Introduction to Cryptography 03/06/2018

Lecture 15: MAC and Hash Functions

Instructor: Vipul Goyal *Scribe: Yimin Yang*

1 Review

Any encryption scheme guarantees hiding, but it does not guarantee that the adversary cannot sample a message in some intelligible way.

We have already discussed the example of one-time pad:

$$\text{Alice} \xrightarrow[Eve]{c = k \oplus s} \text{Bob}$$

Alice and Bob share a key k of a one-time pad, and Alice has two possible messages:
s_0: Sell MSFT
s_1: Buy MSFT
Alice can send $c = k \oplus s$ (s is s_0 or s_1) to Bob. And an adversary Eve is sitting on the channel. Eve can just take a ciphertext c, and compute $c' = c \oplus s_0 \oplus s_1$. Eve does not know the key that Alice and Bob share and the message Alice encrypted. But now, if $s = s_0$, c' is the encryption of s_1; if $s = s_1$, c' is the encryption of s_0.

2 Message Authentication Code(MAC)

$$\text{Alice } (k) \xrightarrow{m_1,\ \sigma_1} \text{Bob } (k)$$
$$\xleftarrow{m_2,\ \sigma_2}$$
$$\xrightarrow{m_3,\ \sigma_3}$$

For Alice and Bob who share a secret key k with each other, MAC is used to authenticate the messages that one sent to another.

2.1 Definition of MAC

Definition 1 (Message Authentication Code) *A Message Authentication Code consists of 3 PPT algorithms:*

- *Gen(n): takes security parameter n as input, outputs shared secret key k*
- *Auth(k, m): takes key k and message m as input, outputs σ*
- *Verify(k, m, σ): outputs 0/1 (reject/accept)*

2.2 Properties of MAC

- Correctness
  $\forall k \leftarrow Gen(n),\ \forall m,\ Verify(k, m, Auth(k, m)) = 1.$

- Unforgeability

  $\forall PPT\ A,\ Pr[A\ wins\ the\ forging\ game]$ is negligible.

  Forging Game:

  Challenger C and Adversary A

  1) C generates the key k

  2) A sends message m_i to C, gets back $\sigma_i = Auth(k, m_i)$

  3) Repeat 2) any polynomial times

  4) Guess: A outputs (m, σ). A wins the guessing game if $\forall i,\ m \neq m_i$ and $Verify(k, m, \sigma) = 1$

Remark 1 *We will see Digital Signature in the next class. The basic difference between Digital Signature and MAC is that MAC is not publicly verifiable. You need the same secret key to generate the MAC and verify it, but everybody can use the public key to verify the digital signature.*

2.3 Construction

A PRF on message m with the key k is also a MAC with message m with key k.

- $Gen(n)$: output $k = Gen_{PRF}(n)$
- $Auth(k, m)$: output $PRF(k, m)$
- $Verify(k, m, \sigma)$: check if $\sigma = PRF(k, m)$

Proof Sketch

Suppose there exists a PPT adversary A which can win the Guessing Game with probability $\epsilon = noticeable(n)$. It means A can guess $PRF(k, m)$ given $\{PRF(k,\ m_i)\}_{i=1}^{poly}$ s.t. $\forall m_i \neq m$.

Now replace PRF by a Random Function. We can see such A cannot exist. Because given A, we can construct an adversary B, which can win the Guessing Game of PRF with probability $\epsilon = noticeable(n)$.

■

3 Collision-Resistant Hash Functions(CRHF)

3.1 Definition of CRHF

Definition 2 (Collision-Resistant Hash Functions) *A set of functions $H = \{h_i : D_i \to R_i\} i \in I$ is a CRHF family if:*

- *Easy to sample: there exists a PPT algorithm Gen(n) which can output a uniform $i \in I$ from the family.*
- *Easy to evaluate: $\forall i,\ \forall x,\ h_i(x)$ can be computed in polynomial time.*
- *Compression: $\forall i,\ |D_i| > |R_i|$.*
- *Collision-resistance:*

  $\forall$ PPT adversary A, $Pr[i \leftarrow Gen(n), (x, x') \leftarrow A(i) : x \neq x', h_i(x) = h_i(x')] \leq negl(n)$.

3.2 Facts

- Fact 1: No single hash function can be collision-resistant.

  If you have a single hash function, such that the domain is larger than the range, there will definitely exist collisions.

- Fact 2: If H is a CRHF family, H is also a family of one-way functions, if $\forall i,\ |h_i(x)| \leq \frac{|x|}{c}$ (c can be any constant greater than 1).

  Proof idea

  Most inputs in domain D_i have colliding inputs.

  Suppose h_i is not a one-way function. There exists a PPT adversary A which can invert h_i.

  Given A, we can build a PPT algorithm B which can find collisions:

  Pick a random x from D_i, compute $y = h_i(x)$.

  Then get $x' = A(y)$.

  $Pr[x = x'] = \frac{1}{number\ of\ pre-images\ of\ h_i(x)} = \frac{1}{p}$

  $Pr[x \neq x'] = 1 - \frac{1}{p}$

  It's very likely for the inverter to give a different pre-image rather than the original input. Then we can find collisions.

■

3.3 Hash then MAC

Given a CRHF family $H = \{h_i : \{0,1\}^* \to \{0,1\}^n\} i \in I$ and MAC scheme $(Gen, Auth, Verify)$, construct the Hash then MAC scheme $(Gen', Auth', Verify')$ as follows:

- Gen': same as $Gen(n)$, also sample a hash function from H: $i \leftarrow I$. $(k,\ i)$ is the key.
- $Auth'((k,\ i),\ m)$: compute $y = h_i(m)$, then output $Auth(k,\ y) = \sigma$.
- $Verify'((k,\ i),\ m,\ \sigma)$: compute $y = h_i(m)$, $\sigma' = Auth(k,\ y)$. Check if $\sigma' = \sigma$.

This scheme can handle arbitrary length of messages.

Proof of Security for $m \in \{0,1\}^*$:
Adversary is given $\{m_j\}_j$, then it can compute $\{h_i(m_j)\}_j$, $\{Auth(h_i(m_j),\ k)\}_j$.
Adversary outputs m, $h_i(m)$, $Auth(h_i(m),\ k)$.
If the adversary wins the forging game, there are two cases:

- case 1:

  $\forall j,\ h_i(m) \neq h_i(m_j)$

  => The adversary has to forge the underlying MAC scheme.

- case 2:

  $\exists j$, s.t. $h_i(m) = h_i(m_j)$, but $m \neq m_j$

  => collision in h_i

To forge the Hash then MAC scheme, the adversary should either forge the underlying MAC scheme, or find collisions in the CRHF family H.

■

3.4 Construction Based on Discrete Logarithm Assumption

Assumption 1 (Discrete Logarithm Assumption) *Given G_q which is a multiplicative group of prime order q and a generator $g \in G_q$, then for every PPT A,*

$Pr[x \xleftarrow{\$} Z_q : A(g^x) = x] \leq negl(\log q)$

Construction:

Family H is parameterized by group G of prime order q, and a generator g in G.

- $Gen(n)$: sample h from G at random, where $h = g^r$. $i = h$.
- $Evaluate(h, x)$: parse x as (x_1, x_2). x_1, x_2 must be in Z_q. Then output $g^{x_1}h^{x_2}$.

  The hash function can be evaluated in polynomial time.
- Compressing:

  $g^{x_1}h^{x_2} \in G$, so the size of output is $\log q$.

  $|x| = 2|x_1|,\ x_1 \in Z_q$

  $|x| = 2\log q$, so the size of input is $2\log q$.

  So the input is almost double of the size of the output.
- Collision-Resistance:

Proof.

Suppose there is a PPT adversary A. Given $g^{x_1}h^{x_2}$, A can output x, x' s.t. $x \neq x'$.
We can construct a PPT adversary B to compute discrete log of h. B works as follows:
B gets from A:

$x = (x_1,\ x_2)$
$x' = (x'_1,\ x'_2)$
s.t. $g^{x_1}h^{x_2} = g^{x'_1}h^{x'_2}$

This means $g^{x_1+r\cdot x_2} = g^{x'_1+r\cdot x'_2}$
$x_1 + r \cdot x_2 = x'_1 + r \cdot x'_2$
If $x \neq x'$, $x_2 \neq x'_2$
Then $r = \dfrac{x'_1 - x_1}{x_2 - x'_2}$
Now we can get the discrete log of h. This is a contradiction.

■

Remark 2 *We can extend this construction to get an arbitrarily compressing hash function. The idea is:*

- *Gen(n): sample* $h_1, h_2, ..., h_m \in G$ *s.t.* $h_j = g^{r_j}$
  $i = (h_1,\ h_2,\ ..,\ h_m)$
- $Eval(i, x) : x = (x_0,\ x_1,\ ...,\ x_m)$
  output $g^{x_0} h_1^{x_1} h_2^{x_2} ... h_m^{x_m} \in G$

The input is very long, but the output is an element in group G.

Proof: exercise

Hint: you should be able to find the discrete log of at least one h_i*, thus reach a contradiction.*