Lecture 3: Message Authentication Codes (MACs)

Logistics: Lab 1 is out, due Thursday Sept. 30th.

So far: Authentication people:

* something you know (password)
* something you have (device)
* something you are (biometrics)

Collision resistant hash functions

authenticating files, code, data ⟵ A user authenticating its own files

Today: Authenticate communication

$$\boxed{\text{Alice}} \xrightarrow{\text{Msg}} \boxed{\text{Bob}}$$

Goal: Bob wants to know that the message indeed came from Alice

Example: $\boxed{\text{Alice}} \xrightarrow{\text{password, instruction}} \boxed{\text{Server}}$

How does the server know the instruction came from Alice?

Message Authentication Code:

Assumes the communicating parties share a random secret key K.

$\overset{K}{\boxed{\text{Alice}}} \xrightarrow{\text{Msg}} \overset{K}{\boxed{\text{Bob}}}$

Imossible unless Alice knows a secret that the adv doesn't know, and that Bob can somehow verify

It Consists of two function:

A signing function S(K,M) that produces a "tag" for the message M.

A verification function V(K,M,tag) and outputs 0/1.

Often V checks tag by recomputing S(K,M), in which case we can define a single alg, often referred to as MAC(K,M)

Correctness:

For every K in the key space, and every M in the message space:

$V(K,M,S(K,M))=1$.

Security: ??

Attacker Power:

Chosen message attack:

Attacker can obtain valid tags for any messages of his choice: M1, M2,...

A common real word attack: The attacker sends Alice emails of his choice. Alice will store these emails on her disc, but will tag them first. Then the attacker can steal her disc.

Attacker Goal:

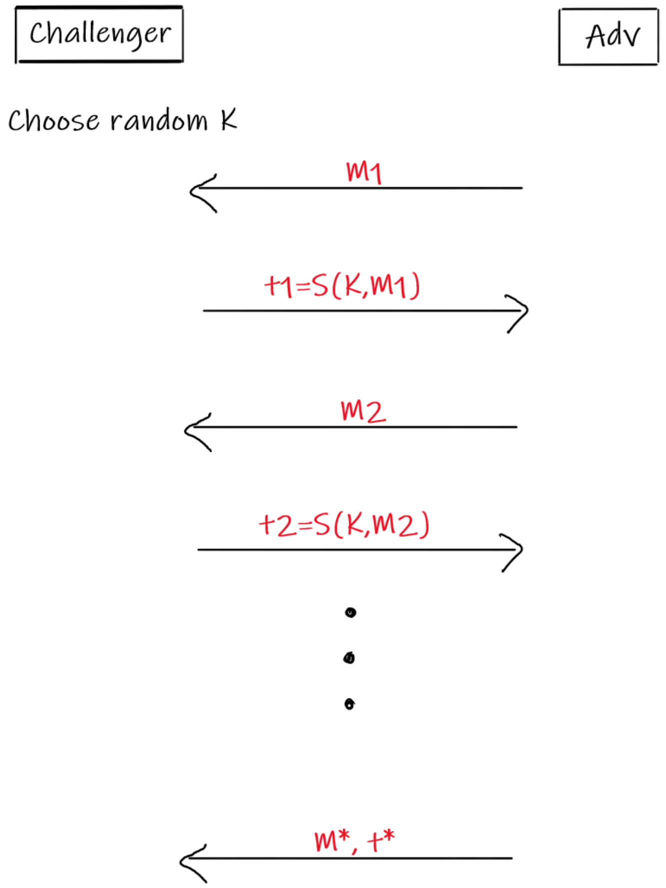
Existential forgery:

An attacker who is given tags $t1=S(K,M1)$, $t2=S(k,M2)$,... for messages M1, M2,... of his choice cannot produce a valid tag for any new message M*.

Note: Adv wins even if M* is gibberish.

This can still be devastating, since sometimes parties MAC a secret key, which is gibberish.

Security as a game:



Adv wins if m^* is different from $m_1, m_2, \ldots$ and $V(K, m^*, t^*)=1$

Strong security: Adv wins if (m^*, t^*) is different from (m_i, t_i) for every i, and $V(K, m^*, t^*)=1$

(strong)

Def: The MAC is secure (existentially secure against adaptive chosen message attacks)

if any efficient adversary wins in this game with negligible probability

How do we construct a MAC??

May seem impossible!

How can we use a single fixed size secret K, to generate more and more unpredictible tags?

Moreover, the MACs used in practice generate random looking tags!

How can we take a fixed size random secret K, and deterministically generate more and more randomness???

Impossible!

We cannot generate randomness "out of thin air"!

Magically: We can generate "pseudo randomness" out of hardness!

Pseudo-Random Functions (PRF):

Informal Definition: A function F is pseudo-random if for a random secret K,

for any (adaptively chosen) inputs x1, x2,... F(K,x1), F(K,x2),... all "look random"

Formal Definition (using a security game):

A function F is a PRF if any efficient adversary A wins in the following game with probability at most 1/2+negligible:

Security as a game:

Challenger		Adv
Choose random bit b.	$\xleftarrow{x1}$	
If b=0 let F be a truly random function.	$\xrightarrow{F(x1)}$	
If b=1, let F be a PRF with a random key K.	$\xleftarrow{x2}$	
	$\xrightarrow{F(x2)}$	
	⋮	
	$\xleftarrow{b^*}$	

Adv wins if $b^*=b$

Theorem: There exists a PRF assuming one-way functions exist.

Definition: A one-way function (OWF) is a function that is easy to compute but hard to invert.

Go to 6.875 if you are interested in the (beautiful!) proof of this theorem

PRF in practice: AES

Advanced Encryption Standard. This is a block cipher (which we will talk about later in this course)

Go to 6.857 if you are interested in the details of the AES construction.

AES is a keyed function that takes 128 bit input to 128 bit output.

Key size has three options: 128, 192 or 256 bits.

AES is assumed to be a PRF.

Actually, as we will see later in this course, AES is a permutation, and hence assumed to be a pseudorandom permutation (PRP)

Question: Is every PRF F with domain D also a secure MAC for messages from D, where the tag of M is F(K,M)?

No!

Note: tag cannot be too small. If tag is only 4 bits, the MAC cannot be secure!

If we think of 2^{-128} as negligible, then tag needs to be at least 128 bits long.

Theorem: Every PRF with domain D and range R where 1/|R| is "negliglbe" is a MAC for messages from D.

Corollary: AES is a secure MAC for messages of length 128 bits.

Question: How can we MAC messages of arbitrary length?

Going from small MAC to big MAC:

This problem should sound familiar:

Recall the construction of a collision resistant hash function (CRHF),

where we started with SHA2 and extended it using a tree construction. ⟵ small CRHF to big CRHF

One approach: Use CRHF!

$S(K,M) = F(K,H(M))$

where F is a PRF (such as AES) and H is a CRHF

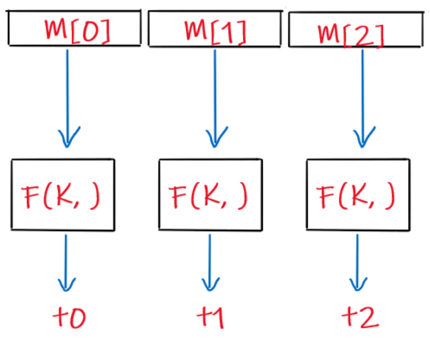
This is secure!

Problem: We need a hash function that outputs 128 bits, but hash functions in practice outputs 256 bits.

This is due to the birthday paradox, which allows finding collisions in time sqrt of the range size.

AES-based MAC (There is also a hash-based MAC called HMAC)

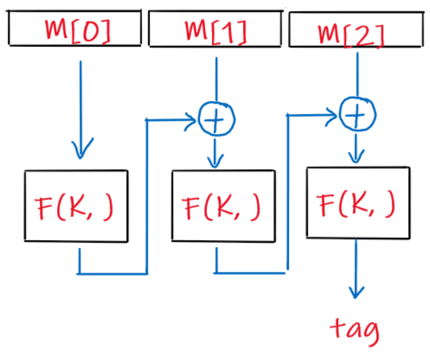
Try 1:



Insecure!

Adversary can use the tag for message (M[0],M[1]) to tag the message (M[1],M[0]).

Try 2:



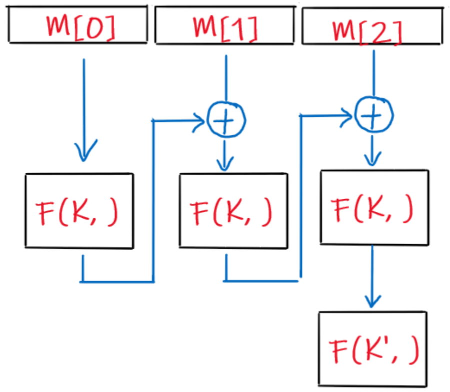
Insecure! Adversary can use the tag for message (M[0],M[1],m[2]) and tag' for message m'[0],

to tag the message (M[0].M[1],M[2],tag xor M'[0]).

Final try:

CBC MAC:

Cipher Block Chaning



The secret key is (K,K')

This additional secret key prohibits these "extension attacks"

Similar to why adding the message length in the construction of a collision resistant hash function is needed to make it secure.

Note: Need to pad the message so that its length will be a multiple of 128.

This is a HW problem (in Pset 1).

The standardized version of CBC MAC is called CMAC