University of Illinois, Urbana Champaign
CS/ECE 498AC3/4 Applied Cryptography

Instructor: Dakshita Khurana
Scribe: Nishant Kumar
Date: September 10, 2020

LECTURE

6

MACs and Collision-Resistance

We discussed at the beginning of the lecture that HW1 is out and is due in one week. Everyone is asked to start early, collaborate and make abundant use of office hours.

To recap, in the last lecture, we started looking at Message Authentication Codes (MACs). These are cryptographic primitives which allow one party, say Alice, to communicate with another party, say Bob, a message and a tag. The security guarantee provided by MACs allows Bob to be rest assured that the message is indeed coming from Alice and has not been tampered. In this lecture we will continue our discussion of MACs and see how to build it for messages of unbounded length from block ciphers and collision resistant hash functions. Thereafter, we see how to build these collision resistant hash functions themselves.

6.1 Message Authentication Codes (MACs) - Recap

Recall from previous lecture, that MACs are made up of two algorithms - a signing algorithm $S(k, m)$, which takes as input a key k, message m and outputs a tag t, and a verification algorithm $V(k, m, t)$, which takes as input the key k, message m, tag t and outputs a bit $0/1$, indicating whether the tag was accepted or rejected.

For correctness to hold, we required $V(k, m, t) = 1$, when $t \leftarrow S(k, m)$. For security, intuitively, we wanted that the adversary should not be able to produce a new valid (message, tag) pair even after seeing polynomially many valid (message, tag) pairs of messages of his choice. This is formalized by a security game wherein first, the key k is sampled uniformly at random, second, the adversary provides messages of his choice m_i and receives corresponding tages $t_i \leftarrow S(k, m_i)$. After seeing polynomially many such (m_i, t_i), it outputs a (m', t') and wins the game if $(m', t') \notin \{(m_i, t_i)\}_i \wedge V(k, m', t') = 1$.

We had also seen in the last lecture that a PRF can be used as a MAC. But the problem with this construction is that since PRFs have some predetermined input size, we can't produce MACs for messages of unbounded length with this construction. In this lecture, we will see two ways this can be overcome. In particular, we will discuss two ways to construct MACs - CBC-MAC and HMAC.

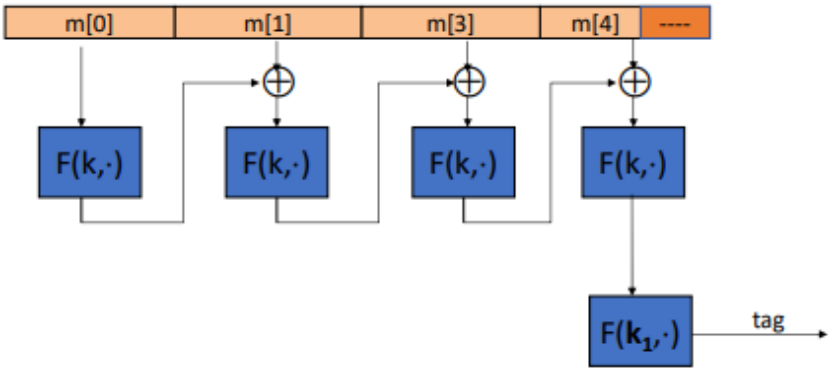

FIGURE 6.1: Encrypted CBC-MAC

6.2 Encrypted CBC-MAC

This construction of generating MACs is similar to the CBC mode of encryption we had seen in the previous lectures. It uses a PRF function F. The PRF function can be instantiated with a block cipher, which as we have seen earlier, is assumed to work like a Pseudo-random permutation (PRP) and hence a PRF. But, as will see, we won't be requiring the invertability of the PRP and only the pseudorandomness of it - which is why the weaker cryptographic primitive of PRF also suffices.

Next, we formally describe the construction. Assume that the PRF F has input and output domain of size b bits, i.e. the block length is b. The construction uses two PRF keys k, k_1. For simplicity, for the time being, assume that the input message m can be equally partitioned in blocks of size b bits, i.e. $m = m[0]||m[1]||\dots||m[k]$, where $\forall i \in \{0, 1, \dots k\}, |m[i]| = b$. Then the MAC is produced by calculating $t_0 = F(k, m[0])$, $t_i = F(k, m[i] \oplus t_{i-1}) \forall i \in \{1, 2, \dots k\}$. The final tag is output as $F(k_1, t_k)$.

In case the last message block is not of size b, we pad the same to generate a message of size b bits. But we need to be careful on how we do this padding. A first attempt might be simply pad the message with zeros. But this leads to problems because the padded message for the following messages (and hence the generated tags) are identical : 0^b, $1||0^{b-1}$. A second attempt to pad might be to append first a 1 and then the remaining positions with zeros. For example, a message $m = 010$ is padded to produce message $m' = 010||1||0^{b-4}$. This way the padding leads to a unique padded message. Indeed, the problem with padding with all zeros is that two different messages can lead to the same padded message, which is why we need to make sure to define our padding in a way that no two different messages pad to the same message. Our second attempt of padding with first a 1 and then the all zeros string leads to a unique way of padding because given any padded message, we can look from the end till we hit a 1. The unpadded message is then the remaining string after removing that 1.

A figurative description of the construction is given in Figure 6.1.

We also make some interesting observations on if and why the last PRF $F(k_1, \cdot)$ is required in the construction. Indeed, we show how to mount an explicit forgery attack if this PRF is omitted. Consider the message of a single block length $m[0]$ and its corresponding tag

$t_0 \leftarrow F(k, m[0])$ (in the modified construction without the last PRF). Given this valid message and tag pair, the adversary can output the following $m' = m[0]||m[1] = m[0]||(m[0] \oplus t_0), t' = t_0$. Indeed t' is a valid tag for message m' because the final tag in the modified construction would be output as $F(k, m[1] \oplus t_0) = F(k, m[0] \oplus t_0 \oplus t_0) = F(k, m[0]) = t$. We also note that for similar reasons, the final PRF usage should be with a different key k_1 than used before in the construction.

6.3 Hash-then-MAC

The idea of hash-then-mac is simply to hash any given message of arbitrary length down to some fixed size, thereafter which we can use the PRF construction for MACs which we had seen in the last lecture. As we will see next, while the CBC-MAC we saw in the last section, leads to a serialized way of generating the MAC, there is a modified construction of what we discuss in today's lecture which lends itself to a fast parallel execution. In fact, the parallel version of the serialized construction we will see in this lecture would be covered in HW1.

But given this idea of hashing and then generating the MAC, we ask what properties we need for the hash function H to satisfy. Its easy to see that we need what is called as collision-resistance, meaning the adversary shouldn't be able to find two different messages (m_0, m_1) which hash to the same thing. If indeed it could, then in our MAC security game, it could ask a tag t for m_0 and then simply output $m' = m_1, t' = t$. This would constitute a valid message, tag pair since after hashing both m_0 and m_1 produce the same thing and hence t constitutes a valid tag for both the messages.

Lets define the above intutitive idea formally.

Definition 6.1. Let the hash function be $H : \mathcal{M} \rightarrow \mathcal{T}$, from some message space $\mathcal{M}$ to target space $\mathcal{T}$ with $|\mathcal{M}| \gg |\mathcal{T}|$. We define a collision for H as a pair (m_0, m_1) s.t. $H(m_0) = H(m_1) \wedge m_0 \neq m_1$. The hash function H is called collision-resistant, if $\forall$ PPT algorithms $\mathcal{A}$, $Pr[\mathcal{A}(1^k)$ outputs a collision for $H] = \text{negl}(k)$.

An example of a heuristic collision-resistant hash function (henceforth referred as CRHF) is SHA-256. We then have the following theorem.

Theorem 6.2. *Let (S, V) be a MAC for short messages with key-space $\mathcal{K}$, message-space $\mathcal{M}$ and tag-space $\mathcal{T}$. Let $H : M^{big} \rightarrow \mathcal{M}$ be a hash function. Define a new MAC (S^{big}, V^{big}) for long messages with key-space $\mathcal{K}$, message-space M^{big} and tag-space $\mathcal{T}$ as $S^{big}(k, m) = S(k, H(m)), V^{big}(k, m, t) = V(k, H(m), t)$. Then assuming (S, V) is a secure MAC and H is a CRHF, we have that (S^{big}, V^{big}) is a secure MAC over message space M^{big}.*

Hence, given a CRHF H, its easy to build a MAC for messages of arbitrary size. We will next look into more details at these CRHFs.

6.4 Collision Resistant Hash Functions (CRHF)

CRHFs are used widely for ensuring file integrity in software packages. A secure read-only public space can contain the hashes of the files to be downloaded. Users go on to download their desired files and can then calculate the hash of the file themselves and check it against

the hash posted on the public space. Spoofing the file downloaded by the user with a different one by an adversary amounts to finding a collision in the underlying hash function, which by definition of CRHFs is hard.

We next give some insight on attacks that can be mounted on CRHFs. Note that since the hash function is by definition compressing (i.e. its a many-to-one mapping), there do exist collisions. The security guarantee of CRHFs says that finding such collisions by a PPT adversary should be hard. One might expect that the security achieved by CRHFs is then around 2^n for a input domain of size 2^n, i.e. an adversary can't do better than checking each of the input values one-by-one. But this is in fact incorrect. A trivial attack, called the birthday attack and which we show next, can allow the adversary to find collisions with probability more than 0.5 in time $2^{\frac{n}{2}}$.

6.4.1 Birthday Attack

The birthday attack is a generic attack strategy by an adversary which allows it to find a collision for any CRHF in time and space $O(2^{\frac{n}{2}})$. In particular, given a CRHF H, the adversary follows the following attack strategy:

1. Choose $2^{\frac{n}{2}}$ messages at random from the message space. Call these $m_1, m_2 \dots m_{2^{\frac{n}{2}}}$.
2. $\forall i \in \{1, 2 \dots 2^{\frac{n}{2}}\}$, compute $t_i = H(m_i)$.
3. Check if $\exists i, j$ s.t. $i \neq j \wedge t_i = t_j$. If yes, output it, else repeat from step 1.

Analysis To analyze how well this attack strategy works, we look at the generlized birthday problem and find the probability of collision. In particular, we have the following theorem.

THEOREM 6.3. *Suppose n items $r_1, r_2 \dots r_n$ are sampled independently and identically at random from the set $\{1, 2 \dots B\}$. When $n = 1.2 \times B^{\frac{1}{2}}$, we have $Pr[\exists i, j \in \{1, 2, \dots n\}$ s.t. $i \neq j \wedge r_i = r_j] \geq 0.5$.*

Proof.

$$
\begin{aligned}
Pr[\exists i, j \in \{1, 2, \dots n\} \text{ s.t. } i \neq j \wedge r_i = r_j] &= 1 - Pr[\forall i, j \in \{1, 2, \dots n\}, r_i \neq r_j] \\
&= 1 - \left(\frac{B-1}{B}\right)\left(\frac{B-2}{B}\right)\dots\left(\frac{B-(n-1)}{B}\right) \\
&= 1 - \left(1 - \frac{1}{B}\right)\left(1 - \frac{2}{B}\right)\dots\left(1 - \frac{n-1}{B}\right) \\
&= 1 - \prod_{i=1}^{i=n-1}\left(1 - \frac{i}{B}\right) \\
&\geq 1 - \prod_{i=1}^{i=n-1} e^{\frac{-i}{B}} \quad \left(\text{using } \left(1 - \frac{i}{B}\right) \leq e^{\frac{-i}{B}}\right) \\
&\geq 1 - e^{\frac{-n(n-1)}{2B}}
\end{aligned}
$$

With $n = 1.2 \times B^{\frac{1}{2}}$, we have $Pr[\exists i, j \in \{1, 2, \dots n\}$ s.t. $i \neq j \wedge r_i = r_j] \geq 0.53 \geq 0.5$. □

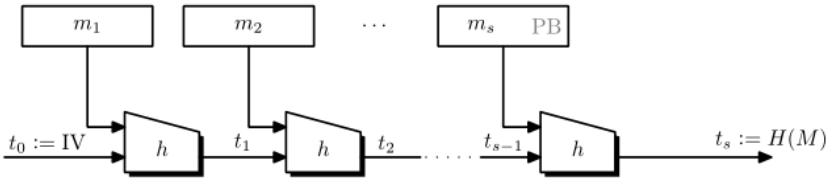

FIGURE 6.2: The Merkel-Damgard construction of CRHF

Hence, we see by the above theorem, we see that one iteration of the adversary has 0.5 probability of finding a collision for the hash function. The expected number of such iterations is then 2.

To conclude, the birthday attack is a generic attack strategy that an adversary can follow to find collisions for a hash function. This takes time and space $O(2^{\frac{n}{2}})$. Concretely, for SHA1, since the output bit size is 160 bits, this attack can find collisions in time around 2^{80}. Note that pragmatically such attacks might be prohitive to actually carry out. Interestingly, the best attacks which have been sucessfully run on SHA1 run in time 2^{51}.

6.4.2 Constructing CRHFs

In this section we discuss how to construct CRHFs of variable domain size, given a CRHF of a fixed domain size. The construction we will see in more detail here is called the Merkle-Damgard construction, and it ensures the hash function constructed is collision resistant given a hash function of a fixed domain size which is collision resistant. Thereafter, we will see how to instantiate this fixed-domain CRHF which is used in the construction using a block cipher. In particular, we will be seeing the Devies-Meyer construction.

Merkle-Damgard construction of CRHF The Merkle-Damgard construction is used to build a CRHF $H : \mathcal{X}^{\leq L} \to \mathcal{T}$, using a one-way compression function $h : \mathcal{T} \times \mathcal{X} \to \mathcal{T}$. The construction is described in Figure 6.2. Let the block length $= |\mathcal{X}| = b$. The construction uses a fixed intialization value IV and a padding block PB at the end of the last message block. The padding block will have the following form : $PB = 1000\ldots0||\langle s \rangle$, where $\langle s \rangle$ encodes in binary the number of b length blocks in the message. If the input message is such that there is not enough space for the padding block in the end, then an additional block is added at the end of the message for the padding block. The fact that the padding block contains an encoding of the number of b length blocks of the message will be used in our security proof, as we show below. Given this, we formally define the construction as follows.

Assume after applying the padding block the message is given by : $m = m[1]||m[2]||\ldots m[s]$. Let $t_0 = IV$. Then $\forall i \in \{1, 2 \ldots s\}, t_i = h(t_{i-1}, m[i])$. Output the final value t_s.

We then have the following theorem.

THEOREM 6.4. *If $h : \mathcal{T} \times \mathcal{X} \to \mathcal{T}$ is a CRHF, then $H : \mathcal{X}^{\leq L} \to \mathcal{T}$ given by the Merkle-Damgard construction is also a CRHF.*

Proof. Suppose for the sake of contradiction that H is not a CRHF. Then $\exists$ a PPT adversary $\mathcal{A}$ which can find a collision for H - say $(m, m') = (m[1]||m[2]\ldots m[s], m'[1]||m'[2]||\ldots m'[v])$.

Here we are assuming the last blocks in each of m, m' contain the padding block PB. Also, let the values output from h be denoted by variables t_i and t'_i for m and m' respectively (as shown in Figure 6.2). In particular, $t_0 = t'_0 = IV$, $i \neq 0 \wedge i \in \{1, 2 \dots s\}, t_i = h(m[i]||t_{i-1})$ and $i \neq 0 \wedge i \in \{1, 2 \dots v\}, t_i = h(m'[i]||t'_{i-1})$.

Now, we start analyzing the construction backwards from the last block. In particular, since $H(m) = h(m[s]||t_{s-1})$ and $H(m') = h(m'[v]||t_{v-1})$. $H(m) = H(m') \iff h(m[s]||t_{s-1}) = h(m'[v]||t_{v-1})$. But since h is a CRHF, either we have just found a collision for h or the values are in fact equal. If former, we are done, since we already have a collision for h. If the latter, this means $m[s] = m'[v]$ and $t_{s-1} = t'_{v-1}$. But since each of $m[s]$ and $m'[v]$ contains the padding block, this means $s = v$.

We next iterate inductively backwards one block at a time. $\forall i \in \{1, 2 \dots s-1\}, t_i = t'_i \iff h(m[i]||t_{i-1}) = h(m'[i]||t'_{i-1})$. But this means either we have found a collision for h or the values are equal. If the former, we are done. Else, we recurse.

At the end of the above inductive argument, we get that $\forall i \in \{1, 2 \dots s\}, m[i] = m'[i]$. We had already shown that in fact length of second message $v = s$, the length of the first message.

Hence, by the above argument, either we break the collision resistance of h or prove that the messages m and m' are in fact equal. □

Devies-Meyer Construction of h Through the Merkle-Damgard construction, we see how to construct a CRHF H on messages of unbounded size, assuming a CRHF h which works on a fixed domain. We next turn to the question of how to construt such fixed domain hash function h using block ciphers. In particular, we will take a look at the Devies-Meyer construction.

The Devies-Meyer compression function uses a block cipher $E : \mathcal{K} \times \mathcal{X} \to \mathcal{X}$, where $\mathcal{X} = \{0,1\}^n$. The function $h : \mathcal{X} \times \mathcal{K} \to \mathcal{X}$ is given by

$$h(t, m) = E(m, t) \oplus t$$

Figure 6.3 shows the same construction. Note that the input message m of the function h is fed in as the key of the block cipher. This goes against our intuition of using the message as a key in cryptographic constructions. But as we show below, the function h indeed results in a CRHF assuming E is an ideal cipher. Recall in the ideal cipher model is stronger than assuming that the block cipher is a PRP. While PRP assumption requires the key of the block cipher to be chosen randomly, the ideal cipher assumption allows us to assume that E is a PRP for every key. We then have the following theorem.

THEOREM 6.5. *Assuming $E : \mathcal{K} \times \mathcal{X} \to \mathcal{X}$ is an ideal cipher, the function $h : \mathcal{X} \times \mathcal{K} \to \mathcal{X}$ is CRHF assuming $|\mathcal{X}|$ is large.*

We refer the reader to [1] for more details on the proof of this theorem. We also note that if instead of defining the function h as above, we defined it as $h(t, m) = E(m, t)$ (essentially removing the xor operation at the end), then the function is not collision resistant. In fact, its pretty easy to mount an explicit attack in this case. Note that $h(t', m') = h(t, m) \iff E(m', t') = E(m, t)$. Then the adversary can choose m, t, m' as per his choice and find t' satisfying the above equation as $t' = D(m', E(m, t))$.

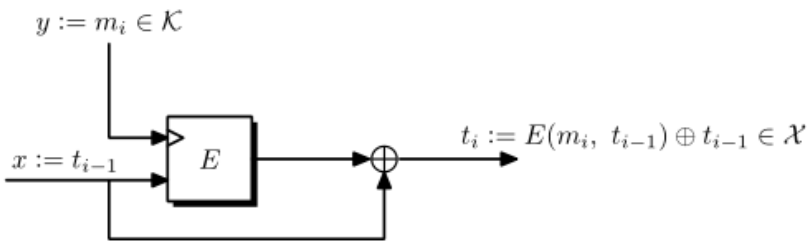

FIGURE 6.3: The Devies-Meyer compression function

6.5 HMAC

While we saw in Section 6.3 how to use a CRHF (like SHA256) and a PRF (like AES) to build a MAC scheme, and subsequently in Section 6.4 how to build the CRHF (like SHA) itself using a block cipher, we ask here if we can get a simpler construction of a MAC scheme for messages of arbitrary sizes using just a CRHF like SHA256. The reason for doing this is the construction given in Section 6.3 lends itself to offline attacks, where the adversary can keep searching for collisions in the underlying hash functions offline. Another reason for trying to build a MAC using a CRHF constructed in Section 6.4, is that the practical instantations of a MAC scheme based on Section 6.3 might end up using both software (for SHA) and hardware (for AES). Ideally it would be nice if we can get a MAC scheme based only on one primitive.

Indeed, the HMAC is a MAC scheme which is built directly from a hash function like the one discussed in Section 6.4. It is in fact the most widely used MAC scheme on the internet. Formally, its defined as follows:

$$S((k_1, k_2), m) = H(k_1 \oplus \text{opad} \,||H(k_2 \oplus \text{ipad} \,||m))]$$

Taking H as something like SHA256, the above results in a secure MAC scheme. This is also used widely in the TLS protocol. In fact, HMAC can even be proven to be a secure PRF under certain pseudorandomness conditions on the function h used in the Merkle-Damgard construction from Section 6.4. We again refer the reader to [1] for more details on this construction.

6.6 Summary

In this lecture, we took a closer look at how to construct Message Authentication Codes (MACs) and Collision Resistant Hash Functions (CRHFs). In particular, we looked at the most widely used MAC scheme on the internet - the HMAC construction.

Acknowledgement

These scribe notes were prepared by editing a light modification of the template designed by Alexander Sherstov.

References

[1] D. Boneh and V. Shoup. A graduate course in applied cryptography. *Version 0.5*, 2020.