# CSC 580
# Cryptography and Computer Security

*Message Authentication Codes*
*(Sections 12.1-12.5)*

March 29, 2018

## Overview

Today:

- Quiz over HW7 material
- Discuss message authentication codes

Next:

- Complete ungraded HW 8
- Read Chapter 12.7-12.9
- ***Project Progress Report due Tuesday!***

## Message Authentication Requirements
### From Textbook, Section 12.1

Attacks on network communication include

1. Disclosure
2. Traffic analysis

*Confidentiality issues* (1–2)

3. Masquerade
4. Content modification
5. Sequence modification
6. Timing modification (incl replay)

*Message Authentication* (3–6)

7. Source repudiation
8. Destination repudiation

*Digital Signatures* (7)

## Message Authentication Requirements
### From Textbook, Section 12.1

Attacks on network communication include

1. Disclosure
2. Traffic analysis

} *Confidentiality issues*

3. Masquerade
4. Content modification
5. Sequence modification
6. Timing modification (incl replay)

} *Message Authentication*

7. Source repudiation ← *Digital Signatures*
8. Destination repudiation

Basics: Message authentication is a procedure to verify that received messages come from the alleged source and have not been altered. (By including tamper-proof sequence numbers and timestamps, can protect other properties.)

## Using Symmetric Encryption

Consider using a non-malleable cipher

If decryption is "sensible" then most likely:

- Message wasn't tampered with (non-malleable)
- Source was desired sender (only they know the key)

*Problem*: What does "sensible" decryption mean?

*And what if message can be arbitrary binary data?*

Can add some structure or redundancy and look for on decryption

But -- is there a more direct solution?

## Authenticator: Concept

| *Message* | *Authenticator* |
|---|---|
| Send the army to … leaving at 10:30am. | 7c91ad850b513 |

Authenticator computed from message
Message and authenticator both transmitted
Receiver recomputes from message - must match!

*Question*: Will a cryptographic hash function work?

*Specifically: How is this related to second preimage resistance?*

## Authenticator: Concept

| *Message* | *Authenticator* |
|---|---|
| Send the army to … leaving at 10:30am. | 7c91ad850b513 |

Authenticator computed from message
Message and authenticator both transmitted
Receiver recomputes from message - must match!

*Question*: Will a cryptographic hash function work?
*Specifically: How is this related to second preimage resistance?*

Attacker can't replace message, using *same* authenticator

*But*: if authenticator is a known hash function, can compute a new authenticator and replace the original.


## Authenticator: Concept

| *Message* | *Authenticator* |
|---|---|
| Send the army to … leaving at 10:30am. | 7c91ad850b513 |

Authenticator computed from message
Message and authenticator both transmitted
Receiver recomputes from message - must match!

*Question*: Will a cryptographic hash function work?
*Specifically: How is this related to second preimage resistance?*

Attacker can't replace message, using *same* authenticator

*But*: if authenticator is a known hash function, can compute a new authenticator and replace the original.


Sender and receiver share secret → Then attacker can't compute!
*If only sender and receiver know secret, authenticates source too*

## Message Authentication Codes

A first, naive attempt:

For message made of up n blocks $M_1, M_2, \ldots, M_n$:

1. Calculate $S = M_1 \oplus M_2 \oplus \ldots \oplus M_n$
2. Calculate tag $T = \mathrm{E}(K,S)$ using a non-malleable cipher

*Question 1*: Can you find *any* other message with same tag?

## Message Authentication Codes

A first, naive attempt:

For message made of up n blocks $M_1, M_2, \ldots, M_n$:

1. Calculate $S = M_1 \oplus M_2 \oplus \ldots \oplus M_n$
2. Calculate tag $T = \mathrm{E}(K,S)$ using a non-malleable cipher

*Question 1*: Can you find *any* other message with same tag?

*XOR is commutative and associative, so just rearrange blocks*

*Question 2*: Can you construct a message mostly of your own choosing with the same tag?

## Message Authentication Codes

A first, naive attempt:

For message made of up n blocks $M_1, M_2, \ldots, M_n$:

1. Calculate $S = M_1 \oplus M_2 \oplus \ldots \oplus M_n$
2. Calculate tag $T = \mathrm{E}(K,S)$ using a non-malleable cipher

*Question 1*: Can you find *any* other message with same tag?

*XOR is commutative and associative, so just rearrange blocks*

*Question 2*: Can you construct a message mostly of your own choosing with the same tag?

For any n-1 block forgery $F_1, F_2, \ldots, F_{n-1}$, compute
$F_n = F_1 \oplus F_2 \oplus \ldots \oplus F_{n-1} \oplus S$,
so $F_1 \oplus F_2 \oplus \ldots \oplus F_{n-1} \oplus F_n = S$

## Message Authentication Codes

Function MAC: $\mathcal{K} \times \mathcal{M} \rightarrow \{0,1\}^h$

*Keyspace* ($\mathcal{K}$), *Message space* ($\mathcal{M}$), *Authenticator (or "tag")* ($\{0,1\}^h$)

Important properties:

- Given M and T = MAC(K,M), can't find M' with MAC(K,M') = MAC(K,M)
  - *Like second preimage resistance*
- Given M and MAC(K,M), can't calculate K
  - *Similar to preimage resistance (one-way)*
  - *Brute force attack takes time $|\mathcal{K}|/2$ on average*
- Given M and T = MAC(K,M), can't find M' and T' s.t. T'=MAC(K,M')

So… was sent by someone who knows K, and M hasn't been tampered with

## Formal Security of MACs

Consider: What is best algorithm to take a set of message/tag pairs, generated with an unknown key K:

$\{ (M_1, MAC(K,M_1)) , (M_2, MAC(K,M_2)), \ldots , (M_n, MAC(K,M_n)) \}$

*Security challenge*: Find a pair ($M$, $T$) where

1. $M \notin \{M_1, M_2, \ldots, M_n\}$ (i.e., M hasn't been seen before)
2. $T = MAC(K, M)$

($M$,$T$) is called a forgery

In a real attack, probably want $M$ to be chosen or at least meaningful

In formal model, tilt advantage toward attacker: $M$ can be anything

- This is called an *existential forgery*
- A MAC that is secure against this is called *existentially unforgeable*

## Formal Security of MACs

*Next*: Where does the set of known message/tag pairs come from?

Some options:

- Provided or random messages (think: captured communications)
- Attacker picks all $n$ messages $M_1$, $M_2$, …, $M_n$ then gets all tags
- Attacker picks $M_1$ and gets $T_1$, then picks $M_2$ and gets $T_2$, etc.

Each option gives attacker more power than previous option.

Design against strongest possible adversary - the last option

- This is called an *adaptive chosen message attack*
- So best possible goal: *existential unforgeability against adaptive chosen message attack* (*EUF-CMA*)
- Note: More commonly used as security goal for signatures, but same idea

## Making a MAC from a Hash Function

### Insecure first attempt

*Idea*: Need a hash function with a secret key, so start with a standard hash function

Attempt 1 - Insecure

(but a lot of people do this anyway - don't be one of those people)

Idea: Concatenate key and message, and hash: $T = H(K \| M)$

Can't figure out key if H is preimage resistant. Can't pick different M (for same T) if H is collision resistant.

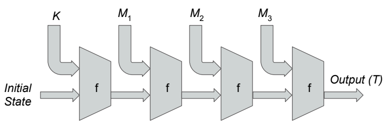
*So… what's the problem?*

# Making a MAC from a Hash Function

## Insecure first attempt

Recall Merkle-Damgard hash structure - 3 block example

(used by SHA1, SHA2 family (SHA256, SHA512, etc.)

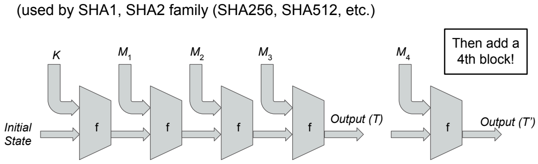



# Making a MAC from a Hash Function

## Insecure first attempt

Recall Merkle-Damgard hash structure - 3 block example

(used by SHA1, SHA2 family (SHA256, SHA512, etc.)

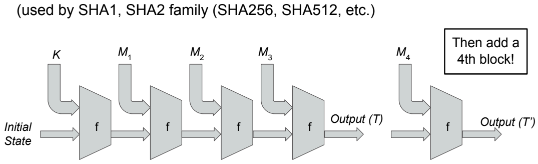



# Making a MAC from a Hash Function

## Insecure first attempt

Recall Merkle-Damgard hash structure - 3 block example

(used by SHA1, SHA2 family (SHA256, SHA512, etc.)




So: Given $M_1$, $M_2$, $M_3$, and T = MAC($K$,$M_1$||$M_2$||$M_3$)

➔ Can pick $M_4$ and compute $T'$ = f($T$, $M_4$) = MAC($K$,$M_1$||$M_2$||$M_3$||$M_4$) - forgery!

This is called an ***extension attack***

- Problem with any Merkle-Damgard hash function used this way
- Is not problem with SHA3!

## HMAC - The Right Way

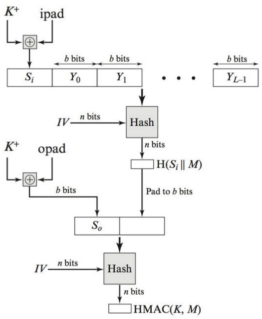



**Figure 12.5** HMAC Structure

*Key point*:

Don't know H($S_i$ || *M*) so can't extend message!

## HMAC - Proven Security!

*Theorem (informally stated)*: If H is a Merkle-Damgard style hash function in which the compression function is a pseudorandom function (PRF), then HMAC using H is a pseudorandom function.

Proved in: Mihir Bellare. "New Proofs for NMAC and HMAC: Security without Collision-Resistance," *2006 Conference on Advances in Cryptology* (*CRYPTO '06*).