Lecture 4 - Message Authentication Codes
Plan: MACs
- Definition
- PRF
- Construction: Short Msg
- Small mac to big MAC

Last time...

CRHF

\[ \begin{align*}
    d_1 &= H(m_{s1}) \\
    d_2 &= H(m_{s2})
\end{align*} \]

Problem: Need to fetch digest of each file?

Today: MAC requires shared secret. How to get?
Message Authentication Codes (MAC)

\[ K \in \mathbb{Z}_0, 13^{128} \]

![Diagram of message authentication codes (MAC)]

- How can server be sure that msg came from client & not from attacker?

- Parties share a secret key - e.g. random 128-bit string
  * Why 128 bits?
  * How do they agree on shared secret?

Plan: Client appends an authentication “tag” \( t \) to each msg. Server can check that \((m, t)\) pair is valid before accepting msg \( m \).

For now, we are not trying to hide \( m \) from attacker. ▼ No encryption
**MAC Syntax**

- **Key space** $K = \{0,1\}^{128}$
- **Msg space** $M = \{0,1\}^{40}$
- **Tag space** $T = \{0,1\}^{128}$

**Examples**

- $\{0,1\}^a$ for some $a$
- $\{0,1\}^+$
- $\{0,1\}^{\text{poly}(1)}$

**In theory**

- $\{0,1\}^*$

---

**One algorithm**

$$\text{MAC}(k, m) \rightarrow t$$

---

**Security - What is the right notion?**

- Attacker gets to see tags on many msgs of their choosing.
- Cannot produce a tag on new msg.
- Why would we give attacker so much power?

- Defined using a game
  
  * Can think of challenge as grades for lab assignment - determines a successful attack
MAC Security

Challenger

\( k \in \mathbb{G}_k \)

\( m_i \in \mathcal{M} \)

\( t_i \leftarrow \text{MAC}(k, m_i) \)

\( (m^*, t^*) \)

\( \frac{1}{2} \) if \( \text{MAC}(k, m^*) = t^* \)

& \( m^* \notin \{m_1, m_2, \ldots \} \)

0, o.w.

Adversary A

 repeats poly many times

A MAC \( \Xi(\text{MAC, Ver}_S) \) is secure if for all eaves and A

\[ \Pr[A \text{ wins in MAC game }] \leq \text{"negligible"} \]

"Existential unforgeability under chosen msg attack" EUF-CMA
General warning:

* You always use a pre-built MAC — never try to build your own.
* Lots of tricky details (padding, etc.) that are easy to mess up.

Common source of sec problems.
**MAC for Short Messages**

**PLAN**
1. Construct MAC with gigantic random key. (impractical)
2. Replace long key with short key

We want to construct MAC on n-bit msgs (e.g. \( n = 128 \))

\[
\begin{align*}
\mathcal{M} &= \{0,1\}^n \\
\mathcal{T} &= \{0,1\}^\lambda \\
\mathcal{D} &= \{0,1\}^{2^n}
\end{align*}
\]

**Construction**

\[
\text{MAC}(k = (r_0, r_1, r_2, \ldots), m) := \text{Output } t_m
\]

**Security:** To forge MAC on new msg \( m^* \), adv must guess \( r_{m^*} \). \( \Pr[\text{win}] \leq 2^{-\lambda} \)

**Problem:** Exponentially large key.
Pseudorandom Function (PRF)

We would like to generate a gigantic large random-looking key from a small key.

\[ \rightarrow \text{Not possible to generate more "true" randomness.} \]

**But**, it is possible to generate pseudo-randomness "looks random" to any comp. bounded observer.

**Key Primitive**: Pseudo-random Function (PRF)

- **Key Space**: \( \mathcal{K} \)
- **Input Space**: \( \mathcal{K} \)
- **Output Space**: \( \mathcal{Y} \)

\[ \text{F}: \mathcal{K} \times \mathcal{K} \rightarrow \mathcal{Y} \]

**Intuition**: If \( k \) is secret, random

\[ \text{F}(k,1), \text{F}(k,2), \text{F}(k,3), \ldots \]

all "look random."

We use PRF for many things—not just MAC.
PRF Security

Adv gets to make arbitrary queries to $F(k, \cdot)$ or to random fn. Can’t distinguish $\Rightarrow$ PRF secure.

Challenger (b)

\[
\begin{align*}
\mathcal{C}_b \leftarrow \mathcal{X} \\
\mathcal{S}_0(x) & := F(k, \cdot) \\
\mathcal{S}_1(x) & \leftarrow (\text{Random} \ S_n \text{ from} \ x \rightarrow y) \\
\end{align*}
\]

Adversary $\mathcal{A}$

\[
\begin{align*}
\mathcal{A} \leftarrow \text{repeats} \\
\end{align*}
\]

\[
\mathcal{S}_b(x) \rightarrow
\]

\[
\text{Let } \mathcal{W}_b \text{ be the event that} \\
\text{A outputs } “1” \text{ in world } b.
\]

\[
\text{PRF}_{\text{Adv}}[\mathcal{A}, F] := | Pr[\mathcal{W}_b] - Pr[\mathcal{W}_s]| 
\]

\[
\text{We say a PRF } F \text{ is secure if } \forall \epsilon < \text{adv } \mathcal{A} \nabla \\
\text{PRF}_{\text{Adv}}[\mathcal{A}, F] \leq \text{negl}
\]
Constructing PRF

* As with CRTF, we don't know whether PRFs exist unconditionally. Need assumptions.

* Can build from any "one-way fn" (not obvious)
  - factoring, SHA2, SHA3, etc.

* Most common ones are fixed in govt standards
  - AES block cipher (actually PRP) - 1998

  AES: $K \times \{0,1\}^{128} \rightarrow \{0,1\}^{128}$

    - 128 bits
    - 256

  HMAC constructions are also popular PRFs.

  Best attack on AES 128: time $= 2^{126}$

* We just assume that AES is a good PRF

  Could always be wrong.

  BUT, under assumption that AES is secure PRF, can construct secure MAC
MAC for Short Msgs from PRF

Let $F: \mathbb{K} \times \mathbb{X} \rightarrow \mathbb{Y}$ be PRF

MAC Scheme:

$\mathcal{M} = \mathbb{K}$

$\mathcal{Y} = \mathbb{Y}$

$\text{Mac}(k, m) := F(k, m)$

MAC for long msgs?

If you have PRF w/ 256-bit input, can hash with CRHF & memo the hash

$H: \mathbb{M} \rightarrow \mathbb{K}$

$F: \mathbb{K} \times \mathbb{X} \rightarrow \mathbb{Y}$

$\text{Mac}(k, m) := F(k, H(m))$

“Hash-and-Sign”

Problem: CRHF is relatively slow.
**Bad Idea**

MAC for two-block msg

\[ \text{MAC}_{\text{big}}(k, m_1 \| m_2) := \text{MAC}(k, m_1) \| \text{MAC}(k, m_2) \]

Problem: Mix & match attack

Given \( \text{MAC}_{\text{big}}(k, 000 \| m) \) and \( \text{MAC}_{\text{big}}(m \| 000) \) can construct \( \text{MAC}_{\text{big}}(m \| m) \)
MAC for long msgs with keyed hashing

Let $H : \mathcal{K} \times X \to X$

We say that $H$ is a universal hash fn if

$$\Pr_{k \in \mathcal{K}} \left[ H(k, m) = H(k, m') \right] \leq \text{negl.}$$

Example: $\mathcal{K} = X = \mathbb{Z}_p$, $p$ prime $\leq 2^{256}$

$$H(k, (m_0, m_1, m_2, \ldots, m_{l-1})) := m_0 + m_k + m_2 k^2 + \ldots + m_{l-1} k^{l-1}$$

$$\Pr_{k \in \mathcal{K}} \left[ H(k, m) = H(k, m') \right]$$

$$= \Pr_{k \in \mathcal{K}} \left[ H(k, m) - H(k, m') = 0 \right]$$

$$= \Pr_{k \in \mathcal{K}} \left[ (m_0 - m'_0) + (m_1 - m'_1) k + (m_2 - m'_2) k^2 + \ldots + (m_{l-1} - m'_{l-1}) k^{l-1} = 0 \right]$$

$$= \Pr \left[ \text{non-zero degree } \leq l-1 \text{ poly evaluates to 0 on random point} \right]$$

$$\leq \frac{l-1}{p}$$

Bonus: Can evaluate $H$ in parallel on many cores.
MAC for long msgs: Construction

[Boneh-Shoup Thm 7.7]

PRF: \( F: \mathcal{K} \times \mathcal{X} \to \mathcal{Y} \)

UHF: \( H: \mathcal{K} \times \mathcal{X}^2 \to \mathcal{Y} \)

\[ \Rightarrow \text{MAC key space } \mathcal{K}^2 \]

\[ \text{msg space } \mathcal{X}^2 \]

\[ \text{tag space } \mathcal{Y} \]

\[ \text{MAC } (k_1, k_2, m) := F(k_1, H(k_2, m)) \]

Notes:

1. Need \( |\mathcal{X}| \geq 2^{256} \) for 128-bit security

   \[ \Rightarrow \text{Can't use with AES PRF!} \]

   \[ \Rightarrow \text{B/c of birthday attack on } H \]

2. UHF \( H \) is much faster than SHA256 on machine w/o AVX support for SNA

   \[ \approx 4900 \text{ MB/s} \quad \text{UHF (poly1305)} \]

   \[ \approx 500 \text{ MB/s} \quad \text{SHA256} \]

   UHF has hidden key \( \Rightarrow \) attacker's job harder
MACs we use in practice

* Typically we use MACs in conjunction with encryption. “AEAD.” AES GCM, ChaCha20-Poly1305

* Underlying MACs look like the UHF construction

* Main difference is that they use a slightly different keyed hash with fresh hash key on each MAC tag (derived from PKE)

  - Slightly stronger security for given choice of hash output size.

* “Carter - Wegman MAC” is most common construction - underlies AES-GCM, Poly1305.

* HMAC is another very popular one based on SHA2, SHA3, etc. instead of AES