Overview

Today:
- Quiz over HW9 material
- Discuss message authentication codes

Next:
- Complete Homework 10
- Read Chapter 12.7-12.9

Message Authentication Requirements

From Textbook, Section 12.1

Attacks on network communication include

1. Disclosure
2. Traffic analysis
3. Masquerade
4. Content modification
5. Sequence modification
6. Timing modification (and replay)
7. Source repudiation
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

<table>
<thead>
<tr>
<th>Message</th>
<th>Authenticator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send the army to ... leaving at 10:30am.</td>
<td>7c91ad850b513</td>
</tr>
</tbody>
</table>

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 \(T = E(K, S)\) 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: \( K \times \{0,1\}^* \rightarrow (0,1)^h \)

- **Keyspace**
- **Message space**
- **Authenticator (or "tag")**

Important properties:
- Given \( M \) and \( T = MAC(K,M) \), can’t find \( M' \) with \( MAC(K,M') = MAC(K,M) \)
  - Similar to preimage resistance
  - Brute force attack takes time \(|K|/2\) on average
- Given \( M \) and \( MAC(K,M) \), can’t calculate \( K \)
  - Similar to preimage resistance (one-way)
- 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 \( M_1, M_2, \ldots , M_n \) then gets all tags
- Attacker picks \( M_i \) and gets \( T_i \), then picks \( M_j \) and gets \( T_j \), 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: **existentially 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 if H is collision resistant. So… what’s the problem?

Recall Merkle-Damgard hash structure - 3 block example
(used by SHA1, SHA2 family (SHA256, SHA512, etc.)

\[
\begin{array}{c}
\text{Initial State} \\
K \\
M_1 \\
M_2 \\
M_3 \\
\text{Output (T)}
\end{array}
\]

Then add a 4th block!

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

\( \Rightarrow \) Can pick \( M_4 \) and compute \( T' = H(T \ || \ M_4) = MAC(K, M_1 || M_2 || M_3 || M_4) \) - 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

\[
\begin{array}{c}
\text{Key point:} \\
\text{Don’t know } H(S \ || \ M) \text{ so can’t extend message!}
\end{array}
\]
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.