F_k. It follows that

$$Pr[Mac - forge_{A,\Pi'}(n) = 1] \leq 2^{-n}$$

because for any new message m the value t $= f(m)$ is uniformly distributed in $\{0,1\}^n$ from the view point of A.
We have to show that there is a negligible function negl such that

$$|Pr[Mac - forge_{A,\Pi}(n) = 1] - Pr[Mac - forge_{A,\Pi'}(n) = 1]| \leq negl$$

Assume that Π is not secure and Pr[$Mac - forge_{A,\Pi}$(n) $= 1] > \frac{1}{p(n)}$. Then we can build a distinguisher D who is given access to an oracle O of a function and it's aim is to determine whether this function is a TRF or PRF. The reduction steps are as follows:

- Run A(1^n). Whenever A requests for a tag on a message m, D queries the oracle with m and obtains response t, and returns t to A.
- When A outputs (m', t') at the end of the execution, D queries O with m' and obtains response t^c.

- D outputs 1 if $t^c = t$ and A never queried oracle O on message m' before.

When D's oracle access is a PRF, then view of A is identical to the view of $Mac{-}forge_{A,\Pi}$(n) and D outputs 1 only when $Mac - forge_{A,\Pi}$(n) = 1. Therefore

$$Pr[D^{F_k(.)}(1^n) = 1] = Pr[Mac - forge_{A,\Pi}(n) = 1]$$

When D's oracle access is a TRF, then view of A is identical to the view of $Mac - forge_{A,\Pi'}$(n) and D outputs 1 only when $Mac - forge_{A,\Pi'}$(n) = 1. Therefore

$$Pr[D^{f(.)}(1^n) = 1] = Pr[Mac - forge_{A,\Pi'}(n) = 1]$$

Since F is a PRF, there exists a negligible function negl such that:

$$|Pr[D^{F_k(.)}(1^n) = 1] - Pr[D^{f(.)}(1^n) = 1]| \leq negl$$

This gives us that

$$\begin{aligned}|Pr[Mac - forge_{A,\Pi}(n) = 1] - Pr[Mac - forge_{A,\Pi'}(n) = 1]| &= |Pr[D^{F_k(.)}(1^n) = 1] - Pr[D^{f(.)}(1^n) = 1]| \\ &\leq negl\end{aligned}$$

□

4.2 Domain Extension for MACs

In this section, we show how to construct a general MAC scheme, that can handle arbitrary length messages using fixed-length MAC scheme for messages of length n. When we discussed domain extention for SKE, we divided the message into blocks and then encrypted each block using fixed-length scheme (minimum security notion CPA). This method was inefficient in terms of ciphertext length, then we looked at various efficient modes of operation such as CBC, OFB and CTR. In case of MAC, simply dividing the message into blocks and generating tag for each block seprartely does not even provide security, we need to employ additional tricks to make it work which we are going to look at in this section. In the next section, we will look at efficient MAC scheme for arbitrary length messages.
Let Π = (Mac, Vrfy) be a secure-fixed length MAC for messages of length n and the message to be authenticated is m = $m_1, m_2, ..., m_d$; where every block m_i is of length n. We first discuss some simple ideas which do not work.

- **Authenticate each block separately:** We compute $t_i := Mac_k(m_i)$ for all i, and output (t_1, t_2, ..., t_d) as the tag.
  What do we achieve? This scheme will prevent an adversary from coming up with a valid tag for some new message.
  Possible attack - Block re-ordering attack: The adversary can reorder the tag and corresponding message blocks to create a new message and valid tag pair. For example, consider m = (m_1,m_2) where $m_1 \neq m_2$ has tag t = (t_1,t_2), then (t_2, t_1) is a valid tag on the different message (m_2, m_1).

- **Authenticate block index along with each block:** We compute $t_i := Mac_k(i||m_i)$ for all i, and output (t_1, t_2, ..., t_d) as the tag.
  What do we achieve? This scheme will prevent the previous attack by authenticating a block index with each block so that re-ordering does not give a valid message and tag pair.
  Possible attack - truncation attack: The attacker can simply drop the blocks from the end and their corresponding tags from the tag string. For example, $(t_1, t_2, ..., t_{d-1})$ is a valid tag for message m = (m_1, m_2, ..., m_{d-1}).

- **Authenticate message length and block index with each block:** We compute $t_i := Mac_k(\ell||i||m_i)$ for all i, where ℓ is the length of message m in bits and output (t_1, t_2, ..., t_d) as the tag.
  What do we achieve? This scheme prevent truncation attack by additionally authenticating the message length with each block.
  Possible attack - mix-and-match attack: The adversary combines blocks from two different messages of same length to create a new message. For example, if adversary obtains ($m_1||m_2||m_3$, $t_1||t_2||t_3$) and ($m'_1||m'_2||m'_3$, $t'_1||t'_2||t'_3$) where $|m| = |m'|$. Then ($m_1||m'_2||m_3$, $t_1||t'_2||t_3$) is a valid new (message, tag) pair.

- **Authenticate fresh random identifier, message length, block index with each block:** We compute $t_i := Mac_k(r||\ell||i||m_i)$ for all i, where ℓ is the length of message m in bits, r is a random identifier which is chosen fresh for each message. We output (t_1, t_2, ..., t_d) as the tag.
  What do we achieve? This scheme prevent mix-and-match attack by including a random message identifier with each block which prevents blocks from different messages from being combined.
  This construction is secure. But this is highly inefficient. Since in each block a random identifier, message length and block index are stored apart from message bits, the number of message bits present in a block is less than n since block length is fixed. Infact, each block has only n/4 message bits. 3n/4 bits are occupied by random identifier, message length and block index in each block. So, if $|m|$ = dn bits, then it requires 4d invocations of Mac algorithm and tag size is 4dn bits.

4.3 CBC-MAC

CBC-MAC is an efficient secure-MAC for arbitrary length messages. The construction of CBC-MAC uses pseudorandom functions. Let F: $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a pseudorandom function, whose key k is agreed upon by Alice and Bob beforehand. Let Alice has a message m with length dn, where d is a polynomial in n. The CBC-MAC construction is as follows:

- **Mac:** On input key k $\in \{0,1\}^n$ and message m of length dn do the following:
  1. Parse m as m = $m_1, m_2, ..., m_d$ where each m_i is of length n.
  2. Set $t_0 = |m|$. Then, for i = 1 to d:
     Set $t_i := F_k(t_{i-1} \oplus m_i)$.

Output t_d as the tag.

- **Vrfy:** On input key k $\in \{0,1\}^n$, a message m, and a tag t, output 1 if and only if t $= Mac_k(m)$. This is canonical verification.

This construction is secure. The tag consist of only n bits and only d invocations of PRF are required. This is highly efficient scheme. Note that in the construction we prepend the length of message m. However, appending the length gives an insecure scheme.

5 Information-theoretic MACs

In previous sections, we have explored message authentication codes with computational security, i.e., where the bounds on the attacker's runnning time are assumed. In this section, we will see under which conditions information-theoretic security - security against a computationally unbounded adversary is attainable. It is impossible to achieve perfect security - probability of an adversary producing a valid tag on a new message is 0. This is because the adversary can simply guess a tag t on any message and the guess will be correct with probability at least $2^{-|t|}$, where $|t|$ denote the tag length of scheme. IT-security can be achieved only if we place some bound on the number of messages authenticated by the honest parties. We look at the most basic setting, where the honest party authenticate just a single message. We call this one-time message authentication.

5.1 One-time IT-Secure MAC

Now we construct a one-time IT-secure MAC. As mentioned in previous paragraph, 1-time MAC can be used to authenticate a single message and forgery should be difficult based on a single previously seen authentication. We formalize this requirement by considering the modified Mac-forge experiment in which adversary is computationally unbounded and restricted to a single query. Let Π = (Gen, Mac, Vrfy) be MAC scheme.
One-time message authentication experiment $Mac-forge_{A,\Pi}^{1-time}$

- The challenger runs the Gen algorithm to get a key k.
- Unlike the CMA game, the adversary A can query $Mac_k(.)$ oracle only for a single message, say m. It is given back tag, t $\leftarrow Mac_k(m)$.
- In the **challenge phase**, the adversary sends a forged tag and message pair (m', t').
- Adversary succeeds if and only if (1) $Vrfy_k(m', t')$=1 and (2) $m' \neq$ m. In this case the output of the experiment is defined to be 1.

A MAC Π = (Gen, Mac, Vrfy) is said to be one-time $\varepsilon-$secure, if for all adversaries (even unbounded):

$$Pr(Mac-forge_{A,\Pi}^{1-time} = 1) \leq \varepsilon \tag{3}$$

5.2 Constructing one-time IT-secure MACs

In this section we show how to build $\varepsilon-$secure MAC based on **strongly universal function (SUF)**. We then look at a simple construction of strongly universal function.
Let $h : K \times M \to T$ be a keyed function whose first input is a key from key space K and second input is from message space M. The output is denoted by $h_k(m)$ and belongs to tag space T. Then h is *strongly universal (or pairwise independent)* if for any two distinct inputs $m, m' \in M$ and uniformly chosen k $\in K$, the values $h_k(m)$ and $h_k(m')$ are uniformly and independently distributed in T.

Definition 1 (Strongly universal function)**.** A function $h : K \times M \to T$ is strongly universal if for all distinct $m, m' \in M$ and all $t, t' \in T$ it holds that

$$Pr[h_k(m) = t \ \wedge \ h_k(m') = t'] = 1/|T|^2 \tag{4}$$

where the probability is taken over uniform choice of $k \in K$. $\diamondsuit$

Now we construct an IT one-time $(1/|T|)$-secure MAC with strongly universal function $h : K \times M \to T$. Define a MAC for messages in M as follows:

- **Gen:** Choose uniform k $\in K$ and output it as the key.
- **Mac:** It takes as input key k $\in K$ and message m $\in M$, and outputs the tag t := $h_k(m)$.
- **Vrfy:** It takes as inputs a key k $\in K$, a message m $\in M$ and a tag t $\in T$, it outputs 1 if and only if t = $h_k(m)$.

Theorem 2. *If h is SUF then the above construction is an IT one-time $(1/|T|)$-secure MAC.*

Proof. Let A be an adversary. A receives (m,t) and outputs (m', t'). We can consider that the output pair (m', t') is a deterministic function of tag t and message m that A received. Then we have

$$\begin{aligned} Pr[Mac - forge^{1-time}_{A,\Pi} = 1] &= \sum_{t \in T} Pr[Mac - forge^{1-time}_{A,\Pi} = 1 \ \wedge \ h_k(m) = t] \\ &= \sum_{t \in T} Pr[h_k(m') = t' \ \wedge \ h_k(m) = t] \\ &= \sum_{t \in T} \frac{1}{|T|^2} \\ &= \frac{1}{|T|} \end{aligned}$$

This proves the theorem. $\square$

Note that for a one-time (1/p) MAC, forging probability is negligible for a sufficiently large value of p.

5.3 Constructing SUF over the Field $(Z_p, +, .)$

Let us define $Z_p = \{0, ..., p-1\}$, where p is a prime. We define message space $M = Z_p$, tag space $T = Z_p$ and the key space $K = Z_p \times Z_p$. A key k = (a,b) consists of a pair of elements from Z_p. We define h as $h_{a,b}(m) = [a.m + b \ mod \ p]$.

Theorem 3. For any prime p, the function h is strongly universal.

Proof. Consider two distinct messages $m, m' \in M$ and their corresponding tags $t, t' \in T$. Then there exists a unique key (a,b) such that $h_k(m) = t$ and $h_k(m') = t'$ only if $a.m + b = t \ mod \ p$ and $a.m' + b = t' \ mod \ p$. We thus have two linear equations in two unknowns a, b. These two equations are both satisfied exactly when a = $[(t-t').(m-m')^{-1}$ mod p] and b = [t - a.m mod p]. This gives us that

$$\begin{aligned} Pr[h_k(m) = t \ \wedge \ h_k(m') = t'] &= Pr[a = ((t-t').(m-m')^{-1} \ mod \ p) \ \wedge b = (t - a.m \ mod \ p)] \\ &= \frac{1}{|K|} \\ &= \frac{1}{|T|^2} \ as \ required. \end{aligned}$$

□

The disadvantage of this construction is that the **key is twice as large as the message**.

5.4 A more efficient one-time IT-secure MAC

In this section we look at a more efficient one-time IT-secure MAC - **Carter-Wegman MAC**.
Tag-generation algorithm: It takes input m = $(m_1, m_2, ..., m_v)$ where m $\in Z_p^{\leq \ell}$ and key k = (a,b) where k $\in Z_p^2$. It computes tag t based on m and key k as t := $(a^{v+1} + m_1.a^v + ... + m_v.a)$ + b; t $\in Z_p$.

Theorem 4. Carter-Wegman MAC is a IT one-time $(\frac{\ell+1}{p})$-secure MAC.

Proof. A key k = (a,b) is chosen randomly from the space Z_p^2. Let adversary A learns (m,t) where t := $Tag-gen_{a,b}(m)$. The adversary outputs a tag t' for a new message m'. We will now show that the probability that the tag t' is a valid tag for message m' is $(\frac{\ell+1}{p})$.
Let $m' = (m'_1, m'_2, ..., m'_u)$ and $m = (m_1, m_2, ..., m_v)$. Let $f(X) = X_{v+1} + m_1.X_v + ... + m_v.X$ and $g(X) = X_{u+1} + m'_1.X_v + ... + m'_v.X$. Define $h(X) = f(X) - g(X)$.
Using above definitions, tag t := $f(a) + b$ and tag $t' := g(a) + b$. Then $t - t' = f(a) - g(a) = h(a)$.

$$Pr[t' := Tag - gen_{a,b}(m')] = Pr[a \ is \ root \ of \ (h(X) - (t - t'))] = \frac{\ell+1}{p}$$

□

6 References

[1] Arpita Patra, `https://www.csa.iisc.ac.in/~cris/e0_235.html`. Course Materials.
[2] Introduction to the Modern Cryptography by Jonathan Katz and Yehuda Lindell, second edition 2014.
[3] A Graduate Course in Applied Cryptography by Dan Boneh and Victor Shoup.