So far...

- PRG, PRF, and Symmetric Encryption all from OWFs.
- These are primitives about “hiding” some information.
- What about “authenticating” a message or a source?
- Ideas?
- Can we use Symmetric Encryption?
- Scribe notes volunteers?
Brainstorming

What should a *message authentication code* (MAC) do?

- Should guarantee that only the messages from the intended source are accepted.
  - If MAC comes from the authorized source, it should verify. *(correctness)*
  - Only authorized source can generate the MAC. *(unforgeability)*

What is the adversary allowed to do?

- Can ask to see many MACs on messages of his choice, i.e., \((m_1, \sigma_1), (m_2, \sigma_2), \ldots\)
- Want: cannot generate the MAC for any new message
Message Authentication Codes

Definition (Message Authentication Code)

A message authentication code (MAC) consists of $\{M, K, KG, Tag, Verify\}$ where $M, K$ are message-space and key-space respectively, and:

- $KG(1^n)$ is a PPT key-generation algorithm; it returns a $k \in K$.
- $Tag(k, m)$ is a PPT algorithm which takes as input a key $k \in K$ and a message $m \in M$ and outputs a code $\sigma$.
- $Verify(k, m, \sigma)$ is a PPT algorithm which on input a key $k$, a message $m$, and a code $\sigma$, outputs 1 (accept) or 0 (reject).

The scheme must satisfy:

(correctness): $\forall k \in K, m \in M$, $Verify(k, m, Tag(k, m)) = 1$.

(unforgeability): $\forall$ non-uniform PPT $A$, $\exists$ negligible $\mu$ s.t. $\forall n$: $Pr[A \text{ wins ForgingGame}] \leq \mu(n)$.
The **ForgingGame**$(1^n)$ proceeds between a challenger $Ch$ and adversary $A$ in three steps:

1. **Init:** The challenger generates a key: $k \leftarrow KG(1^n)$.

2. **Learn:** $A$ learns many codes on messages of his choice.
   - $A$ sends a message $m_i \in M$ to $Ch$
   - $Ch$ sends back a code $\sigma_i \leftarrow \text{Tag}(k, m_i)$

   Let $L = \{m_i\}$ be the set of all messages $A$ sends to $Ch$.

3. **Guess:** $A$ outputs a message-code pair $(m, \sigma)$

   $A$ wins if and only if $m \notin L \land \text{Verify}(k, m, \sigma) = 1$. 
A Remark

1. MACs require the two parties to share a secret key
2. Digital Signatures – public-key variant where the secret-key is not shared. (Later classes)
A MAC based on PRF

**Theorem**

\[ PRF \implies MAC \]

- Let \( F \) be a PRF with input-space \( \mathcal{M} = \{0, 1\}^n \), key-space \( \mathcal{K} = \{0, 1\}^n \), and \( KG \) as key-generation algorithm.

- Our MAC scheme has the same message space, key space, and key-generation \( KG \).

- The other two algorithms work as follows:
  - \( \text{Tag}(k, m) = F_k(m) \).
  - \( \text{Verify}(k, m, \sigma) \) outputs 1 if and only if \( \sigma = F_k(m) \).

- Correctness: by definition \( \text{Tag}(k, m) = F_k(m) \) for all \( k, m \).

- What about unforgeability?
Proof of Unforgeability

- Suppose that our MAC is not unforgeable. This means, there is a PPT $A$ who wins the ForgingGame with some noticeable probability $\varepsilon$.

- Therefore, by definition, $A$ outputs $(m, \sigma)$ such that $\sigma = F_k(m)$ with $\varepsilon$ probability such that for $m \notin L$ where $L$ is the list of all messages asked by $A$.

- What happens if we replace $F$ with a truly random function $RF$?
- In the ForgingGame, the challenger does not use $F$ to answer $A$’s queries; instead:
  - It builds a table $T$ (to represent the truly random function $RF$)
  - For each new $m_i$, sends a random $\sigma_i$, and stores $(m_i, \sigma_i)$ in $T$.
  - For each existing $m_i$, simply returns the entry in $T[m_i]$. 
Suppose that $A$ wins the new ForgingGame (which now uses $RF$) with probability $\varepsilon'$.

By security of PRF, $|\varepsilon - \varepsilon'| \leq \mu(n)$ where $\mu$ is negligible; 
$\Rightarrow \varepsilon' \geq \varepsilon - \mu(n)$

But $RF$ is truly random $\Rightarrow$ no-one can guess $RF(m) = \sigma$ with more than $\frac{1}{2^n}$ probability.

Therefore $\varepsilon' < 2^{-n} \Rightarrow \varepsilon - \mu < 2^{-n} \Rightarrow \varepsilon < 2^{-n} + \mu$.

I.e., $\varepsilon$ cannot be noticeable. (Contradiction) $\square$
One-time MAC

- Weaker Security: Adversary is allowed only one query
- Advantage: Unconditional security!
- Analogue of OTP for authentication
- Related reading: Section 7.6 [Boneh-Shoup]