- Recap: Definition of MACs (Message Authentication Codes)
- MACs + CPA secure encryption \(\rightarrow\) CCA secure encryption
- Constructions of MACs
- Hash functions

**Def:** A MAC consists of a PPT alg MAC that takes as input a key \(K\) and a msg \(M\) and outputs \(\text{MAC}(K,M)\). It authenticates the msg \(M\); often called "tag".

If MAC is randomized it is also associated w. a verification procedure that given \((K,M,\text{tag})\) outputs 0/1. We assume that MAC is deterministic and \(K\) is uniform (o.w. need Gen alg)

A MAC is secure against adaptive chosen msg attacks if

\[
\forall PPT A \exists \text{negl function } \mu : N \rightarrow [0,1] \text{ s.t. } \forall n \in N
\]

\[
\Pr \left[ A^{\text{MAC}_{K}}(1^n) = (M,\text{tag}) : M \text{ not queried & } \text{tag} = \text{MAC}(K,M) \right] \leq \mu(n)
\]

- MACs are like digital signatures, where the former is in the secret key setting and the latter is in the public key setting (stay tuned...)

- tag cannot be too short, o.w. it can be guessed (typically 128 bits)
**Theorem:** CPA secure encryption + secure MAC

\[ \implies \]

CCA secure encryption

**Recall:** An encryption scheme \((\text{Gen}, \text{Enc}, \text{Dec})\) is CPA (CCA) secure if \(\forall \text{PPT adv} \exists \text{negl function } \mu: \mathbb{N} \rightarrow [0,1] \text{ s.t. } \forall n \in \mathbb{N}, \forall m_0, m_1 \in M\)

\[
\Pr[A^{\text{Enc}(K, \cdot), \text{Dec}(K, \cdot)}(\text{Enc}(K, m_b)) = b] \leq \frac{1}{2} + \mu(n)
\]

*Given only in CCA, in which case A cannot query decryption oracle with challenge CT*

**Proof:** Append to the CPA secure CT a MAC:

\[
\text{Enc}(K, M) \quad \text{MAC}(K_2, CT)
\]

\[
\text{CT} \quad \text{tag}
\]
Formally, given a CPA secure enc. scheme \((\text{Gen, Enc, Dec})\)
& given MAC, define the following CCA secure scheme
\((\text{Gen}', \text{Enc}', \text{Dec}')\):

\(\text{Gen}'(1^n)\) : Generate \(K_1 \leftarrow \text{Gen}(1^n)\) , \(K_2 \leftarrow \{0,1\}^n\)
output \((K_1, K_2)\)

\(\text{Enc}'((K_1, K_2), M)\) : Compute \(\text{CT} \leftarrow \text{Enc}(K_1, M)\)
\(\text{tag} \leftarrow \text{MAC}(K_2, \text{CT})\)

Output \(\text{CT}' = (\text{CT}, \text{tag})\)

\(\text{Dec}'((K_1, K_2), (\text{CT}, \text{tag}))\) : Check if \(\text{tag} = \text{MAC}(K_2, \text{CT})\)
If not output \(\bot\).
Otherwise output \(M = \text{Dec}(K_1, \text{CT})\).

Intuitively, the reason this is CCA secure is that the decryption oracle is useless unless adv. knows a valid tag. But it knows a valid tag only if it got this CT from the enc oracle, in which case the decryption oracle is useless.
**Hash Functions:**

**Def:** A hash family is a family of functions $H = \{ h_s \}$ where $h_s : \{0,1\}^* \rightarrow \{0,1\}^d$. Typically think of input space as $\{0,1\}^*$ and output space children.

Given $s, x$ one can efficiently compute $h_s(x)$. $s$ is called the seed, and is public.

$h_s(x)$ is called the hash value.

**Ideal Hash Function**

Truly random function $H : \{0,1\}^* \rightarrow \{0,1\}^d$

A truly random function is not efficiently computable so we give all parties oracle access to this ideal hash function.

This idealized (theoretical) model is called the Random Oracle Model (ROM).

Many cryptographic primitives use hash functions, and these primitives are often proved to be secure in the ROM.

In practice, the hash family is usually implemented with a single hash function SHA-256, Secure Hash Algorithm.
Desired properties

1. **One-Way**: \( \forall \text{PPT } A \forall n \in \mathbb{N} \)
   \[ \Pr[A(s, h_0(x)) \in h_0^{-1}(h_0(x))] = \text{negl}(d) \]
   \[ s,x \leftarrow \text{fool}^* \]
   \( \text{ROM:} \) Time to invert \( \Theta(2^d) \)

2. **Collision-Resistant**: \( \forall \text{PPT } A \forall n \in \mathbb{N} \)
   \[ \Pr[A(s) = (x, x') \text{ s.t. } x \neq x' \text{ & } h_0(x) = h_0(x')] = \text{negl}(d) \]
   \[ s \leftarrow \text{fool}^* \]
   \( \text{ROM:} \) Time to find collisions \( \Theta(2^{2d}) \)
   \( \checkmark \) - Birthday paradox:
   choose randomly \( x_1, x_2, \ldots \) until a collision is found.

3. **Target Collision-Resistant**: \( \forall \text{PPT } A \forall n \in \mathbb{N} \)
   \[ \Pr[A(s, x) = x' \text{ s.t. } x' \neq x \text{ & } h_0(x) = h_0(x')] = \text{negl}(d) \]
   \( \text{ROM:} \) Time to find a target collision \( \Theta(2^d) \)

4. **Pseudo Randomness**: A PPT adversary cannot distinguish between oracle access to \( h_0 \) & a truly random \( H \).

   Formally, \( \forall \text{PPT } A \forall n \in \mathbb{N} \)
   \[ \left| \Pr_{s \leftarrow \text{fool}^*}[A_{h_0}(s) = 1] - \Pr_{H}[A_{H}(s) = 1] \right| = \text{negl}(d) \]

   where \( H: \{0,1\}^* \rightarrow \{0,1\}^d \) is a random function.

   \( \text{ROM:} \checkmark \)