A Fixed-Length Message Authentication Code (MAC)
Authentication

Alice sends Bob a message (likely in code, but that will not concern us)

Or does she?
Authentication

Alice sends Bob a message (likely in code, but that will not concern us)

Or does she?

Maybe it was Eve!
Authentication

Alice sends Bob a message (likely in code, but that will not concern us)

Or does she?

Maybe it was Eve!
We need Alice to be able to Authenticate that the message really came from Bob (and vice versa).
Formal MAC

We define a message Authentication Code (MAC).

Def: A MAC is \( \Pi = (GEN, MAC(m), V(m, t)) \) where Security parameter \( n \)

- \( GEN \) chooses a uniform key \( k \in \{0, 1\}^n \).
- \( MAC : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n \). Use as \( MAC_k(m) \). Often say \( t = MAC_k(m) \) and call \( t \) the tag. Note: Think of \( m \) as an ENCODED message even though we call it \( m \). We are not concerned with how it was encoded, only with authentication that it came from Bob.
- \( V : \{0, 1\}^n \times \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\} \). Use as \( V_k(m, t) \).
- Require that if \( MAC_k(m) = t \) then \( V_k(m, t) = 1 \) (Yes)
- \( MAC \) is PPT, \( V \) is P.

How to Use: Alice and Bob have \( \Pi = (GEN, MAC, V) \).

1. Alice generates \( k \) via \( GEN \) and sends it to Bob privately.
2. Bob wants to send \( m \in bits^* \) to Alice. He computes \( t = MAC_k(m) \) and sends

\[
(ENC_k(m), t)
\]
Formal Def of MAC being Secure

We actually defined security of the pair \((\Pi_1, \Pi_2)\) where 
\[ \Pi_1 = (\text{GEN}_1, \text{ENC}, \text{DEC}) \] and 
\[ \Pi_2 = (\text{GEN}_2, \text{MAC}, \text{V}). \]
We actually defined security of the pair \((\Pi_1, \Pi_2)\) where \(\Pi_1 = (GEN_1, ENC, DEC)\) and \(\Pi_2 = (GEN_2, MAC, V)\).
You will do this on a HW.

**Hint1:** Definition is via a Game

**Hint2:** Alice has \(k \in \{0, 1\}^n\), key, which Eve does NOT have.

**Hint3:** Eve’s goal is to produce SOME \((m, t)\) such that \(V_k(m, t) = 1\).

**Hint4:** Eve has black-box access to \(MAC_k\) but is not allowed to, for some \(m\), ask \(MAC_k(m)\) and then output \((m, MAC_k(m))\).
Example of a MAC

Messages are in \{0, \ldots, p - 1\}.

1. \( k \in \{0, \ldots, p - 1\} \) unif.
2. \( MAC_k(m) = m + k \)
3. \( V_k(m, t) = 1 \) if \( t = m + k \).

Not Secure: Eve can find \( MAC_k(0) = 7 \) and then know \( k = 7 \).
Example of a MAC

Messages are in \( \{0, 1\}^n \).

1. \( k \in \{0, 1\}^n \) unif.
2. \( MAC_k(m) = m \oplus k \)
3. \( V_k(m, t) = 1 \) if \( t = m \oplus k \).

VOTE: Secure, NOT Secure, Depends on HA, UK:
Example of a MAC

Messages are in \( \{0, 1\}^n \).

1. \( k \in \{0, 1\}^n \) unif.
2. \( MAC_k(m) = m \oplus k \)
3. \( V_k(m, t) = 1 \) if \( t = m \oplus k \).

**VOTE:** Secure, NOT Secure, Depends on HA, UK:
Not Secure: Eve can find \( MAC_k(0) = 0011010 \) and then know \( k = 0011010 \).
Example of a MAC

Messages are in \( \{0, 1\}^n \).

1. \( k \in \{0, 1\}^n \) unif.
2. \( MAC_k(m) = m \oplus k \)
3. \( V_k(m, t) = 1 \) if \( t = m \oplus k \).

**VOTE:** Secure, NOT Secure, Depends on HA, UK:
**Not Secure:** Eve can find \( MAC_k(0) = 0011010 \) and then know \( k = 0011010 \).
**Need:** A function \( f_k \) such that \( f_k(0) \) does not give away \( k \).
Example of a MAC

Messages are in \( \{0, 1\}^n \).

1. \( k \in \{0, 1\}^n \) unif.
2. \( MAC_k(m) = m \oplus k \)
3. \( V_k(m, t) = 1 \) if \( t = m \oplus k \).

**VOTE:** Secure, NOT Secure, Depends on HA, UK:
**Not Secure:** Eve can find \( MAC_k(0) = 0011010 \) and then know \( k = 0011010 \).
**Need:** A function \( f_k \) such that \( f_k(0) \) does not give away \( k \).
Psuedo-Random Functions!
Construction

Let $F$ be a length preserving pseudorandom-function (aka block cipher)
Construct the following MAC $\Pi$:

1. $Gen$: choose a uniform key $k$ for $F$
2. $Mac_k(m)$: output $F_k(m)$
3. $Vrfy_k(m, t)$: output 1 iff $F_k(m) = t$

Theorem: $\Pi$ is a secure MAC
Proof Sketch: If not secure then $F_k$ would not be pseudorandom.
Drawbacks?

- This only works for *fixed-length* messages
- This only works for *short* messages
  - E.g. AES has a 128-bit block size (shorter than a tweet!)
- So the previous construction is limited to authenticating short, fixed-length messages

**Note:** There are extensions that deal with these issues. We omit.