CSE 594 : Modern Cryptography Date:03/02/17

Lecture 11: Message Authentication

Instructor: Omkant Pandey *Scribe: Venkata Jyothsna Donapati,Malini Mahalakshmi Venkatachari*

1 Message Authentication

So far we have discussed about pseudo random generators, pseudo random functions,symmetric encryption.All those primitives are about hiding information.Authentication is about validating if message came from a reliable source. Say for example if message claims its from FBI we want to make sure it came from FBI.

2 How do we Authenticate?

2.1 Symmetric Encryption

Two parties share a key and one party sends an encrypted message and the second party decrypts the message and sends the message back so the sender can verify.But the problem with this approach is we are validating the person but not the message and symmetric encryption just hides the information and what if someone taps onto message and tampers with it? so symmetric encryption looks like a good idea but doesn't really work.

2.2 How about Pseudo random functions ?

Pseudo random functions are basically random functions which takes key and message and generate random values.Pseduo random function generates random output every time but generates same value for a message.The sender and receiver share a key and the sender queries pseudo random function on a message and sends the received value(σ_1) and message to receiver.The receiver also queries pseudo random function on that message and gets (σ_2) and compares if (σ_1) and (σ_2) are equal.Please note that here the message is not hidden we are just authenticating the message.

3 Message Authentication Code

The sender generates message authentication code(MAC) and sends it along with the message and the receiver verifies it. What should Message Authentication Code do ?

- Correctness
- Unforgeability

3.1 Correctness

If the MAC is generated by an authorized source it should verified correctly.

3.2 Unforgeability

Only the authorized source can generate a MAC.By authorized we mean that whoever has the key can only generate the MAC.No one should be able to forge the MAC.

(Note - In MAC we provide authentication by sharing the secret key unlike digital signatures(key is public)).

3.3 Steps involved in MAC:

A message authentication code (MAC) consists of $\{M$, K,KG, Tag, Verify$\}$ where M, K are message space and key space respectively.

1. Key generation: KG(1^n) is a PPT key generation algorithm it returns a k $\in K$.
2. MAC generation: Tag(k, m) runs in polynomial time and takes a key $k \in K$ and a message $m \in M$ as input and outputs a code σ .
3. Verification: Verify(k, m, σ) is a polynomial time algorithm which takes key k, a message m, and a code as input and verifies the code and outputs 1 if it can verify correctly 0 otherwise.

The above scheme must satisfy:

1. Correctness: It should verify correctly for a correct code σ generated by reliable source that is for $k \in K$ and a message m Verify(k, m,Tag$(k, m)) = 1$
2. Unforgeability: No efficient algorithm can win the forging game with more than negligible advantage.That is there is negligible function μ such that Adversary wins the game is close to zero(negligible).That is $\forall$ non-uniform PPT A,$\exists$ negligible μ s.t. $\forall$ n:Pr[A wins forging game] $\leqslant \mu$(n)

4 Goal of Adversary

Here the goal of adversary is to generate a MAC for a message for which the adversary hasn't seen the MAC before. Here the adversary has to generate a message MAC pair after learning as many message and MAC pairs (m1, σ_1),(m2, σ_2),... and adversary can ask for MACs for the messages of his choice.

4.1 Forging Game

Forging game is that Adversary should forge a MAC after learning multiple message MAC pairs.The following are the steps involved in forging game

1. Initialization : The challenger simulates a game and generates a key: k $\leftarrow$ KG(1^n)
2. Learning : Adversary learns as many codes on messages of his choice.
   - Adversary sends a message $m_i \in M$ to Challenger
   - Challenger sends back a code $\sigma_i \leftarrow$ Tag(k, m_i)

Let L = $\{m_i\}$ be the set of messages Adversary sends to Challenger

3. Guess: Adversary outputs a message-code pair (m, σ) for a m that is not in L and Adversary wins only when he can forge the MAC correctly i.e., if m $\notin$ L $\wedge$ Verify$(k, m, \sigma) = 1$.

As discussed before MACs require the two parties to share a secret key.On the other hand *DigitalSignatures* use public-key variant instead od sharing the secret-key.We will discuss about them in later classes.

5 A MAC based on PRF

Theorem 1 $PRF \implies MAC$

Construction of MAC:
Let F be a function from a family of pseudorandom functions.Let the input space be as follows:

1. Message Space $M = \{0,1\}^n$
2. Key space $K = \{0,1\}^n$
3. KG $\leftarrow$ Key _ Generation _ Algorithm
4. $Tag_k(M) \rightarrow$ Value of PRF on the message that we want to authenticate.
5. $Verify(m, \sigma) \rightarrow$ Checks if value of σ is equal to output of PRF on message.

If there exists a PRF , then the above scheme is a MAC.

5.1 Correctness

The correctness follows from deterministic nature of PRF. If it generates correctly then

$$verify\,(k, m, Tag\,(k, m)) = 1$$

5.2 Unforgeability:

Unforgeability can be proved by using hybrid arguments. Instead of PRF if we start providing values of a Random function R, the adversary should not be able to differentiate as long as we keep the inputs consistent.(i.e) for a fixed I/P , we should provide a fixed output. As the adversary has seen only the random function on the inputs that he has queried so far

$$\text{p(Guessing value of Random function on I/P that he has not seen)} = \tfrac{1}{2^n}$$

This is essentially because whatever we know so far does not contribute to what comes next.

$$\text{p(guessing it right)} = \tfrac{1}{2} + \mu\,(n)$$

If the adversary has to guess with a noticeable probability then he has to possess the knowledge of the algorithm that populates the table for random function.

Suppose our MAC is not unforgeable, then it implies a PPT adversary wins the forging game with a noticeable probability ϵ.

Therefore by definition A outputs $(m, \sigma) \ni \sigma = F_k(m)$ with a probability $\epsilon \ni \forall m \notin L$ where L is the list of messages queried by A.

Let us now say that forging games now uses a truly random function RF instead of F.

In the ForgingGame, the challenger does not use F to answer As queries; instead:

1. It builds a table T (to represent the truly random function RF)
2. For each new m_i, sends a random σ_i, and stores (m_i, σ_i) in T.
3. For each existing m_i, simply returns the entry in $T[m_i]$.

Considering Hybrid Arguments for this case
H0 := Forging game with PRF $\rightarrow$ A wins with probability ϵ
H1 := Forging game with PRF replaced by Random Function $\rightarrow$ A wins with probability ϵ'

By security of PRF we have

$$|\epsilon - \epsilon'| <= \mu\,(n) \text{where } \mu \text{ is negligible}$$
$$\implies \epsilon' >= \epsilon - \mu\,(n)$$

But we have RF to be a truly randon function. It implies no one can guess $RF(m) = \sigma$ with a probability $> \frac{1}{2^n}$.
If he is winning the first experiment with ϵ then he should be winning the second part as well as he can't tell the difference between the two.

$$\epsilon' \leq 2^-n$$
$$\implies \epsilon - \mu \leq 2^-n$$
$$\implies \epsilon \leq 2^-n + \mu$$
$$\implies \epsilon \text{ is not noticeable}$$

It is a contradiction. This implies our scheme is secure.

5.3 One-time MAC

It provides weaker security. Adversary is allowed only one query. It might take time but it can be eventually broken. But it is advantageous as it can be unconditionally secure. It is analogous to OTP for authentication