MACs and Hash Functions

Q1: What is the security guarantee provided by the MAC schemes? Compare and contrast with that of encryption schemes.

A1: The threat model of MAC is "Adaptive Chosen-Message Attack", where we assume the attacker can induce the sender to authenticate messages of attacker's choice. The security goal is "Existential Unforgeability", where we want the attacker to be unable to forge a valid tag on any message not authenticated by the sender. So here, we care about the ownership and authentication of the message being sent, rather than it's security. In fact, MAC is related to integrity, where other encryption schemes we have seen so far have been about security. These are orthogonal concerns!

Q2: Compare MAC schemes with other integrity detection or protection methods, such as error-detection codes, error-correction codes, erasure codes, CRC checksums. How are the usage scenarios different? What type of guarantee does each one provide?

A2: Error-detection codes, error-correction codes, erasure codes and CRC checksums all are designed to prevent or recover from random errors that occur during transmission. However, in the case of MAC, what would be perceived as "error" is not actually a random error, but in fact it is the result of an adversary tampering with the message! So, MAC is harder, as the latter methods are designed against random errors, while MAC is designed against an efficient adversary that can do whatever it wants!

Q3: List at least three scenarios where confidentiality alone, without integrity, is not enough of a security measure.

A3: An active attacker can tamper with the data and look at the result, without even actually looking at what is being sent. Several examples are: Replay attacks, unauthenticated messages, or even messages that are thought to be geniune but are actually tampered with.

Q4: Formally define a MAC scheme and write down full security definition (existential unforgeability under adaptive chosen message attack). Why does this definition not provide a VerifyMac oracle to the adversary?

A4: Let us define the $\texttt{Mac-Forge}_{\mathcal{A},\Pi}(n)$ game:

(1) Challenger send 1^n to the adversary.

(2) Challenger chooses a key $k \xleftarrow{R} K$.
(3) Adversary can make $\texttt{poly}(n)$ queries to a Mac oracle, which returns the tag of a message of adversaries choice.
(4) Then the adversary makes an existential forgery attack, by sending a message and a tag (m, t), such that this pair was not obtained from the oracle accesses before.
(5) The challenger looks at (m, t), and if $Vrfy_k(m, t) = 1$ then it outputs 1, which means the adversary wins. Otherwise, it outputs 0.

Existential unforgeability under adaptive chosen message attack is when for all PPT adversaries there exists a negligibly function $\texttt{negl}$ such that:

$$Pr[\texttt{Mac-Forge}_{\mathcal{A},\Pi}(n) = 1] \leq \texttt{negl}(n)$$

The adversary is not given oracle access for $Vrfy$ because if it had one, it could just create any message and get it's tag, then submit it, which would be correct.

Q5: Let F be a pseudorandom function. Show that the following MAC for messages of length $2n$ is insecure: Gen outputs a uniform $k \in \{0,1\}^n$, and to authenticate a message $m_1||m_2$ with $|m_1| = |m_2| = n$, compute the tag $F_k(m_1)||F_k(F_k(m_2))$

A5: Quite simply, we will do 2 queries only and win the game with advantage 1. Let $m_z \leftarrow \{0,1\}^n$.

(1) Query Mac_k with $m_a||m_z$, which yields the tag $t_{az} = F_k(m_a)||F_k(F_k(m_z))$
(2) Query Mac_k with $m_z||m_b$, which yields the tag $t_{zb} = F_k(m_z)||F_k(F_k(m_b))$
(3) We can know use either (m, t) pairs below:
- $m = m_z||m_z$ and $t = t_{zb}[0]||t_{az}[1]$
- $m = m_a||m_b$ and $t = t_{az}[0]||t_{zb}[1]$

Both will be accepted and verified by the challenger!

Q6: Why does one need two different keys for the underlying PRF in the CBC-MAC and NMAC constructions? What happens if one does not perform the step with the second key?

A6: If you query m and obtain t, then query with t and obtain t', you can break the scheme with message $\langle m||t \rangle$ and the tag t'.

Q7: How can one use a single key, instead of multiple keys required in the constructions presented? Essentially, you need to think about ways to convert a single key to multiple keys, in a secure manner. Note

that there are multiple ways of achieving this. Try to write down at least two-three different methods.

A7: A single key can be used to generate a number of keys, an example would be to use a keyed PRF for this, as in: k is the single key, and other keys k_i are obtained with $F_k(i)$. You can also use PRG for this.

Q8: What does it mean for a security bound to be tight? How would you show that some bound is tight?

A8: It means that when a scheme is used for a certain number of messages with a single key, it becomes insecure and the key must be refreshed. This makes the bound a tight one. An example is to say that $Pr[\mathcal{A} \text{ wins}]$ is tightly bounded by $1/2^{\lambda}$ instead of $Pr[\mathcal{A} \text{ wins}] < negl(\lambda)$.

Q9: Compare CPA and CCA security definitions.

A9: In CCA, in addition to CPA, the adversary has the ability to decrypt ciphertexts of it's choice. In the experiment, this is obtained by giving the adversary the access to a decryption oracle. Other than that, the experiments are same.

Q10: How can one obtain authenticated encryption (and CCA-security) using encryption schemes and MAC schemes together? Which methods are proven secure, which are not?

A10: There are three ways to do this:

- **encrypt-and-authenticate**: To send a message m, you calculate both $c \leftarrow Enc_{k_E}(m)$ and $t \leftarrow Mac_{k_M}(m)$ and send $\langle c, t \rangle$. This is so not secure, that it is actually vulnerable to an eavesdropping (ct-only) attack.
- **authenticate-then-encrypt**: To send a message m, you first calculate $t \leftarrow Mac_{k_M}(m)$ and then calculate $c \leftarrow Enc_{k_E}(m||t)$, finally you send just the c. This method is not completely secure, however, if used with rand-CTR mode or rand-CBC, this actually provides authenticated encryption.
- **encrypt-then-authenticate**: To send a message m, you first calculate $c \leftarrow Enc_{k_E}(m)$ and then calculate $t \leftarrow Mac_{k_M}(c)$, finally you send $\langle c, t \rangle$. This method is proven to be an authenticated encryption, always.

Q11: Formally define collision resistance for a hash function family.

A11: Given a hash function $H : X \to Y$, no efficient adversary should be able to find an $x \in X$ and $x' \in X$, such that $H(x') = H(x)$ and $x' \neq x$.

Q12: Write down a full reduction proof for the collision resistance of the Merkle-Damgard scheme, assuming that the underlying hash function is collision resistant. You may be asked to volunteer to perform the proof in class.

A12: Let us call the underlying hash function h, and the full function as H.

Theorem 0.1. *If h is a collision resistant hash function, then H is a collision resistant hash function.*

Proof. Suppose $H(m) = H(m')$. We will build a collision for h. (Contrapositive of the theorem). Let us first assign names to the chaining variables for both hashes:

- $IV = H_0, H_1, \ldots, H_t, H_{t+1} = H(m)$
- $IV = H'_0, H'_1, \ldots, H'_t, H'_{t+1} = H(m')$

Notice how if $H(m) = H(m')$ then $H_{t+1} = H'_{t+1}$. This would mean:

$$h(H_t, m_t||PB) = H_{t+1} = H'_{t+1} = h(H'_t, m'_t||PB')$$

Now thanks to the equality we have a candidate collision for h, it is:

$$h(H_t, m_t||PB) = h(H'_t, m'_t||PB')$$

Note that a collision is when the inputs are different! So if $H_t \neq H'_t$ or $m_t \neq m'_t$ or $PB \neq PB'$ then there is infact a collision.

Suppose that $H_t = H'_t$ and $m_t = m'_t$ and $PB = PB'$. Then this gives us the equality:

$$h(H_{t-1}, m_{t-1}) = H_t = H'_t = h(H'_{t-1}, m'_{t-1})$$

which tells in short:

$$h(H_{t-1}, m_{t-1}) = h(H'_{t-1}, m'_{t-1})$$

So if $H_{t-1} \neq H'_{t-1}$ or $m_{t-1} \neq m'_{t-1}$ then there is collision.

Suppose that $H_{t-1} = H'_{t-1}$ and $m_{t-1} = m'_{t-1}$. Notice how this is like repeating what we did above, one by one from the end of the message to the beginning! Eventually we iterate to the beginning of the messages, and we are left with the two options:

(1) Find collision for h,
(2) $\forall i : m_i = m'_i \implies M = M'$

The second option means that the messages are infact the same, so this wouldn't be a collision. The first one says that there must be a collision in h for there to be a collision on H. Our theorem stated that if h is collision resistant then H is collision resistant, and this proves it. □

Q13: Formally define second preimage resistance for a hash function family.

A13: Given a hash function $H : X \to Y$ and an element $x \in X$, no efficient adversary should be able to find an $x' \in X$ such that $H(x') = H(x)$ and $x' \neq x$.

Q14: Formally define preimage resistance for a hash function family.

A14: Given a hash function $H : X \to Y$ and an element $y \in Y$, no efficient adversary should be able to find an $x \in X$ such that $H(x) = y$.