Message integrity: Confidentiality alone is not sufficient; also need message integrity. Otherwise adversary can tamper with the message.

(e.g., “Send $100 to Bob” → “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: tag should be hard to forge).

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

\[
\begin{align*}
\text{Sign}: K \times M & \rightarrow T \\
\text{Verify}: K \times M \times T & \rightarrow \{0,1\}
\end{align*}
\]

\[ \text{Must be efficiently-computable} \]

Correctness: \( \forall k \in K, \forall m \in M: \)

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

\[ \text{Sign can be a randomized algorithm} \]

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

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

Definition. A MAC \( \text{Timac} = (\text{Sign}, \text{Verify}) \) satisfies existential unforgeability against chosen message attacks (EUF-CMA) if for all efficient adversaries \( A \), \( \text{MACAdv}[A, \text{Timac}] = \Pr[W=1] = \neg\text{neg}1(L) \), where \( W \) is the output of the following security game:

As usual, \( L \) denotes the length of the MAC secret key.

\( (e.g., \log |K| = \text{poly}(n)) \)

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 \( t_1 \leftarrow \text{Sign}(k, m_i) \) be the challenger's responses. Then, \( W=1 \) if and only if:

\[ \text{Verify}(k, m^*, t^*) = 1 \text{ and } (m^*, t^*) \neq (m_1, t_1), \ldots, (m_q, t_q) \]

MAC security notion says that adversary cannot produce a new tag on any 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.
MAC's from PRFs: Let $F: K \times M \rightarrow T$ be a PRF. We construct a MAC $\text{TMAC}$ over $(K, M, T)$ as follows:

1. **Sign**($k, m$): Output $t = F(k, m)$
2. **Verify**($k, m, t$): output 1 if $t = F(k, m)$ and 0 otherwise.

Theorem. If $F$ is a secure PRF with a sufficiently large range, then $\text{TMAC}$ defined above is a secure MAC. Specifically, for every efficient MAC adversary $A$, there exists an efficient PRF adversary $B$ such that $\text{MACAdv}[A, \text{TMAC}] \leq \text{PRFAdv}[B, F] + \frac{1}{|T|}$.

**Intuition for proof**: 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 $\frac{1}{|T|}$.

**Implication**: 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.

**Drawback**: 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
so far, we have focused on constructing a large-domain PRF from a small-domain PRF in order to construct a MAC on long messages

alternative 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 \( H: M \rightarrow T \) is collision-resistant if for efficient adversaries \( A \),

\[
\text{CRHFAdv}(A, H) = \Pr[(m, m') \leftarrow A : H(m) = H(m')] = \text{negl}.
\]

As stated, definition is problematic: if \( |M| > |T| \), then there always exists a collision \( m^*, m^\prime \) so consider the adversary that has \( m^*, m^\prime \) hard-coded and outputs \( m^*, m^\prime \)

- This same adversary always exists (even if we may not be able to write it down explicitly)
- Formally, we need the hash function as being parameterized by an additional parameter (e.g., a "system parameter" or key) so adversary cannot output a hard-coded collision
- 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 \((\text{Sign}, \text{Verify})\) with key space \( K \), message space \( M_0 \) and tag space \( T \)
- A collision-resistant hash function \( H : M_1 \rightarrow M_0 \)

Define \( S'(k, m) = S(k, H(m)) \) and \( V'(k, m, t) = V(k, H(m), t) \)

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

\[
\text{MACAdv}(A, \text{Timac}) \leq \text{MACAdv}(B_0, \text{Timac}) + \text{CRHFAdv}(B_1, H).\]

**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 \( \text{Timac} \)
- 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 \( \text{Timac} \). Thus, \( A \) breaks security of \( \text{Timac} \).

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