Message integrity: Confidentiality alone not sufficient, also need <u>message integrity</u>. Otherwise adversary can tamper with the message (e.g., "Send \$100 to Bob" $\longrightarrow$ "Send \$100 to Eve")

In some cases (e.g., software patches), integrity more important than confidentiality

Idea: Append a "tag" (also called a "signature") to the message to prove integrity (property we want is tags should be hard to forge)

this tolerates a <u>single</u> error (better error-correcting codes can do much better)

Observation: The tag should be computed using a keyed-function

↳ Example of keyless integrity check: CRC (cyclic redundancy check) [simple example is to set tag to be the parity]

↳ this was used in SSH v1 (1995) for data integrity! Fixed in SSH v2 (1996)

↳ also used in WEP (802.11b) protocol for integrity — also broken!

Problem: If there is no key, <u>anyone</u> can compute it! Adversary can tamper with message and compute the new tag.

Definition. A message authentication code (MAC) with key-space $\mathcal{K}$, message space $\mathcal{M}$ and tag space $\mathcal{T}$ is a tuple of algorithms $\Pi_{MAC} = (\mathsf{Sign}, \mathsf{Verify})$:

$$\left.\begin{array}{l}\mathsf{Sign}: \mathcal{K} \times \mathcal{M} \to \mathcal{T} \\ \mathsf{Verify}: \mathcal{K} \times \mathcal{M} \times \mathcal{T} \to \{0,1\}\end{array}\right\} \text{ Must be efficiently-computable}$$

Correctness: $\forall k \in \mathcal{K}$, $\forall m \in \mathcal{M}$:

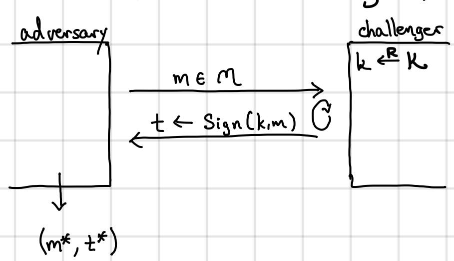
$$\Pr[\mathsf{Verify}(k, m, \mathsf{Sign}(k,m)) = 1] = 1$$

Sign can be a <u>randomized</u> algorithm

Defining security: Intuitively, adversary should not be able to compute a tag on any message without knowledge of the key

↳ Moreover, since adversary might be able to see tags on existing messages (e.g., signed software updates), it should not help towards creating a new MAC

Definition. A MAC $\Pi_{MAC} = (\mathsf{Sign}, \mathsf{Verify})$ satisfies existential unforgeability against $\underbrace{\text{chosen message attacks}}_{\text{adversary gets to choose messages to be signed}}$ (EUF-CMA) if for all efficient adversaries $\mathcal{A}$, $\mathsf{MACAdv}[\mathcal{A}, \Pi_{MAC}] = \Pr[W = 1] = \mathsf{negl}(\lambda)$, where W is the output of the following security game:



As usual, λ denotes the length of the MAC secret key (e.g., $\log|\mathcal{K}| = \mathrm{poly}(\lambda)$)

Note: the key can also be sampled by a special KeyGen algorithm (for simplicity, we just define it to be uniformly random)

Let $m_1, \ldots, m_Q$ be the signing queries the adversary submits to the challenger, and let $t_i \leftarrow \mathsf{Sign}(k, m_i)$ be the challenger's responses. Then, $W = 1$ if and only if:

$$\mathsf{Verify}(k, m^*, t^*) = 1 \text{ and } (m^*, t^*) \notin \{(m_1, t_1), \ldots, (m_Q, t_Q)\}$$

MAC security notion says that adversary cannot produce a <u>new</u> tag on <u>any</u> message even if it gets to obtain tags on messages of its choosing.

First, we show that we can directly construct a MAC from any PRF.

<u>MACs from PRFs</u>: Let $F: \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ be a PRF. We construct a MAC Π_{MAC} over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$ as follows:

Sign(k, m): output $t \leftarrow F(k, m)$

Verify(k, m, t): output 1 if $t = F(k, m)$ and 0 otherwise

<u>Theorem</u>. If F is a secure PRF with a sufficiently large range, then Π_{MAC} defined above is a secure MAC. Specifically, for every efficient MAC adversary A, there exists an efficient PRF adversary B such that

$$\text{MACAdv}[A, \Pi_{MAC}] \leq \text{PRFAdv}[B, F] + \frac{1}{|\mathcal{T}|}.$$

<u>Intuition for proof</u>:
1. Output of PRF is computationally indistinguishable from that of a truly random function.
2. If we replace the PRF with a truly random function, adversary wins the MAC game only if it correctly predicts the random function at a new point. Success probability is then exactly $1/|\mathcal{T}|$.

<u>Implication</u>: Any PRF with large output space can be used as a MAC.

↳ AES has 128-bit output space, so can be used as a MAC

<u>Drawback</u>: Domain of AES is 128-bits, so can only sign 128-bit (16-byte) messages

How do we sign longer messages? We will look at two types of constructions:

1. Constructing a large-domain PRF from a small-domain PRF (i.e., AES)
2. Hash-based constructions

Approach: "compress" the message itself (e.g., "hash" the message) and MAC the compressed representation

Still require unforgeability: two messages should not hash to the same value [otherwise trivial attack: if $H(m_1) = H(m_2)$, then MAC on m_1 is also MAC on m_2]

↳ counter-intuitive: if hash value is shorter than messages, collisions always exist — so we can only require that they are hard to find

Definition. A hash function $H: \mathcal{M} \to \mathcal{T}$ is collision-resistant if for efficient adversaries $\mathcal{A}$,

$$\mathrm{CRHFAdv}[\mathcal{A}, H] = \Pr[(m_0, m_1) \leftarrow \mathcal{A} : H(m_0) = H(m_1)] = \mathrm{negl.}$$

As stated, definition is problematic: if $|\mathcal{M}| > |\mathcal{T}|$, then there always exists a collision m_0^*, m_1^* so consider the adversary that has m_0^*, m_1^* hard coded and outputs m_0^*, m_1^*

↳ Thus, some adversary always exists (even if we may not be able to write it down explicitly)

↳ Two ways to handle

1. Consider a "keyed" hash function $H: \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$.

   Then, for a randomly chosen key $k \xleftarrow{R} \mathcal{K}$, adversary cannot find $m_0 \neq m_1 \in \mathcal{M}$ where $H(m_0) = H(m_1)$

   Note that adversary is given the key — however it can no longer precompute a collision beforehand since it does know the key before it is sampled

2. From an asymtotic standpoint, hash function is a family of functions indexed by a security parameter (which controls the input length and output length. Longer outputs should mean harder to find collisions. Thus we can define hash family $\mathcal{H} = \{ H_\lambda : \{0,1\}^{n(\lambda)} \to \{0,1\}^{\lambda} \}_{\lambda \in \mathbb{N}}$.

   We now can restrict the collision-finding adversary to be "uniform." This means there is a single algorithm that finds a collision for every security parameter $\lambda \in \mathbb{N}$. Namely, algorithm takes λ as input and must output $m_0 \neq m_1$ such that $H_\lambda(m_0) = H_\lambda(m_1)$

   This adversary that has a hard-coded collision only works for one value of λ.

↳ In practice, we have a concrete function (e.g., SHA-256) that does not include security or system parameters

↳ believed to be hard to find a collision even though there are infinitely-many (SHA-256 can take inputs of arbitrary length)

MAC from CRHFs: Suppose we have the following

- A MAC (Sign, Verify) with key-space $\mathcal{K}$, message space $\mathcal{M}_0$ and tag space $\mathcal{T}$ [e.g., $\mathcal{M}_0 = \{0,1\}^{256}$, $\mathcal{M}_1 = \{0,1\}^*$]
- A collision-resistant hash function $H: \mathcal{M}_1 \to \mathcal{M}_0$

Define $S'(k,m) = S(k, H(m))$ and

$V'(k,m,t) = V(k, H(m), t)$

Theorem. Suppose $\Pi_{MAC} = (\text{Sign}, \text{Verify})$ is a secure MAC and H is a CRHF. Then, Π'_{MAC} is a secure MAC. Specifically, for every efficient adversary A, there exist efficient adversaries B_0 and B_1 such that

$$\text{MACAdv}[A, \Pi'_{MAC}] \leq \text{MACAdv}[B_0, \Pi_{MAC}] + \text{CRHFAdv}[B_1, \mathcal{H}]$$

Proof Idea. Suppose A manages to produce a valid forgery t on a message m. Then, it must be the case that

- t is a valid MAC on $H(m)$ under Π_{MAC}
- If A queries the signing oracle on $m' \neq m$ where $H(m') = H(m)$, then A breaks collision-resistance of H
- If A never queries signing oracle on m' where $H(m') = H(m)$, then it has never seen a MAC on $H(m)$ under Π_{MAC}. Thus, A breaks security of Π_{MAC}.

[See Boneh-Shoup for formal argument — very similar to above: just introduce event for collision occurring vs. not occurring]