Message Authentication

Adam O’Neill
Based on http://cseweb.ucsd.edu/~mihir/cse207/
Authenticity and Integrity

- Message actually comes from claimed sender
- Message was not modified in transit
Electronic Banking

Users want that Henrik actually sent this message, and was not modified.
Message Authentication Codes

A message authentication code $\mathcal{T} : \text{Keys}(\mathcal{T}) \times \text{Dom}(\mathcal{T}) \to \text{Range}(\mathcal{T})$ is a family of functions. The envisaged usage is shown below, where $A$ is the adversary:
MAC Usage
Let $\mathcal{T}$: $\text{Keys}(\mathcal{T}) \times \text{Dom}(\mathcal{T}) \rightarrow \text{Range}(\mathcal{T})$ be a message authentication code. Let $A$ be an adversary.

The uf-cma advantage of adversary $A$ is

$$\text{Adv}^{\text{uf-cma}}_T(A) = \Pr \left[ \text{UFCMA}_A^T \Rightarrow \text{true} \right]$$
Explaination

Adversary $A$ does not get the key $K$.

It can call $\text{Tag}$ with any message $M$ of its choice to get back the correct tag $T = T_K(M)$.

To win, the adversary $A$ must output a message $M \in \text{Dom}(T)$ and a tag $T$ that are

- Correct: $T = T_K(M)$
- New: $M \not\in S$, meaning $M$ was not a query to $\text{Tag}$

**Interpretation:** $\text{Tag}$ represents the sender and $\text{Finalize}$ represents the receiver. Security means that the adversary can't get the receiver to accept a message that is not authentic, meaning was not already transmitted by the sender.
Replay

Suppose Alice transmits \((M_1, T_1)\) to Bank where \(M_1 = \text{“Pay $100 to Bob”}\). Adversary

- Captures \((M_1, T_1)\)
- Keeps re-transmitting it to bank

Result: Bob gets $100, $200, $300,...

Our UF-CMA notion of security does not ask for protection against replay, because \(A\) will not win if it outputs \(M, T\) with \(M \in S\), even if \(T = T_K(M)\) is the correct tag.

**Question:** Why not?

**Answer:** Replay is best addressed as an add-on to standard message authentication.
Timestamps

Let $Time_A$ be the time as per Alice’s local clock and $Time_B$ the time as per Bob’s local clock.

- Alice sends $(M, T_K(M), Time_A)$
- Bob receives $(M, T, Time)$ and accepts iff $T = T_K(M)$ and $|Time_B - Time| \leq \Delta$ where $\Delta$ is a small threshold.

Does this work?

You can change the time?
Timestamps

Let $Time_A$ be the time as per Alice’s local clock and $Time_B$ the time as per Bob’s local clock.

- Alice sends $(M, T_K(M \parallel Time_A), Time_A)$
- Bob receives $(M, T, Time)$ and accepts iff $T_K(M \parallel Time) = T$ and $|Time_B - Time| \leq \Delta$ where $\Delta$ is a small threshold.
Counters

Alice maintains a counter $ctr_A$ and Bob maintains a counter $ctr_B$. Initially both are zero.

- Alice sends $(M, T_K(M || ctr_A))$ and then increments $ctr_A$
- Bob receives $(M, T)$. If $T_K(M || ctr_B) = T$ then Bob accepts and increments $ctr_B$.

Counters need to stay synchronized.
Unconditional MAC

**Attack Game 7.5 (pairwise unpredictability).** For a keyed hash function $H$ defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, and a given adversary $A$, the attack game runs as follows.

- The challenger picks a random $k \leftarrow \mathcal{K}$ and keeps $k$ to itself.
- $A$ sends a message $m_0 \in \mathcal{M}$ to the challenger, who responds with $t_0 = H(k, m_0)$.
- $A$ outputs $(m_1, t_1) \in \mathcal{M} \times \mathcal{T}$, where $m_1 \neq m_0$.

We say that $A$ wins the game if $t_1 = H(k, m_1)$. We define $A$’s advantage with respect to $H$, denoted $\text{PUF}_{\text{adv}}[A, H]$, as the probability that $A$ wins the game. □

**Definition 7.7.** We say that $H$ is an $\epsilon$-bounded pairwise unpredictable function, or $\epsilon$-PUF for short, if $\text{PUF}_{\text{adv}}[A, H] \leq \epsilon$ for all adversaries $A$ (even inefficient ones).
If $F$ is PRF-secure then it is also UF-CMA-secure:

**Theorem [GGM86,BKR96]:** Let $F : \{0, 1\}^k \times D \rightarrow \{0, 1\}^n$ be a family of functions. Let $A$ be a uf-cma adversary making $q$ \textbf{Tag} queries and having running time $t$. Then there is a prf-adversary $B$ such that

$$\text{Adv}_{F}^{uf-cma}(A) \leq \text{Adv}_{F}^{prf}(B) + \frac{2}{2^n}.$$

Adversary $B$ makes $q + 1$ queries to its $F_n$ oracle and has running time $t$ plus some overhead.
Proof

- Switch to truly random function in the game by PRP security

- Probability that adversary predicts the value of the function at an unknown point is $\frac{1}{2^{k+1}}$
Variable-input-length (PRF domain extension)

A family of functions $F: \text{Keys}(F) \times D \rightarrow R$ is

- **FIL (Fixed-input-length)** if $D = \{0, 1\}^\ell$ for some $\ell$
- **VIL (Variable-input-length)** if $D$ is a “large” set like $D = \{0, 1\}^*$ or $D = \{ M \in \{0, 1\}^* : 0 < |M| < n2^n \text{ and } |M| \mod n = 0 \}$ for some $n \geq 1$ or ...

We have families we are willing to assume are PRFs, namely blockciphers and compression functions, but they are FIL.

**PRF Domain Extension Problem:** Given a FIL PRF, construct a VIL PRF.
CBC MAC

Let $E : \{0, 1\}^k \times B \rightarrow B$ be a blockcipher, where $B = \{0, 1\}^n$. View a message $M \in B^*$ as a sequence of $n$-bit blocks, $M = M[1] \ldots M[m]$.

The basic CBC MAC $T : \{0, 1\}^k \times B^* \rightarrow B$ is defined by

\[
\text{Alg } T_K(M) \\
C[0] \leftarrow 0^n \\
\text{for } i = 1, \ldots, m \text{ do } C[i] \leftarrow E_K(C[i-1] \oplus M[i]) \\
\text{return } C[m]
\]
Splicing Attack

**Algorithm** $T_K(M)$

- $C[0] \leftarrow 0^n$
- for $i = 1, \ldots, m$ do
  - $C[i] \leftarrow E_K(C[i - 1] \oplus M[i])$
- return $C[m]$ 

**Adversary** $A$

- Let $x \in \{0, 1\}^n$
- $T_1 \leftarrow \text{Tag}(x)$
- $M \leftarrow x||T_1 \oplus x$
- Return $M, T_1$

---

prefix-free encoding
Solutions

The basic CBC MAC is a candidate construction but we saw above that it fails to be UF-CMA and thus also fails to be a PRF. The exercises explored other solutions.

We will see solutions that work including

- ECBC: The encrypted CBC-MAC
- CMAC: A NIST standard
- HMAC: A highly standardized and used hash-function based MAC
How long does a tag have to be?

For $l$-bits of UF-CMA security, how long do the tags need to be?

Claim. They need to be $\geq 2^l$ bits.

Proof. Consider the adversary that picks a tag at random. It has advantage $\geq 2^{-l}$. 
Birthday Attack

There is a large class of MACs, including ECBC MAC, CMAC, HMAC, ... which are subject to a birthday attack that violates UF-CMA using about \( q \approx 2^{n/2} \) Tag queries, where \( n \) is the tag (output) length of the MAC.

Furthermore, we can typically show this is best possible, so the birthday bound is the “true” indication of security.

The class of MACs in question are called iterated-MACs and work by iterating some lower level primitive such as a blockcipher or compression function.
ECBC MAC

Let \( E : \{0, 1\}^k \times B \rightarrow B \) be a block cipher, where \( B = \{0, 1\}^n \). The encrypted CBC (ECBC) MAC \( T : \{0, 1\}^{2k} \times B^* \rightarrow B \) is defined by

\[
\text{Alg} \ T_{K_{in}||K_{out}}(M)
\]
\[
C[0] \leftarrow 0^n
\]
for \( i = 1, \ldots, m \) do
\[
C[i] \leftarrow E_{K_{in}}(C[i - 1] \oplus M[i])
\]
\( T \leftarrow E_{K_{out}}(C[m]) \)
return \( T \)

\[
E_{K_{in}}(M_i) = E_{K_{in}}(C_{m-1})
\]
Birthday Attack

Let $E: \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a blockcipher and $\mathcal{T}$ the corresponding ECBC MAC. Let $q \leq 2^{n/2}$.

Give an adversary $A$ that, via a birthday attack, achieves

$$\text{Adv}_{\mathcal{T}}^{\text{uf-cma}}(A) = \Omega \left( \frac{q^2}{2^n} \right)$$

using $q$ Tag queries and running time $O(nq \cdot \log(nq))$. 
Security Theorem

**Theorem:** Let $E : \{0, 1\}^k \times B \rightarrow B$ be a family of functions, where $B = \{0, 1\}^n$. Define $F : \{0, 1\}^{2k} \times B^* \rightarrow \{0, 1\}^n$ by

\[\text{Alg } F_{K_{\text{in}}||K_{\text{out}}}(M)\]

\[C[0] \leftarrow 0^n\]

for $i = 1, \ldots, m$ do $C[i] \leftarrow E_{K_{\text{in}}}(C[i-1] \oplus M[i])$

$T \leftarrow E_{K_{\text{out}}}(C[m])$; return $T$

Let $A$ be a prf-adversary against $F$ that makes at most $q$ oracle queries, these totalling at most $\sigma$ blocks, and has running time $t$. Then there is a prf-adversary $B$ against $E$ such that

\[\text{Adv}_{F_{\text{prf}}}^E(A) \leq \text{Adv}_{E_{\text{prf}}}^E(B) + \frac{\sigma^2}{2^n}\]

and $B$ makes at most $\sigma$ oracle queries and has running time about $t$. 
Proof
On the $s^{th}$ query $F(M^s)$  

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>$P \leftarrow \text{Prefix}(M^1, \ldots, M^s)$</td>
</tr>
<tr>
<td>101</td>
<td>$C \leftarrow \Upsilon[P]$</td>
</tr>
<tr>
<td>102</td>
<td>for $j \leftarrow |P|_n + 1$ to $m$ do</td>
</tr>
<tr>
<td>103</td>
<td>$X \leftarrow C \oplus M^s_j$</td>
</tr>
<tr>
<td>104</td>
<td>$C \leftarrow {0, 1}^n$</td>
</tr>
<tr>
<td>105</td>
<td>if $C \in \text{Range}(\pi)$ then $\text{bad} \leftarrow \text{true}$, $C \leftarrow \text{Range}(\pi)$</td>
</tr>
<tr>
<td>106</td>
<td>if $X \in \text{Domain}(\pi)$ then $\text{bad} \leftarrow \text{true}$, $C \leftarrow \pi(X)$</td>
</tr>
<tr>
<td>107</td>
<td>$\pi(X) \leftarrow C$</td>
</tr>
<tr>
<td>108</td>
<td>$\Upsilon[M^s_{1\rightarrow j}] \leftarrow C$</td>
</tr>
<tr>
<td>109</td>
<td>return $C$</td>
</tr>
</tbody>
</table>

On the $s^{th}$ query $F(M^s)$  

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>$P \leftarrow \text{Prefix}(M^1, \ldots, M^s)$</td>
</tr>
<tr>
<td>201</td>
<td>$C \leftarrow \Upsilon[P]$</td>
</tr>
<tr>
<td>202</td>
<td>for $j \leftarrow |P|_n + 1$ to $m$ do</td>
</tr>
<tr>
<td>203</td>
<td>$X \leftarrow C \oplus M^s_j$</td>
</tr>
<tr>
<td>204</td>
<td>$C \leftarrow {0, 1}^n$</td>
</tr>
<tr>
<td>205</td>
<td>if $X \in \text{Domain}(\pi)$ then $\text{bad} \leftarrow \text{true}$</td>
</tr>
<tr>
<td>206</td>
<td>$\pi(X) \leftarrow C$</td>
</tr>
<tr>
<td>207</td>
<td>$\Upsilon[M^s_{1\rightarrow j}] \leftarrow C$</td>
</tr>
<tr>
<td>208</td>
<td>return $C$</td>
</tr>
</tbody>
</table>

On the $s^{th}$ query $F(M^s)$  

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>$P \leftarrow \text{Prefix}(M^1, \ldots, M^s)$</td>
</tr>
<tr>
<td>301</td>
<td>$C \leftarrow \Upsilon[P]$</td>
</tr>
<tr>
<td>302</td>
<td>for $j \leftarrow |P|_n + 1$ to $m$ do</td>
</tr>
<tr>
<td>303</td>
<td>$X \leftarrow C \oplus M^s_j$</td>
</tr>
<tr>
<td>304</td>
<td>$C \leftarrow {0, 1}^n$</td>
</tr>
<tr>
<td>305</td>
<td>if $X \in \text{Domain}(\pi)$ then $\text{bad} \leftarrow \text{true}$</td>
</tr>
<tr>
<td>306</td>
<td>$\pi(X) \leftarrow \text{defined}$</td>
</tr>
<tr>
<td>307</td>
<td>$\Upsilon[M^s_{1\rightarrow j}] \leftarrow C$</td>
</tr>
<tr>
<td>308</td>
<td>return $C$</td>
</tr>
</tbody>
</table>

On the $s^{th}$ query $F(M^s)$  

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>$P \leftarrow \text{Prefix}(M^1, \ldots, M^s)$</td>
</tr>
<tr>
<td>401</td>
<td>$C \leftarrow \Upsilon[P]$</td>
</tr>
<tr>
<td>402</td>
<td>for $j \leftarrow |P|_n + 1$ to $m$ do</td>
</tr>
<tr>
<td>403</td>
<td>$X \leftarrow C \oplus M^s_j$</td>
</tr>
<tr>
<td>404</td>
<td>if $X \in \text{Domain}(\pi)$ then $\text{bad} \leftarrow \text{true}$</td>
</tr>
<tr>
<td>405</td>
<td>$\pi(X) \leftarrow \text{defined}$</td>
</tr>
<tr>
<td>406</td>
<td>$C \leftarrow \Upsilon[M^s_{1\rightarrow j}] \leftarrow {0, 1}^n$</td>
</tr>
<tr>
<td>407</td>
<td>$Z^s \leftarrow {0, 1}^n$</td>
</tr>
<tr>
<td>408</td>
<td>return $Z^s$</td>
</tr>
</tbody>
</table>

Figure 7.6: Games used in the CBC MAC analysis. Let $\text{Prefix}(M^1, \ldots, M^s)$ be $\varepsilon$ if $s = 1$, else the longest string $P \in (\{0, 1\}^n)^*$ s.t. $P$ is a prefix of $M^s$ and $M^r$ for some $r < s$. In each game, Initialize sets $\Upsilon[\varepsilon] \leftarrow 0^n$. 
Initialize

Figure 7.6: Games used in the CBC MAC analysis. Let

Bellare and Rogaway

for

\( X \leftarrow C \oplus M_j^s \)

if \( X \in \text{Domain}(\pi) \) then \( \text{bad} \leftarrow \text{true} \)

\( \pi(X) \leftarrow \text{defined} \)

\( C \leftarrow \mathcal{Y}[M_{1 \rightarrow j}] \leftarrow \{0, 1\}^n \)

Game C5

600 for \( s \leftarrow 1 \) to \( q \) do

601 \( P^s \leftarrow \text{Prefix}(M^1, \ldots, M^s) \)

602 \( C \leftarrow \mathcal{Y}[P^s] \)

603 \( X \leftarrow C \oplus M_j^s \) \( ||P^s||_n + 1 \)

604 if \( X \in \text{Domain}(\pi) \) then \( \text{bad} \leftarrow \text{true} \)

605 \( \pi(X) \leftarrow \text{defined} \)

606 \( C \leftarrow \mathcal{Y}[M_{1 \rightarrow j}^s \cup P^s \cup j] \leftarrow \{0, 1\}^n \)

607 for \( j \leftarrow ||P^s||_n + 2 \) to \( m \) do

608 \( X \leftarrow C \oplus M_j^s \)

609 if \( X \in \text{Domain}(\pi) \) then \( \text{bad} \leftarrow \text{true} \)

610 \( \pi(X) \leftarrow \text{defined} \)

611 \( C \leftarrow \mathcal{Y}[M_{1 \rightarrow j}^s] \leftarrow \{0, 1\}^n \)

Game C6

700 for \( X \in \{0, 1\}^+ \) do \( \mathcal{Y}[X] \leftarrow \{0, 1\}^n \)

701 for \( s \leftarrow 1 \) to \( q \) do

702 \( P^s \leftarrow \text{Prefix}(M^1, \ldots, M^s) \)

703 if \( \mathcal{Y}[P^s] \cup M_j^s \cup ||P^s||_n + 1 \in \text{Domain}(\pi) \) then \( \text{bad} \leftarrow \text{true} \)

704 \( \pi(\mathcal{Y}[P^s] \cup M_j^s \cup ||P^s||_n + 1) \leftarrow \text{defined} \)

705 for \( j \leftarrow ||P^s||_n + 2 \) to \( m \) do

706 if \( \mathcal{Y}[M_{1 \rightarrow j-1}^s] \cup M_j^s \in \text{Domain}(\pi) \) then \( \text{bad} \leftarrow \text{true} \)

707 \( \pi(\mathcal{Y}[M_{1 \rightarrow j-1}^s] \cup M_j^s) \leftarrow \text{defined} \)

Game C7

800 for \( X \in \{0, 1\}^+ \) do \( \mathcal{Y}[X] \leftarrow \{0, 1\}^n \)

801 for \( s \leftarrow 1 \) to \( q \) do

802 \( P^s \leftarrow \text{Prefix}(M^1, \ldots, M^s) \)

803 if \( \mathcal{Y}[P^s] \cup M_j^s \cup ||P^s||_n + 1 \in \text{Domain}(\pi) \) then \( \text{bad} \leftarrow \text{true} \)

804 \( \pi(\mathcal{Y}[P^s] \cup M_j^s \cup ||P^s||_n + 1) \leftarrow \text{defined} \)

805 for \( j \leftarrow ||P^s||_n + 1 \) to \( m - 1 \) do

806 if \( \mathcal{Y}[M_{1 \rightarrow j-1}^s] \cup M_j^s \in \text{Domain}(\pi) \) then \( \text{bad} \leftarrow \text{true} \)

807 \( \pi(\mathcal{Y}[M_{1 \rightarrow j-1}^s] \cup M_j^s) \leftarrow \text{defined} \)

Game C8

900 for \( X \in \{0, 1\}^+ \) do \( \mathcal{Y}[X] \leftarrow \{0, 1\}^n \)

901 for \( s \leftarrow 1 \) to \( q \) do \( P^s \leftarrow \text{Prefix}(M^1, \ldots, M^s) \)

902 \( \text{bad} \leftarrow \exists(r, i) \neq (s, j) \) \( (r \leq s) \) \( (i \geq ||P^r||_n + 1) \) \( (j \geq ||P^s||_n + 1) \)

903 \( \mathcal{Y}[P^r] \cup M_{r \rightarrow n+1}^i = \mathcal{Y}[P^s] \cup M_{s \rightarrow n+1}^i \) \( \text{and} \) \( r < s \) or

904 \( \mathcal{Y}[M_{1 \rightarrow j-1}] \cup M_{j+1}^i = \mathcal{Y}[P^s] \cup M_{s \rightarrow n+1}^i \) or

905 \( \mathcal{Y}[M_{1 \rightarrow j-1}] \cup M_{j+1}^i = \mathcal{Y}[M_{1 \rightarrow j-1}] \cup M_{j+1}^i \) or

906 \( \mathcal{Y}[P^r] \cup M_{r \rightarrow n+1}^i = \mathcal{Y}[M_{1 \rightarrow j}] \cup M_{j+1}^i \) or

Game C9
Security of Iterated MACs

The number $q$ of $m$-block messages that can be safely authenticated is about $2^{n/2}/m$, where $n$ is the block-length of the blockcipher, or the length of the chaining input of the compression function.

<table>
<thead>
<tr>
<th>MAC</th>
<th>$n$</th>
<th>$m$</th>
<th>$q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES-ECBC-MAC</td>
<td>64</td>
<td>1024</td>
<td>$2^{22}$</td>
</tr>
<tr>
<td>AES-ECBC-MAC</td>
<td>128</td>
<td>1024</td>
<td>$2^{54}$</td>
</tr>
<tr>
<td>AES-ECBC-MAC</td>
<td>128</td>
<td>$10^6$</td>
<td>$2^{44}$</td>
</tr>
<tr>
<td>HMAC-SHA1</td>
<td>160</td>
<td>$10^6$</td>
<td>$2^{60}$</td>
</tr>
<tr>
<td>HMAC-SHA256</td>
<td>256</td>
<td>$10^6$</td>
<td>$2^{108}$</td>
</tr>
</tbody>
</table>

$m = 10^6$ means message length 16Mbytes when $n = 128$. 
Padding

So far we assumed messages have length a multiple of the block-length of the blockcipher. Call such messages *full*. How do we deal with non-full messages?


The obvious approach is padding. But how we pad matters.
Padding

Padding with $10^*$: For a non-full message


For a full message


This works, but if $M$ was full, an extra block is needed leading to an extra blockcipher operation.
Costs

Handling length-variablity and non-full messages leads to two extra blockcipher invocations in ECBC MAC as compared to basic CBC MAC.

Also ECBC uses two blockcipher keys and needs to rekey, which is expensive.

Can we do better?
CMAC

**Standards:** NIST SP 800-38B, RFCs 4493, 4494, 4615

**Features:** Handles variable-length and non-full messages with
- Minimal overhead
- A single blockcipher key

**Security:**
- Subject to a birthday attack
- Security proof shows there is no better attack

**History:** XCBC[BI Ro], OMAC/OMAC1[IW]
Components

- $E : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a blockcipher, in practice AES.
- $\text{CBC}_K(M)$ is the basic CBC MAC of a full message $M$ under key $K \in \{0, 1\}^n$ and using $E$.
- $J \in \{0, 1\}^n$ is a particular fixed constant.

CMAC uses its key $K \in \{0, 1\}^n$ to derive subkeys $K_1, K_2$ via

\[\textbf{Alg CMAC-KEYGEN}(K)\]
\[K_0 \leftarrow E_K(0)\]
if $\text{msb}(K_0) = 0$ then $K_1 \leftarrow (K_0 \ll 1)$ else $K_1 \leftarrow (K_0 \ll 1) \oplus J$
if $\text{msb}(K_1) = 0$ then $K_2 \leftarrow (K_1 \ll 1)$ else $K_2 \leftarrow (K_1 \ll 1) \oplus J$
Return $(K_1, K_2)$

where $x \ll 1$ means $x$ left shifted by 1 bit, so that the msb vanishes and the lsb becomes 0. These bit operations use simple finite-field operations.
Algorithm

Alg $\text{CMAK}_K(M)$

$(K_1, K_2) \leftarrow \text{CMAC-KEYGEN}(K)$

$M[1] \ldots M[m-1]M[m] \leftarrow M$ // $|M[m]| \leq n$

$\ell \leftarrow |M[m]|$ // $\ell \leq n$

if $\ell = n$ then $M[m] \leftarrow K_1 \oplus M[m]$

else $M[m] \leftarrow K_2 \oplus (M[m]|10^{n-\ell-1})$

$M \leftarrow M[1] \ldots M[m-1]M[m]$ // padding

$T \leftarrow \text{CBC}_K(M)$

return $T$ ↑

In an implementation, \text{CMAC-KEYGEN}(K) is run once, meaning $K_1, K$ are pre-computed, stored and re-used. Performance is then optimal.
MACing with Hashing

The software speed of hash functions (MD5, SHA1) lead people in 1990s to ask whether they could be used to MAC.

But such cryptographic hash functions are keyless.

**Question:** How do we key hash functions to get MACs?

**Proposal:** Let $H : D \rightarrow \{0, 1\}^n$ represent the hash function and set

$$T_K(M) = H(K||M)$$

Is this UF-CMA / PRF secure?
Extension Attack on MD Hash Functions

Can compute tag of $m_1 || m_2$
Suppose $H : D \to \{0,1\}^{160}$ is the hash function. HMAC has a 160-bit key $K$. Let

$$K_o = \text{opad} \oplus K||0^{352} \text{ and } K_i = \text{ipad} \oplus K||0^{352}$$

where

$$\text{opad} = 5D \text{ and } \text{ipad} = 36$$

in HEX. Then

$$\text{HMAC}_K(M) = H(K_o||H(K_i||M))$$

**Diagram:**

```
    ┌────────┐
    │        │
    │        │
    └───K_i∥M─┘
        │        └───H─┐
                      │
                      │
                      │
                      │
                      │
                      │
                      │
                      │
                      │
                      │
                      │
                      │
                      │
                      │
                      │
                      │
                      │
                      │
                      │
                      │
                      │      ┌───K_o∥X─┐
                      │         │   └───H─┐
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      │         │       │
                      └────────┘
```

$\text{HMAC}_K(M)$
HMAC

Features:
- Blackbox use of the hash function, easy to implement
- Fast in software

Usage:
- As a MAC for message authentication
- As a PRF for key derivation

Security:
- Subject to a birthday attack
- Security proof shows there is no better attack [BCK96,Be06]

Adoption and Deployment: HMAC is one of the most widely standardized and used cryptographic constructs: SSL/TLS, SSH, IPSec, FIPS 198, IEEE 802.11, IEEE 802.11b, ...
Recent Developments

**Theorem**: [BCK96] HMAC is a secure PRF assuming
- The compression function is a PRF
- The hash function is collision-resistant (CR)

But recent attacks show MD5 is not CR and SHA1 may not be either.

So are HMAC-MD5 and HMAC-SHA1 secure?
- No attacks so far, but
- Proof becomes vacuous!

**Theorem**: [Be06] HMAC is a secure PRF assuming only
- The compression function is a PRF

Current attacks do not contradict this assumption. This new result may explain why HMAC-MD5 is standing even though MD5 is broken with regard to collision resistance.
To describe the Carter-Wegman MAC first fix some large integer $N$ and set $\mathcal{T} := \mathbb{Z}_N$, the group of size $N$ where addition is defined “modulo $N$.” We use a hash function $H$ and a PRF $F$ that output values in $\mathbb{Z}_N$:

- $H$ is a keyed hash function defined over $(\mathcal{K}_H, \mathcal{M}, \mathcal{T})$,
- $F$ is a PRF defined over $(\mathcal{K}_F, \mathcal{R}, \mathcal{T})$.

The Carter-Wegman MAC, denoted $\mathcal{I}_{CW}$, takes inputs in $\mathcal{M}$ and outputs tags in $\mathcal{R} \times \mathcal{T}$. It uses keys in $\mathcal{K}_H \times \mathcal{K}_F$. The \textbf{Carter-Wegman MAC derived from $F$ and $H$} works as follows (see also Fig. 7.4):

- For key $(k_1, k_2)$ and message $m$ we define
  \[
  S( (k_1, k_2), m ) := \\
  r \leftarrow \mathcal{R} \\
  v \leftarrow H(k_1, m) + F(k_2, r) \quad \in \mathbb{Z}_N \quad \text{// addition modulo $N$} \\
  \text{output } (r, v)
  \]

- For key $(k_1, k_2)$, message $m$, and tag $(r, v)$ we define
  \[
  V( (k_1, k_2), m, (r, v) ) := \\
  v^* \leftarrow H(k_1, m) + F(k_2, r) \quad \in \mathbb{Z}_N \quad \text{// addition modulo $N$} \\
  \text{if } v = v^* \text{ output accept; otherwise output reject}
  \]

The Carter-Wegman signing algorithm uses a randomizer $r \in \mathcal{R}$. As we will see, the set $\mathcal{R}$ needs to be sufficiently large so that the probability that two tags use the same randomizer is negligible.
Difference Unpredictability

**Attack Game 7.3 (difference unpredictability).** For a keyed hash function $H$ defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$, where $\mathcal{T} = \mathbb{Z}_N$, and a given adversary $\mathcal{A}$, the attack game runs as follows.

- The challenger picks a random $k \leftarrow \mathcal{K}$ and keeps $k$ to itself.
- $\mathcal{A}$ outputs two distinct messages $m_0, m_1 \in \mathcal{M}$ and a value $\delta \in \mathcal{T}$.

We say that $\mathcal{A}$ wins the game if $H(k, m_1) - H(k, m_0) = \delta$. We define $\mathcal{A}$’s advantage with respect to $H$, denoted $\text{DUFadv}[\mathcal{A}, H]$, as the probability that $\mathcal{A}$ wins the game. □
A simple modification to $H_{\text{poly}}$ yields a good DUF. For a message $m = (a_1, a_2, \ldots, a_v) \in \mathbb{Z}_p^{\leq \ell}$ and key $k \in \mathbb{Z}_p$ define a new hash function $H_{\text{xpoly}}(k, m)$ as:

$$H_{\text{xpoly}}(k, m) := k \cdot H_{\text{poly}}(k, m) = k^{v+1} + a_1 k^v + a_2 k^{v-1} + \cdots + a_v k \in \mathbb{Z}_p.$$  (7.23)

**Lemma 7.8.** The function $H_{\text{xpoly}}$ over $(\mathbb{Z}_p, (\mathbb{Z}_p)^{\leq \ell}, \mathbb{Z}_p)$ defined in (7.23) is an $(\ell + 1)/p$-DUF.