CS 600.442 – Modern Cryptography October 5, 2016

Lecture 8: Authentication

Instructor: Abhishek Jain *Scribe: Katie Chang*

1 Introduction

We begin by describing the setting. There are two parties, Alice and Bob. Alice wants to send a message to Bob over a public channel such that an active adversary Eve cannot tamper with the message without being detected. In particular, Eve should not be able to replace the message sent by Alice with another message and trick Bob into thinking that it actually came from Alice.

We will discuss two methods using which Bob will be able to verify whether or not the message he receives is sent by Alice. The first method, called *message authentication codes*, requires Alice and Bob to share a secret key. The second method, called *digital signatures*, is the public-key analogue, where Alice will use a secret key and Bob will use a corresponding public key.

Note: We will abbreviate Alice = A, Bob = B, Eve = E.

2 Private Key: MAC

We first consider the private key scenario. We will discuss message authenticated codes, or MAC, where A and B share a private key that is used both for signing and verifying.

2.1 Algorithm overview

MACs have three relevant algorithms. First, a key generator Gen() that outputs a private key. Second, we have a tag generator Tag() that takes as input a signing key and a message and outputs a signature. Finally, there is a verification algorithm that takes as input a verification key, a message and a signature and outputs 1 if the signature is valid, and 0 otherwise.

- Key Generation: $k \leftarrow \text{Gen}(1^n)$, generates a secret key
- Signing: $\sigma \leftarrow \text{Tag}_k(\text{m})$, computes a tag for the message m
- Verification: $\text{Ver}_k(\text{m}, \sigma) = 1$ iff σ is a valid tag of m over k, else 0.

MACs also have the following two properties: Correctness, and Security. With respect to security, we will discuss Unforgability under Chosen-Message Attack (UF-CMA).

Definition 1 (MAC) *Given a key k and a signature σ, MACs have the following properties:*

- ***Correctness:*** $Pr[k \leftarrow Gen(1^n), \sigma \leftarrow Tag_k(m) : Ver_k(m, \sigma) = 1] = 1$
- ***Security:*** *$\forall$ nu PPT adversary A, $\exists$ a negligible function $\mu(\cdot)$ such that:*
  $Pr[k \leftarrow Gen(1^n), (m, \sigma) \leftarrow A^{Tag_k(\cdot)}(1^n) : A \text{ did not query } m \wedge Ver_k(m, \sigma) = 1] \leq \mu(n)$

Another way of looking at the security of MACs is via the following security game. Let Ch = Challenger, A = Adversary.

1. Ch generates a key: $k \leftarrow \text{Gen}(1^n)$
2. A sends a message m_i to Ch
3. Ch generates a tag: $\sigma_i \leftarrow \text{Tag}_k(m_i)$
4. Ch sends σ_i to A
5. A and Ch repeat steps 2 to 4 multiple times
6. A finally sends Ch (m^*, σ^*)

A wins if, for all i, $m^* \neq m_i$, and $\text{Ver}_k(m^*, \sigma^*) = 1$. Unforgeability under Chosen-Message Attack (UF-CMA) security states that the probability that A wins the above game is negligible.

3 Construction of MAC

We now construct a MAC scheme based on pseudorandom functions (PRFs).

Theorem 1 *PRF $\Rightarrow$ MAC*

In order to prove the above theorem, we start by giving a construction of MAC scheme based on PRFs.

Construction. Let $\{f_k\}$ be a family of PRFs. The MAC scheme is described below:

1. Gen(1^n) : Output k $\overset{\$}{\leftarrow}$ $\{0, 1\}^n$
2. Tag$_k$(m) : Output f$_k(m)$
3. Ver$_k$(m, σ) : Output f$_k(m) \overset{?}{=} \sigma$

We now prove the security of the above construction.

Proof of Security. Let A_{MAC} be a PPT adversary that breaks the security of MAC. We want to construct an adversary A_{PRF} that uses A_{MAC} to break the security of PRFs and thus reach a contradiction and conclusion.

Whenever A_{MAC} queries A_{PRF} with a message m_i, A_{PRF} queries its challenger oracle with $x_i = m_i$. A_{PRF} will receive the oracle's output y_i. A_{PRF} will then forward $\sigma_i = y_i$ to A_{MAC}. This process repeats several times. Now, A_{MAC} outputs (m^*, σ^*). At this point, A_{PRF} queries its challenger on $x^* = m^*$ and obtains y^*. If $\sigma^* = y^*$, then A_{PRF} outputs "PRF", else it outputs "RF".

If the Challenger oracle is random, then A_{MAC} has no information on the PRF key, and therefore the probability that it can output a valid signature σ^* is negligible.

If the Challenger oracle is pseudo-random, then our A_{MAC} above will be able to output a valid σ with non-negligible probability. This is a contradiction on the security of PRFs. ■

3.1 One Time MACs

We can also consider MACs with a weaker security definition where the adversary is only allowed one signing query to the challenger. The advantage is that we can construct one-time MACs with unconditional security. This is the analogue to One Time Pads for authentication.

4 Public Key: Digital Signature

Now, we discuss the public key scenario, or digital signatures. The intuition here is that "A uses a private key to sign, and B uses a public key to verify."

4.1 Algorithm overview

Like MACs, we have three relevant algorithms: a key generator, a sign generator, and a verifier. Note the differences below.

Our key generator now produces a pair containing a secret key sk and a public key pk. The tag generator produces a signature using the secret key. Our verifier verifies the message using the public key.

- Key Generation: $(sk, pk) \leftarrow \text{Gen}(1^n)$
- Signing: $\sigma \leftarrow \text{Sign}_{sk}(m)$, a signature
- Verification: $\text{Ver}_{pk}(m, \sigma)$: $M \times S \rightarrow \{0, 1\}$, where M is the message space and S the signature space

Definition 2 (Digital Signatures) *Given a key (sk, pk) and a signature σ, Digital Signatures have the following properties:*

- ***Correctness:*** *$Pr[(sk, pk) \leftarrow Gen(1^n),\ \sigma \leftarrow Sign_{sk}(m) : Ver_{pk}(m, \sigma) = 1] = 1$*
- ***Security:*** *UF-CMA as described above.*

  $\forall$ nu PPT adversaries A, $\exists$ a negligible function $\mu(\cdot)$ such that:

  $Pr[(sk, pk) \leftarrow Gen(1^n),\ (m, \sigma) \leftarrow A^{Sign_{sk}(\cdot)}(1^n) : A$ did not query $m \wedge Ver_{pk}(m, \sigma) = 1\ ] \leq \mu(n)$

5 One Time Signatures

Next, we move on into a discussion of One Time Signatures, which is a digital signature scheme where the adversary can only make one signature query. Below we discuss Lamport's one-time signature scheme based on one-way functions.

Lamport's Construction of OTS. Let f be a OWF. We make the assumption that we will only sign n-bit messages.

- Key Generation:

$$sk = \begin{bmatrix} x_1^0 & x_2^0 & \dots & x_n^0 \\ x_1^1 & x_2^1 & \dots & x_n^1 \end{bmatrix}$$

  Where $x_i^b \xleftarrow{\$} \{0,1\}^n, \forall i \in [n]$ and $b \leftarrow \{0,1\}$.

$$pk = \begin{bmatrix} y_1^0 & y_2^0 & \dots & y_n^0 \\ y_1^1 & y_2^1 & \dots & y_n^1 \end{bmatrix}$$

  Where $y_i^b = f(x_i^b)$.

- Signing: $\text{Sign}_{sk}(m) : \sigma := (x_1^{m_1}, x_2^{m_2}, \dots, x_n^{m_n})$
  Where $\mid \sigma \mid = n$. Basically, for every column of sk, select one of the two x_i^b depending on m_i.

- Verification: $\text{Ver}_{pk}(m, \sigma) : \wedge_{i \in [n]} f(\sigma_i) \overset{?}{=} y_i^{m_i}$.

Proof of Security. We will prove security w.r.t. a weaker security definition called selective security. We describe the security game below:

1. Adversary A states what message m^* it is going to forge and sends that to Challenger Ch.
2. Ch generates and sends back a public key pk.
3. A queries Ch with a message m_i.
4. Ch returns with a sign σ_i.
5. A and Ch repeat steps 3 and 4 multiple times.
6. A finally sends Ch a σ^*.

We now prove that Lamport's construction is selectively secure.

For the sake of contradiction, suppose there is a PPT adversary A_{OTS} that can break the OTS security scheme. We construct a PPT adversary A_{OWF} that uses A_{OTS} to invert a OWF.

1. A_{OWF} receives $y = f(x)$ as input. This input will be embedded into an input for A_{OTS}. This latter input will be a public key that we will generate.
2. A_{OWF} runs A_{OTS} and gets $m^* = m_1^* \dots m_n^*$ where each m_i^* is one bit.
3. Choose $i \xleftarrow{\$} [n]$, and set $b^* = m_i^*$.
4. Set $y_i^{b^*} = y$. For all $j \neq i, b \in \{0,1\}$, choose an $x_j^b \xleftarrow{\$} \{0,1\}^n$ and calculate $y_j^b = f(x_j^b$. Further, choose $x_i^{1-b^*} \xleftarrow{\$} \{0,1\}^n$ and calculate $y_i^{1-b^*} = f(x_i^{1-b^*})$.

   Now, set pk to be the matrix of all y's as computed above.

5. Send this pk to A_{OTS}.

6. Now, A_{OTS} may send a signing query m. If $m_i = m_i^*$, then halt. Otherwise, answer the signing query as $\sigma = (x_1^{m_1}, \ldots, x_n^{m_n})$.

7. A_{OTS} outputs some σ^* as its forgery. A_{OWF} will then set its $x = \sigma_i^*$ as its inversion.

Thus, if A_{OTS} generates a valid forgery with noticeable probability ϵ, then A_{OWF} can invert f with probability at least $\frac{\epsilon}{n}$, which is still noticeable. This is a contradiction. ■

Note that the adversary A_{OTS} declared the message m^* that it was going to forge at the beginning. The reduction didn't know the signing query m that A will send after that, therefore it guessed i such that $m_i \neq m_i^*$. We want this guess to be correct with noticeable probability, since otherwise, our reduction will fail almost always. If we choose i at random, then our guess will be correct with probability $\frac{1}{n}$. This is because A has no clue what i is, which is necessary - otherwise he may choose $m_i = m_i^*$ every time and thus the reduction will always fail.

So far, we have discussed authentication schemes for fixed length messages. But we continue our discussion for the case of arbitrary length messages. In order to construct signature schemes for arbitrary length messages, we first study the notion of collision-resistant hash functions (CRHFs).

6 Collision-Resistant Hash Function

A hash function is a compression function that shrinks a long message to a fixed length message. For our purposes, we need a hash function that ensures minimal collisions when shrinking messages. In particular, we want a hash function h where it is computationally hard to find two different inputs x and x' having the same outputs $h(x)$ and $h(x')$.

Definition 3 (CRHF Family) *A family $H = \{h_i : D_i \rightarrow R_i\}_{i \in I}$ is a collision resistant hash function family if:*

- *Easy to Sample: $\exists$ a PPT Gen algo s.t. $i \leftarrow Gen(1^n), i \in I$*
- *Compression: $\mid R_i \mid < \mid D_i \mid$*
- *Easy to Evaluate: $\exists$ a polytime algorithm Eval s.t. given $x \in D_i$, $i \in I$, Eval$(x, i) = h_i(x)$*
- *Collision Resistance: $\forall$ nu PPT adversary A, $\exists$ negligible function μ s.t.*

  $Pr[i \overset{\$}{\leftarrow} Gen(1^n), (x, x') \leftarrow A(1^n, i) : x \neq x' \wedge h_i(x) = h_i(x')] \leq \mu(n)$

A couple of remarks on hash functions: first, hash functions with one-bit compression can be transformed to hash functions with arbitrary compression. One such mechanism is called Merkle trees. Note, however, that the output size of the hash must be large enough to prevent easy collisions.

One-time Signatures for Long Messages. Using CRHFs, we can transform Lamport's OTS scheme into a new OTS scheme where we can sign arbitrarily long messages. To sign a message m, we first hash it using $h_i(m)$ and then use the signing algorithm in Lamport signing scheme. We rely on the collision resistance of h to argue security.

Universal One-way Hash Functions. While CRHFs are not known based on one-way functions, there is a weaker notion of hash function that can be based on one-way functions and turns out to be sufficient for constructing digital signature schemes for long messages. This notion is called a universal one-way hash function (UOWHF), and it is defined in the same manner as a CRHF, except that the collision-resistance property is described as follows:

Definition 4 (Universal One-Way Hash Function (UOWHF))

$$Pr[(x, state) \leftarrow A(1^n), i \overset{\$}{\leftarrow} Gen(1^n), x' \leftarrow A(i, state) : x \neq x' \wedge h_i(x) = h_i(x')] \leq \mu(n)$$

7 Multi-message Signatures

We end with a discussion on multi-message signatures. The definition of multi-message signatures is in the class lecture slides, as are relevant extra readings. A stateful construction of multi-message signatures can be constructed by creating a "chain" of one-time signatures (the construction is given in the class slides). Further, even stateless construction of multi-message signatures are possible, however they will not be covered in this course.