Cryptography Part 1 **Fall 2017**

Lecture 3: Integrity

Lecturer: Dan Boneh *Scribe: Lily Li*

3.1 Message Authentication Code (MAC)

We use message authentication codes to ensure integrity (but not confidentiality). Two parties, Alice and Bob have a shared key k. When Alice sends message m to Bob she uses a MAC signing algorithm S and generates a short tag $S(k, m)$. She send m along with the tag to Bob. When Bob receives the message and tag he runs the MAC verification algorithm $V(k, m, tag)$ which outputs *yes* or *no* depending on if the message-tag pair is valid or not. More succinctly:

Definition 3.1 *MAC $I = (S, V)$ defined over (K, M, T) is a pair of algorithms such that $S(k, m)$ outputs $t \in T$, $V(k, m, t) \in \{yes, no\}$, and $\forall k \in K \forall m \in M : V(k, m, S(k, m)) = yes$.*

Note: the shared key *is* required for integrity. If Alice only uses a cyclic redundancy check (CRC - used to detect random errors) to generate a tag, an attacker could generate a tag for their own message which fools the CRC.

Lets define security for MACs. Here the attacker can mount a chose message attach. That is: for messages $m_1, m_2, ..., m_q$ the attacker can obtain $t_i \leftarrow S(k, m_i)$. The attacker's goal is to create an existential forgery, that is: a new valid message-tag pair $(m, t) \notin \{(m_1, t_1), ..., (m_q, t_q)\}$. If the attacker cannot produce a valid tag for a new message or a new valid tag for a known message, then the MAC is secure.

Definition 3.2 *For MAC $I = (S, V)$ and adversary A the MAC game is defined between the challenger and adversary as follows. The challenger chooses a random key $k \in K$. The adversary sends the challenger q messages $m_1, ..., m_q$ and gets in return valid tags $t_1, ..., t_q$ associated with each message. Then, the adversary generate a new message-tag pair (m, t) and sends this to the challenger. If*

$$\mathsf{Adv}_{MAC}[A, I] = \mathsf{Pr}[(m, t) \text{ is valid}]$$

is negligible then the MAC is secure.

Note: if the tag is too short, then it is possible for the adversary to guess the tag with non-negligible probability.

3.2 Constructing Secure MACs

There is actually a simple MAC built from a construct that we have seen previously, namely the PRF.

Given PRF $F : K \times X \rightarrow Y$ a derived MAC $I_F = (S, V)$ is simply $S(k, m) = F(k, m)$ and $V(k, m, t)$ outputs *yes* if and only if$F(k, m) = t$.

Theorem 3.3 *If $F : K \times X \to Y$ is a secure PRF and $1/|Y|$ is negligible then I_F is a secure MAC. That is: for every effective MAC adversary A attacking I_F there exists an efficient PRF adversary B attacking F:*

$$\mathsf{Adv}_{MAC}[A, I_F] \leq \mathsf{Adv}_{PRF}[B, F] + \frac{1}{|Y|}$$

Thus I_F is secure if $|Y|$ is long enough.

So we see that it is quite advantageous to be able to create a Big-PRF from a Small-PRF. Two main constructions are: CBC-MAC and HMAC.

Side Note: truncating a secure PRF produces another secure PRF on fewer input bits.

3.2.1 CBC-MAC

Recall our PRF (AES) can only encrypt short message (16-byte) so what we need is a way to encode arbitrary long messages. Enter, encrypted CBC-MAC. Let $F : K \times X \to X$ be a PRP. We define a new PRF $F_{ECBC} : K^2 \times X^{\leq L} \to X$, which mean that there are at most L blocks for some large L. The construction is as follows: break the message into blocks the same length as elements of X. Let these be $m[0], m[1], ..., m[L-1]$. Then the tag is

$$F(k_1, F(k, m[L-1] \oplus F(k, m[L-1] \oplus \cdots F(k, m[0]) \oplus m[1]) \cdots)$$

(drawing a diagram helps).

Note: this is quite similar to CBC except that we do not output intermediate values and we encrypt the last output using an independent key k_1. This construction without the final encryption is call: **raw-CBC** and it is *not* a secure MAC.

Consider the following attack: the adversary asks for a valid tag t associated with a one block message m. Then the two block message $(m, t \oplus m)$ has t as a valid tag. Observe why this is the case. The first step of raw-CBC is to calculate $F(k, m)$ and XOR it with $m \oplus t$. However, $F(k, m) = t$ so $F(k, m) \oplus m \oplus t = m$. Thus the output is again $F(k, m)$ for which t is a valid tag.

3.2.2 NMAC

To build an NMAC we need a PRF $F : K \times X \to K$ (note that the range is the key space and not the input space X as before). The new PRF $F_{NMAC} : K^2 \times X^{\leq L} \to K$. Again the message m is broken up into chunks $m[0], m[1], ..., m[L-1]$. The tag is

$$F(k_1, F(F(\cdots F(k, m[0]), m[1]) \cdots m[L-1]) \parallel fpad)$$

where $fpad$ is the padding needed to get a string in the key space to be the same length as a string in the input space (this is used as the second input to F).

The construction without the last application of F is called a cascade and it should be easy to see why it is not secure. Observe, if we obtain the message key pair (m, t) we can also obtain the tag for any extension $m \parallel w$ for some message block w since we can just run $F(t, w)$ to get a tag t' for $m \parallel w$.

The following is the security bounds on ECBC and NMAC:

Theorem 3.4 *For every efficent q-query PRF adversary A attacking F_{ECBC} or F_{NMAC}, there exists an efficient adversary B such that*

$$\mathsf{Adv}_{PRF}[A, F_{ECBC}] \leq \mathsf{Adv}_{PRP}[B, F] + \frac{2q^2}{|X|}$$

$$\mathsf{Adv}_{PRF}[A, F_{NMAC}] \leq qL\mathsf{Adv}_{PRF}[B, F] + \frac{q^2}{2|K|}$$

Notice that in analysis of ECBC includes a PRP adversary. As always, $L > 0$ is the length of the blocks and q is the number of blocks encrypted with the same key.

Here is the type of attack that would take place if you do not change the keys often enough. Suppose that the underlying PRF F for both ECBC and NMAC are actually PRPs. Then ECBC and NMAC has the extension property:

$$\forall x, y, w : F_{BIG}(k, x) = F_{BIG}(k, y) \Rightarrow F_{BIG}(k, x \parallel w) = F_{BIG}(k, y \parallel w).$$

The attack asks for many (say square-root of the size of the message space) message key pairs (m_i, t_i) and waits for a collision $t_u = t_v$ for two distinct messages $m_u \neq m_v v$ (this will happen with high probability due to the birthday paradox). Then the attacker will query message $m_u \parallel w$ for some block w. Suppose t is a valid tag for $m_u \parallel w$, then t is also a valid tag for $m_v \parallel w$.

3.2.3 MAC Padding

Suppose that the message is not a multiple of the block size. Then we need to pad the message in-order to encode it properly. One obvious but *insecure* way to do this is to pad by a string of zeros. This is insecure since it gives the attacker information about $m \parallel \mathbf{0}$.

Thus the padding function $pad(m)$ must be invertible. The solution for ISO is to append $100\cdots0$ to fill up the last block. If the message is already a multiple of the block size then you would need to append a dummy block. Notice to invert this "append the $10\cdots0$ until it fills up the block" construction all we need to do is scan, starting from the right, to the first one then remove all previous zeros and this first one.

3.2.3.1 CMAC (NIST Standard)

All deterministic padding strategies require this dummy block. However CMAC is one randomized padding strategy which does not. CMAC is a variant of CBC-MAC which takes a triple of keys (k, k_1, k_2) (usually k_1 and k_2 are derived from k using some PRG). Then CMAC runs CBC-MAC until it gets to the last block, say $m[w]$. If $m[w]$ is shorter than the block length, the CMAC pads with a string of $10\cdots0$ as before and XORs with k_1 and the output from the previous blocks to produce the tag. If instead $m[w]$ is exactly the block size then CMAC XORs $m[w]$ with k_2 and the output from the previous blocks.

3.2.4 PMAC

Parallel MAC! Here the message is again broken up into blocks $m[0], m[1], ..., m[L]$. Then each message block i is XORed with $P(k, i)$ for a function P. The result, $F(k_1, m[i] \oplus P(k, i))$ for $0 \leq i \leq L-2$ (notice not $L-1$) are all XORed together then encrypted with the key k_1 one more time. The padding is similar as the one for CMAC.

The security analysis for PMAC is:

Theorem 3.5 *For q which is the number messages map with the same key and L which is the length of all the message. If F is a secure PRF over (K, X, X) then F_{PMAC} is a secure PRF over $(K, X^{\leq L}, X)$.*

For every efficient q-query PRF adversary A attacking F_{PMAC} there exists an efficient PRF adversary B such that

$$Adv_{PRF}[A, F_{PMAC}] \leq \mathsf{Adv}_{PRF}[B, F] + \frac{2q^2L^2}{|X|}$$

PMAC is secure as long as $qL \ll |X|^{1/2}$.

Observe that PMAC has the property that you can reuse your tag computation if two messages are very similar.

3.2.5 One-time MAC

These are analogs of the One-time pad used for encryption. These are much faster than the previous MACs and use each key to encrypt one message.

The security game this time sees the challenger pick a random key $k \in K$. The adversary send the challenger one message $m_1 \in M$ the the challenger returns the associated tag $S(k, m_1)$. Then the adversary attempts to forge a new message tag pair (m, t). The challenger outputs $b = 1$ if and only if (m, t) is valid and $(m, t) \neq (m_1, t_1)$.

If you have a one-time MAC it is possible to modify this into a many-time MAC. In particular let (S, V) be a secure one-time MAC over $(K_I, M, \{0,1\}^n)$ and $F : F : K_F \times \{0,1\}^n \to \{0,1\}^n$ be a secure PRF.

The **Carter-Wegman MAC** is defined as: $CW((k_1, k_2), m) = (r, F(k_1, r) \oplus S(k_2, m)$ for some random string $r \leftarrow \{0,1\}^n$. This is desirable since the fast one-time MAC is used to compute the tag for the long message and the slow PRF is used on the short random string r.

3.3 Collision Resistance

Definition 3.6 *Let $H : M \to T$ be a has function ($|M| \gg |T|$). A **collision** for H is a pair $m_0, m_1 \in M$ such that $H(m_0) = H(m_1)$ but $m_0 \neq m_1$.*

*A function H is **collision resistant** if for all* explicit *efficient algorithms A:*

$$\mathsf{Adv}_{CR}[A, H] = \mathsf{Pr}[A \text{ outputs collisions for } H$$

is negligible.

An example of a collision resistant function is SHA-256 (note the previous version SHA-1 has a function which generates collisions).

Now we can use the this hashing idea to turn a MAC for short messages into one for big messages. Suppose $I = (S, V)$ is a MAC for short messages over (K, M, T) and let $H : M^{big} \to M$ be a hash function.

Definition 3.7 *$I^{big} = (S^{big}, B^{big})$ over (K, M^{big}, T) as*

$$S^{big}(k, m) = S(k, H(m)); \qquad V^{big}(k, m, t) = V(k, H(m), t)$$

Theorem 3.8 *If I is a secure MAC and H is collision resistant, then I^{big} is a secure MAC.*

Consider why collision resistance is necessary here. Suppose that the adversary can find two distinct messages m_0, m_1 which hash to the same value $H(m_0) = H(m_1)$. Then they can ask for the tag for one message and this tag would also be valid for the other message.

3.3.1 Generic Attack on CR Function

This attack is based on the birthday paradox, stated as follows:

Theorem 3.9 *Let $r_1, ..., r_n \in \{1, ..., B\}$ be independently identically distributed integers.*

When $n = 1.2 \times B^{\frac{1}{2}}$:

$$\mathsf{Pr}[\exists i \neq j : r_i = r_j] \geq \frac{1}{2}$$

Proof: Consider the probability of a collision on each of the n choices. On the first try no collisions are possible. On the second try $B-1$ choices are safe — do not cause any collisions. On the third try $B-2$ choices are safe and so on. Thus the probability that there exists a collision is

$$\begin{aligned} 1 - \left(\frac{B-1}{B})\right) \cdot \left(\frac{B-2}{B}\right) \cdots \left(\frac{B-n+1}{B}\right) &= 1 - \prod_{i=1}^{n-1} \left(1 - \frac{i}{B}\right) \\ &\geq 1 - e^{-\frac{1}{B}\left(\sum_{i=1}^{n-1} i\right)} \\ &\geq 1 - e^{-\frac{n^2}{2B}} \end{aligned}$$

If you plug in $n = 1.2 \cdot B^{\frac{1}{2}}$ into the last expression, you get something that is pretty close to 0.5. ■

Now that we understand the paradox we can use it to build the attack. Let $H : M \to \{0,1\}^n$ be a has function ($|M| \gg 2^n$). The following algorithm finds collisions in $O(2^{n/2})$ expected hashes.

1. Choose $2^{n/2}$ random messages in M: $m_1, ..., m_{2^{n/2}}$ (these are distinct with high probability).
2. For $i = 1, ..., 2^{n/2}$ compute $t_i = H(m_i) \in \{0,1\}$.
3. Look for a collision ($t_i = t_j$). If none found go back to step 1.

By the birthday paradox, the expected number of times we have to execute step one is two, so we expect $O(2^{n/2})$ hashes to be performed.

3.3.2 Building Collision Resistant Hash Functions (Merkle-Damgard Paradigm)

Here we are going to show a method for turning a collision resistant CR function for short messages into CR functions for long messages.

Suppose $h : T \times X \to T$ is our CR function for short messages (sometimes called the compression function). We want to build $H : X^{\leq L} \to T$. As always, we break our message into chunks $m[0], m[1], ..., m[i]$ if the last block is shorter than the block length then we pad it with our usual string of $10 \cdots 0$, denoted PB. We first input a fixed IV as the tag and $m[0]$ into h. This produces H_1, the first chaining variable. Then with H_1 as the tag and $m[1]$ as the message, apply h to get chaining variable H_2. Continue in this way until we obtain chaining variable H_i. Let $H_i = H$.

Theorem 3.10 *If h is a CR then so is H.*

Proof: By contraposition. Suppose $H(M) = H(M')$. Let $IV = H_0, H_1, ..., H_t, H_{t+1} = H(M)$ and $IV = H'_0 = H'_1 = \cdots = H'_r = H'_{r+1} = H(M')$. be the chaining variable associated with the hashing of messages M and M' respectively. Observe that since $H(M) = H(M')$, it must be case that $h(H_t, M_t \parallel PB) = h(H'_r, M'_r \parallel PB')$. Since h is a CR function the inputs must be identical. In particular $r = t$ and $M'_r = M_t$. The same argument can be repeated to show that $M'_{t-1} = M_{t-1}$, $M'_{t-2} = M_{t-2}$, ..., $M'_0 = M_0$. But this means that $M = M'$. ■

3.3.3 Building Compression Functions

One way to build a compression function is by using a block cipher. Suppose $E : K \times \{0,1\}^n \to \{0,1\}^n$ is a block cipher. The **Davies-Meyer** compression function: $h(H, m) = E(m, H) \oplus H$.

Theorem 3.11 *Suppose E is an ideal cipher (collection of $|K|$ random permutations). Finding a collision $h(H, m) = h(H', m')$ takes $O(2^{n/2})$ evaluations of (E, D).*

Note: it is essential to do the final $\oplus$ otherwise the pair (H, m) and (H', m') where $H' = D(m', E(m, H))$ is a collision.

Another class of compression functions are the provable compression functions. We choose a random 2000-bit prime p and random $1 \leq u, v \leq p$. for $m, h \in \{0, ..., p-1\}$ define $h(H, m) = u^H \cdot v^m (\mod p)$.

NOTE: if you can find a collision for h then you can solve the *discrete-log* problem modulo p.

3.3.4 HMAC

This is a standard method for converting a collision resistant function (CR) into a MAC.

HMAC does the following:

$$S(k, m) = H\left(k \oplus opad, H\left(k \oplus ipad \parallel m\right)\right)$$

Note that *opad* and *ipad* are both public.

3.3.5 Timing Attacks on MAC Verification

If you check the tag byte by byte and rejecting when a difference is detected you are vulnerable to a timing attack. By slightly modifying the tag guesses the attacker can cause the checker to leak information about the number of correct block. Tags which have a large satisfying prefix will take longer to reject than if the first block of the tag differs.

Possible defenses force the computer to take the same time to compare the tags or obfuscate the tags being compared.