for NMAC, $F: \mathcal{K} \times \mathcal{X} \to \mathcal{K}$

<u>Theorem.</u> Let $F: \mathcal{K} \times \mathcal{X} \to \mathcal{X}$ be a secure PRF. Let Π_{ECBC} be the encrypted CBC MAC formed by F. Then, for all MAC adversaries A, there exists a PRF adversary B where

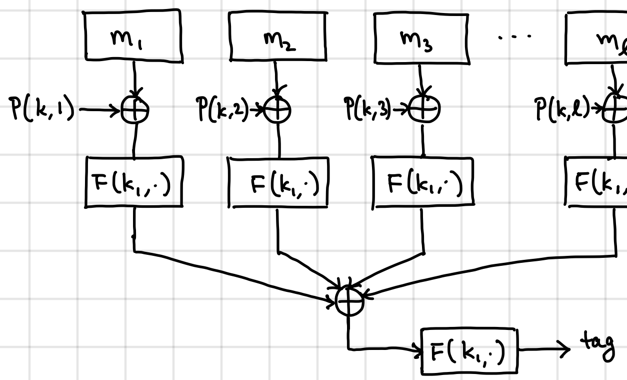
$$\mathrm{MACAdv}[A, \Pi_{ECBC}] \leq 2 \cdot \mathrm{PRFAdv}[B, F] + \frac{Q^2 (\ell+1)^2}{|\mathcal{X}|}$$

quadratic dependence on Q arises for similar reason as in analyzing CPA security (argue that all inputs to PRF are unique)

<u>Proof.</u> See Boneh-Shoup, Chapter 6.

<u>Implication</u>: Block size of PRF is important!
- 3DES: $|\mathcal{X}| = 2^{64}$; need to update key after $< 2^{32}$ signing queries
- AES: $|\mathcal{X}| = 2^{128}$; can use key to sign many more messages ($\sim 2^{64}$ messages)

A parallelizable MAC (PMAC) — general idea:



$P(k,\cdot)$ derived as $F(k_1, 0^n)$ — so key is just k_1

$P(k,\cdot)$ are important — otherwise, adversary can <u>permute</u> the blocks

↳ "mask" term is of the form $\gamma_i \cdot k$ where multiplication is done over $GF(2^n)$ where n is the block size (constants γ_i carefully chosen for efficient evaluation)

Can use similar ideas as CMAC (randomized prefix-free encoding) to support messages that is not constant multiple of block size

Parallel structure of PMAC makes it easily updateable (assuming F is a PRP)

↳ suppose we change block i from $m[i]$ to $m'[i]$:

compute $F^{-1}(k_1, \mathrm{tag}) \oplus \underbrace{F(k_1, m[i] \oplus P(k,i))}_{\text{old value}} \oplus \underbrace{F(k_1, m'[i] \oplus P(k,i))}_{\text{new value}}$

PMAC is "incremental": can make local updates without full recomputation

In terms of performance:
- On sequential machine, PMAC comparable to ECBC, NMAC, CMAC
- On parallel machine, PMAC much better

Best MAC we've seen so far, but not used...
<u>Reason</u>: patents :([not patented anymore!]

<u>Summary</u>: Many techniques to build a large-domain PRF from a small-domain one (domain extension for PRF)

↳ Each method (ECBC, CMAC, PMAC) gives a MAC on <u>variable-length</u> messages

↳ Many of these designs (or their variants) are <u>standardized</u>

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 $H: \mathcal{M} \to \mathcal{T}$ 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)

↳ 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 $\mathcal{K}$, message space $\mathcal{M}_0$ and tag space $\mathcal{T}$
- A collision-resistant hash function $H: \mathcal{M}_1 \to \mathcal{M}_0$

[e.g., $\mathcal{M}_0 = \{0,1\}^{256}$, $\mathcal{M}_1 = \{0,1\}^*$]

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

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

Theorem. Suppose $\Pi_{MAC} = (\text{Sign}, \text{Verify})$ is a secure MAC and H is a CRHF. Then, Π'_{MAC} is a secure MAC. Specifically, for every efficient adversary $\mathcal{A}$, there exist efficient adversaries $\mathcal{B}_0$ and $\mathcal{B}_1$ such that

$$\mathrm{MACAdv}[\mathcal{A}, \Pi'_{MAC}] \leq \mathrm{MACAdv}[\mathcal{B}_0, \Pi_{MAC}] + \mathrm{CRHFAdv}[\mathcal{B}_1, H]$$

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 $H: \{0,1\}^n \to \{0,1\}^\ell$. How might we find a collision in H (without knowing anything more about H)

Approach 1: Compute $H(1), H(2), \ldots, H(2^\ell + 1)$

↳ By Pigeonhole Principle, there must be at least one collision — runs in time $O(2^\ell)$ ← size of hash output space

Approach 2: Sample $m_i \xleftarrow{R} \{0,1\}^n$ 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 $r_1, \ldots, r_\ell \xleftarrow{R} S$. Then,

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

Proof.

$$\begin{aligned}
\Pr[\exists i \neq j : r_i = r_j] &= 1 - \Pr[\forall i \neq j : r_i \neq r_j] \\
&= 1 - \Pr[r_2 \notin \{r_1\}] \cdot \Pr[r_3 \notin \{r_2, r_1\}] \cdot \cdots \cdot \Pr[r_\ell \notin \{r_{\ell-1}, \ldots, r_1\}] \\
&= 1 - \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \cdots \cdot \frac{n-\ell+1}{n} \\
&= 1 - \prod_{i=1}^{\ell-1} \left(1 - \frac{i}{n}\right) \\
&\geq 1 - \prod_{i=1}^{\ell-1} e^{-i/n} \quad \text{since } 1 + x \leq e^x \text{ for all } x \in \mathbb{R}
\end{aligned}$$

automatically holds for $x \leq -1$

$$\left[e^x = 1 + x + \underbrace{\frac{x^2}{2} + \frac{x^3}{6} + \cdots}_{\text{positive for all } x > 0} \right]$$

dominant term when $|x| < 1$

$$\begin{aligned}
&= 1 - e^{\sum_{i=1}^{\ell-1} -i/n} = 1 - e^{-\frac{1}{n}\sum_{i=1}^{\ell-1} i} \\
&= 1 - e^{-\frac{(\ell-1)\ell}{2n}}
\end{aligned}$$

When $\ell \geq 1.2\sqrt{n}$, $\Pr[\text{collision}] = \Pr[\exists i \neq j : r_i = r_j] > \frac{1}{2}$. [For birthdays, $1.2\sqrt{365} \approx 23$] ← number of people in a room to have a common birthday

↳ Birthdays not uniformly distributed, but this only increases collision probability. [Try proving this!]

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

can even do it with constant space! [via Floyd's cycle finding algorithm]

↳ 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 $2^{\ell/3}$ (cube root attack), though requires more space