Lecture 4 - Message Authentication Codes

6.1600 - fall 2023

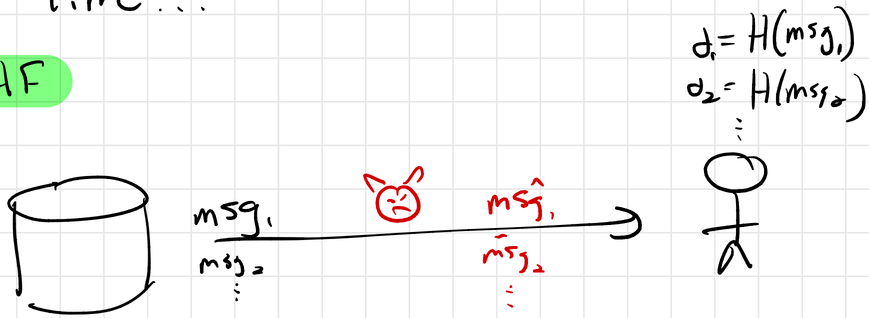
Corrigan-Gibbs & Zeldovich
MIT

Plan: MACs

- Definition
- PRF
- Construction: Short Msg
- Small mac to big MAC

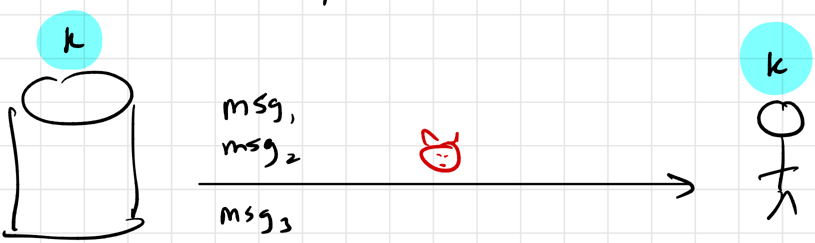
Last time...

CRHF

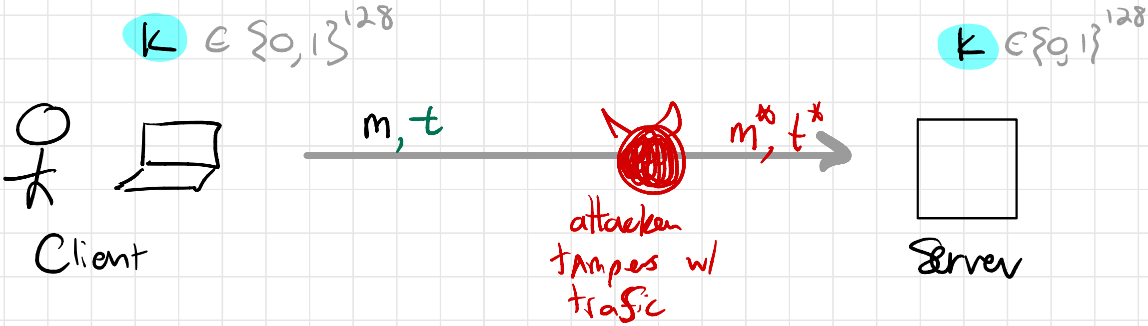


Problem: Need to fetch digest of each file?

Today: MAC Requires shared secret. How to get?



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? Discuss later

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 <u>not</u> trying to hide m from attacker
↳ No encryption

MAC Syntax

		Examples	In theory
Key space	$\mathcal{K}$	$= \{0,1\}^{128}$	$\{0,1\}^{\lambda}$
Msg space	$\mathcal{M}$	$= \{0,1\}^{\leq 2^{40}}$	$\{0,1\}^{poly(\lambda)}$
Tag space	$\mathcal{T}$	$= \{0,1\}^{128}$	$\{0,1\}^{\lambda}$

One algorithm

$$MAC(k, m) \to t$$

When MAC is randomized there can be a separate Verify fn. Won't show.

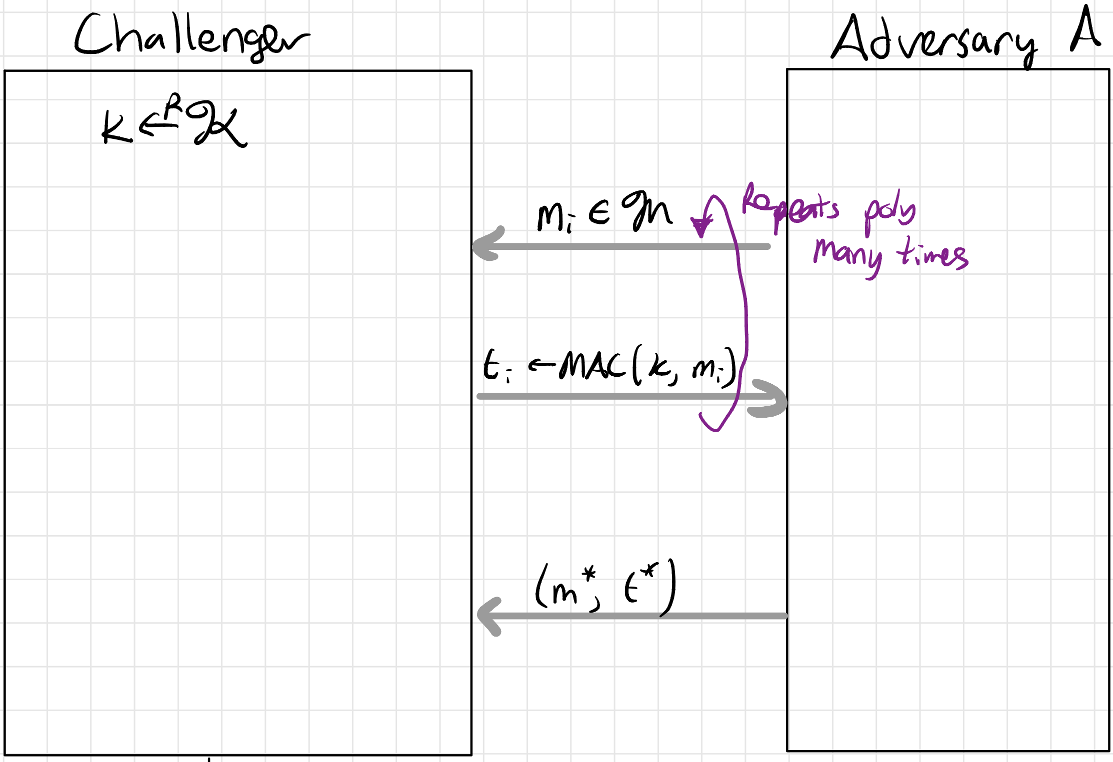
Security – What is the right notion?

- Attacker gets to see tags on many msgs of their choosing.
- Cannot produce a valid tag on new msg.

→ Why would we give attacker so much power?

- Defined using a game
  - * Can think of challenger as grader for lab assignment – determines a successful attack

MAC Security



$$\begin{cases} 1 & \text{if } MAC(k, m^*) == t^* \\ & \quad \& \ m^* \notin \{m_1, m_2, \dots\} \\ 0 & \text{o.w.} \end{cases}$$

A MAC $\Sigma = (MAC, Verify)$ is secure if for all eff adv 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. $n = 128$)

$$\mathcal{M} = \{0,1\}^n$$

$$\mathcal{T} = \{0,1\}^\lambda$$

$$\mathcal{K} = \{0,1\}^{\lambda 2^n}$$

One λ-bit tag for each n-bit msg.

00000	00001	00010		11111
t_0	t_1	t_2		t_{2^n-1}

Construction

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

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

Problem: Exponentially large key!

Pseudorandom Function (PRF)

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

→ Not possible to generate more "true" randomness.

BUT, it <u>is</u> possible to generate $\underbrace{\text{pseudo-randomness}}_{\text{"looks random" to any comp. bounded observer}}$

Key Primitive: Pseudo-random Function (PRF)

Key space $\mathcal{K}$
Input space $\mathcal{X}$
Output space $\mathcal{Y}$

$$F: \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$$

Intuition: If k is secret, random

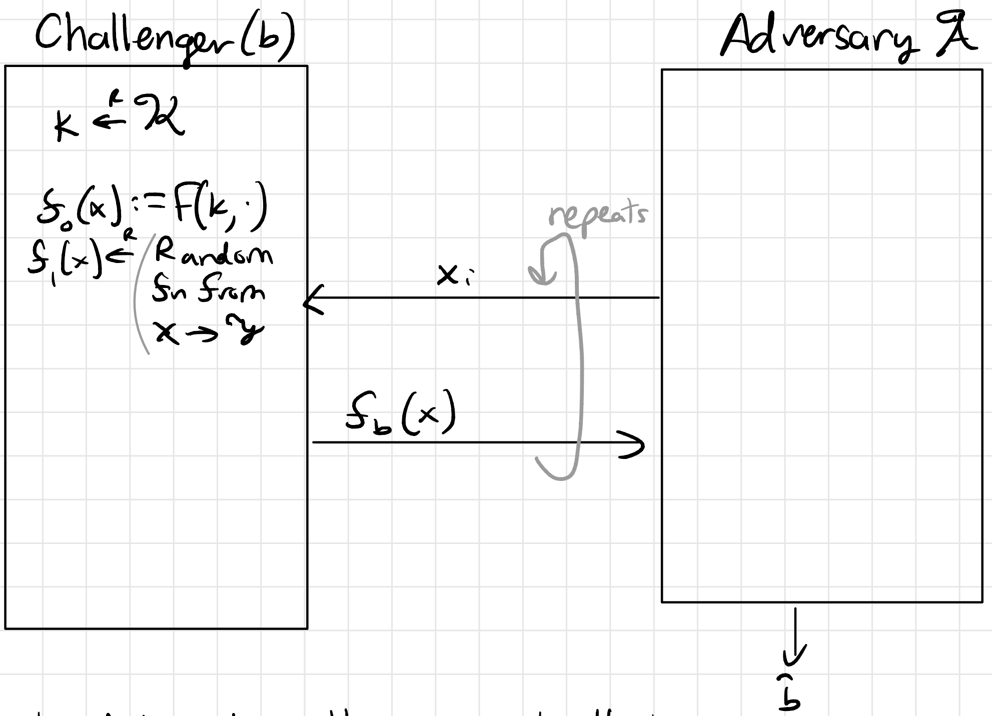
$F(k, \text{"1"}), F(k, \text{"2"}), F(k, \text{"3"}), \ldots$

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.



Let W_b be the event that $\mathcal{A}$ outputs "1" in world b.

$$\text{PRFAdv}[\mathcal{A}, F] := \left| \Pr[W_0] - \Pr[W_1] \right|$$

We say a PRF F is secure if $\forall$ eff adv $\mathcal{A}$

$$\text{PrFAdv}[\mathcal{A}, F] \leq \text{negl}$$

Constructing PRF

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

* Can build from any "one-way fn" (non obvious)
  ↳ factoring, SHA2, SHA3, etc.

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

$$AES : \underbrace{\mathcal{K}}_{\substack{128 \\ 192 \text{ bits} \\ 256}} \times \{0,1\}^{128} \to \{0,1\}^{128}$$

HMAC constructions are also popular PRFs.
Best attack on AES 128 : time $\approx 2^{126}$.

* 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 $F: \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be PRF

MAC Scheme:

$\mathcal{M} = \mathcal{X}$

$\mathcal{T} = \mathcal{Y}$

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

MAC for long msgs?

Why?

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

$H: \mathcal{M} \longrightarrow \mathcal{X}$ [CRHF]
$F: \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$

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

"Hash-and-Sign"

Problem: CRHF is relatively slow.

Bad Idea

MAC for two-block msg

$$MAC_{Big}(k, m_1 \| m_2) := MAC(k, m_1) \| MAC(k, m_2)$$

Problem: Mix & match attack

Given $MAC_{Big}(k, 000 \| m)$ and $MAC_{Big}(m \| 000)$
can construct $MAC_{Big}(m \| m)$

MAC for long msgs with keyed hashing

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

We say that H is a universal hash fn if $\forall\ m \neq m'$

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

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

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

$$\Pr[H(k, m) = H(k, m')]$$

$$= \Pr[H(k, m) - H(k, m') = 0]$$

$$= \Pr[(m_0 - m_0') + (m_1 - m_1')k + (m_2 - m_2')k^2 + (m_{\ell-1} - m_{\ell-1}')k^{\ell-1} = 0]$$

$= \Pr[$non-zero degree $\leq \ell - 1$ poly evaluates to 0 on random point$]$

$$\leq \frac{\ell - 1}{p}.$$

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: $H: \mathcal{K} \times \mathcal{X}^{\ell} \to \mathcal{Y}$

$\Rightarrow$ MAC key space $\mathcal{K}^2$
msg space $\mathcal{X}^{\ell}$
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 [see book for pf]
   ↳ Can't use with AES PRF!
   ↳ B/c of birthday attack on H

2. UHF H is much faster than SHA256 on machine w/o HW support for SHA

   $\approx$ 4900 MB/s UHF (poly1305)
   $\approx$ 500 MB/s SHA256

UHF has hidden key $\Rightarrow$ attacker's job harder
↳ can simplify construction.

MACs we use in practice

* Typically we use MACs in conjuntion 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 PRF)
    ↳ 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