

MACs from PRFs: Let $F: K \times M \rightarrow T$ be a PRF. We construct a MAC Π_{mac} over (K, M, 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

Theorem. 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

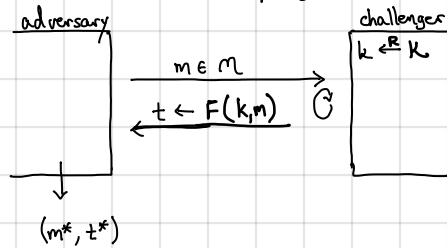
$$\text{MACAdv}[A, \Pi_{\text{mac}}] \leq \text{PRFAdv}[B, F] + \frac{1}{|T|}.$$

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 $1/|T|$.

Proof. We define the following sequence of hybrid experiments:

Hyb_0 : This is the MAC security game:



Goal: Show for all efficient A :

$$\Pr[\text{Hyb}_0(A) = 1] = \text{negl}.$$

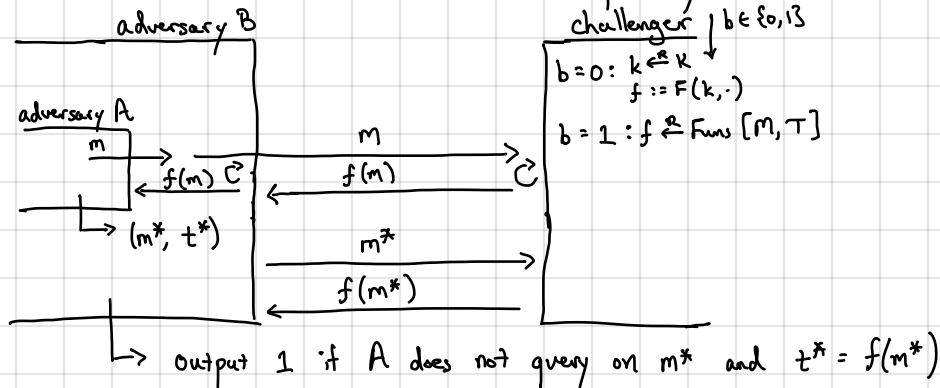
Experiment outputs 1 if adversary did not query on m^* and $t^* = F(k, m^*)$

Hyb_1 : Same as Hyb_0 except we replace $F(k, \cdot)$ with $f(\cdot)$ where $f \xleftarrow{\$} \text{Funs}[M, T]$

Lemma 1. If F is a secure PRF, then for all efficient adversaries A ,

$$|\Pr[\text{Hyb}_0(A) = 1] - \Pr[\text{Hyb}_1(A) = 1]| = \text{negl}.$$

Proof. Suppose there exists efficient A such that above probability is ϵ . We construct B as follows:



$$\left. \begin{array}{l} \Pr[B \text{ outputs } 1 \mid b=0] = \Pr[\text{Hyb}_0(A) = 1] \\ \Pr[B \text{ outputs } 1 \mid b=1] = \Pr[\text{Hyb}_1(A) = 1] \end{array} \right\} \text{PRFAdv}[B, F] = \epsilon$$

Lemma 2. For all adversaries A , $\Pr[\text{Hyb}_1(A) = 1] = \frac{1}{|T|}$.

Proof. $\text{Hyb}_1(A)$ outputs 1 if A predicts value of f at m^* . Since f is uniform, A succeeds with probability at most $1/|T|$.

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

As stated, definition is problematic: if $|M| > |T|$, 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)

→ Formally, we model the hash function as being parameterized by an additional parameter (e.g., a "system parameter" or a "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 (Sign, Verify) with key-space K , message space M_0 and tag space T
 - A collision-resistant hash function $H: M_1 \rightarrow M_0$
- $\left[\text{e.g., } M_0 = \{0,1\}^{256} \right]$
 $M_1 = \{0,1\}^*$

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

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

Theorem. Suppose $\Pi_{\text{MAC}} = (\text{Sign}, \text{Verify})$ 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

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

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 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]

Constructing above is simple and elegant, but not used in practice

- Disadvantage 1: Implementation requires both a secure MAC and a secure CRHF: more complex, need multiple software/hardware implementations

- Disadvantage 2: CRHF is a key-less object and collision finding is an offline attack (does not need to query verification oracle)
Adversary with substantial preprocessing power can compromise collision-resistance (especially if hash size is small)

Birthday attack on CRHFs. Suppose we have a hash function $H: \{0,1\}^n \rightarrow \{0,1\}^l$. How might we find a collision in H (without knowing anything more about H)

Approach 1: Compute $H(1), H(2), \dots, H(2^l + 1)$

↳ By Pigeonhole Principle, there must be at least one collision - runs in time $O(2^l)$

Approach 2: Sample $m_i \in \{0,1\}^n$ and compute $H(m_i)$. Repeat until collision is found.

How many samples needed to find a collision?

Theorem (Birthday Paradox). Take any set S where $|S| = n$. Suppose $r_1, \dots, r_l \xleftarrow{R} S$. Then,

$$\Pr[\exists i \neq j : r_i = r_j] \geq 1 - e^{-\frac{l(l-1)}{2n}}$$

Proof. $\Pr[\exists i \neq j : r_i = r_j] = 1 - \Pr[\forall i \neq j : r_i \neq r_j]$
 $= 1 - \Pr[r_2 \notin \{r_1\}] \cdot \Pr[r_3 \notin \{r_1, r_2\}] \cdot \dots \cdot \Pr[r_l \notin \{r_1, \dots, r_{l-1}\}]$

$$= 1 - \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \dots \cdot \frac{n-l+1}{n}$$

$$= 1 - \prod_{i=1}^{l-1} \left(1 - \frac{i}{n}\right)$$

$$\geq 1 - \prod_{i=1}^{l-1} e^{-i/n} \quad \text{since } 1+x \leq e^x \text{ for all } x \in \mathbb{R} \quad \left[e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \dots \right]$$

$$= 1 - e^{-\sum_{i=1}^{l-1} i/n} = 1 - e^{-\frac{1}{n} \sum_{i=1}^{l-1} i}$$

$$= 1 - e^{-\frac{(l-1)l}{2n}}$$

automatically holds for $x \leq -1$

dominant term when $|x| < 1$
positive for all $x > 0$

When $l \geq 1.2\sqrt{n}$, $\Pr[\text{collision}] = \Pr[\exists i \neq j : r_i = r_j] > \frac{1}{2}$. [For birthdays, $1.2\sqrt{365} \approx 23$]

↳ Birthdays not uniformly distributed, but this only increases collision probability.

[Try proving this!]

For hash functions with range $\{0,1\}^l$, we can use a birthday attack to find collisions in time $\sqrt{2^l} = 2^{l/2}$

↳ For 128-bit security (e.g., 2^{128}), we need the output to be 256-bits (hence SHA-256)

↳ Quantum collision-finding can be done in $2^{l/3}$ (cube root attack), though requires more space

can even do it with constant space!

[via Floyd's cycle finding algorithm]