MESSAGE AUTHENTICATION CODES and PRF DOMAIN EXTENSION

Integrity and authenticity

The goal is to ensure that
- $M$ really originates with Alice and not someone else
- $M$ has not been modified in transit

Integrity and authenticity example

Alice

Pay $100 to Charlie

Adversary Eve might
- Modify "Charlie" to "Eve"
- Modify "$100" to "$1000"

Integrity prevents such attacks.

Message authentication codes

A message authentication code $T : \text{Keys} \times D \to R$ is a family of functions.
The envisaged usage is shown below, where $A$ is the adversary:

We refer to $T$ as the MAC or tag. We have defined

Algorithm $V_K(M', T')$
If $T_K(M') = T'$ then return 1 else return 0
MAC usage

Sender and receiver share key $K$.
To authenticate $M$, sender transmits $(M, T)$ where $T = T_K(M)$.
Upon receiving $(M', T')$, the receiver accepts $M'$ as authentic iff
$\forall K(M', T') = 1$, or, equivalently, iff $T_K(M') = T'$.

UF-CMA

Let $T$: Keys $\times$ D $\rightarrow$ R be a message authentication code. Let $A$ be an adversary.

<table>
<thead>
<tr>
<th>Game UFCMA$_T$</th>
<th>procedure Initialize $K \leftarrow$ Keys; $S \leftarrow \emptyset$</th>
<th>procedure Finalize $(M, T)$ $\text{If } M \notin S \text{ then return false}$ $\text{If } M \notin D \text{ then return false}$ $\text{Return } (T = T_K(M))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>procedure Tag$(M)$ $T \leftarrow T_K(M)$; $S \leftarrow S \cup {M}$ return $T$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The uf-cma advantage of adversary $A$ is

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

Exercise: Tag lengths

Let $T$: Keys $\times$ D $\rightarrow \{0, 1\}^\ell$ be a message authentication code. Specify in pseudocode an efficient adversary $A$ making zero Tag queries and achieving $\text{Adv}^{\text{uf-cma}}_T(A) = 2^{-\ell}$.

Conclusion: Tags have to be long enough.
For 80 bit security, tags have to be at least 80 bits.
Example: Basic CBC MAC

Let $E : \{0, 1\}^k \times B \to B$ 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 $T : \{0, 1\}^k \times B^* \to B$ is defined by

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

Then $T_K(M) = T_K$.

Splicing attack on basic CBC MAC

Let $x \in \{0, 1\}^n$

$$
T_1 \leftarrow \text{Tag}(x) \\
M \leftarrow x || T_1 \oplus x \\
\text{Return } M, T_1
$$

Then,

$$
T_K(M) = E_K(E_K(x) \oplus T_1 \oplus x) \\
= E_K(T_1 \oplus T_1 \oplus x) \\
= E_K(x) \\
= T_1
$$

Exercise

Let $E : \{0, 1\}^k \times \{0, 1\}^n \to \{0, 1\}^n$ be a blockcipher. Let

$$
D = \{ M \in \{0, 1\}^n : 0 < |M| < n^2 \text{ and } |M| \mod n = 0 \}.
$$

Let $T : \{0, 1\}^k \times D \to \{0, 1\}^n$ be defined as follows:

$$
\text{Alg } T_K(M) \\
M[1] \ldots M[m] \leftarrow M; \ M[m+1] \leftarrow \langle m \rangle; \ C[0] \leftarrow 0^n \\
\text{For } i = 1, \ldots, m + 1 \text{ do } C[i] \leftarrow E_K(C[i-1] \oplus M[i]) \\
\text{Return } M, T
$$

Above, $\langle m \rangle$ denotes the $n$-bit binary representation of the integer $m$.

Show that $T$ is not UF-CMA-secure by presenting in pseudocode a practical adversary $A$ making at most 2 Tag queries and achieving $\text{Adv}^\text{uf-cma}_T(A) = 1$. 

Insecurity of basic CBC MAC

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

Then $\text{Adv}^\text{uf-cma}_T(A) = 1$ and $A$ is efficient, so the basic CBC MAC is not UF-CMA secure.
Exercise

Let $E: \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Let

$$D = \{ M \in \{0,1\}^* : 0 < |M| < n2^n \text{ and } |M| \mod n = 0 \}.$$ 

Let $T: \{0,1\}^k \times D \to \{0,1\}^n$ be defined as follows:

\[
\text{Alg } T_K(M) \\
M[1] \ldots M[m] \leftarrow M; \ C[0] \leftarrow E_K(\langle m \rangle) \\
\text{For } i = 1, \ldots, m \text{ do } C[i] \leftarrow E_K(C[i-1] \oplus M[i]) \\
T \leftarrow C[m]; \text{ return } T
\]

Above, $\langle m \rangle$ denotes the $n$-bit binary representation of the integer $m$.

Is $T$ UF-CMA-secure? If you say NO, present a practical adversary $A$ achieving $\text{Adv}^{\text{uf-cma}}_{T}(A) \geq 1/2$. If you say YES, prove this correct assuming $E$ is PRF-secure.

Replay

Suppose Alice transmits $(M_1, T_1)$ to Bank where $M_1 =$ “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. This can be done using timestamps or synchronized counters.

Any PRF is a MAC

If $F$ is PRF-secure then it is also UF-CMA-secure:

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

$$\text{Adv}^{\text{uf-cma}}_{F}(A) \leq \text{Adv}^{\text{prf}}_{F}(B) + \frac{1}{2^n}.$$ 

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

We proceed to the proof.

Adversary $B$

Given $A$ we build $B$ as follows:

- **adversary $B$**
  
  $S \leftarrow \emptyset; (M, T) \leftarrow A^{\text{TagSim}}$
  
  $T' \leftarrow F_n(M)$
  
  If $(T = T')$ and $(M \notin S)$ then $d \leftarrow 1$
  
  Else $d \leftarrow 0$
  
  Return $d$

- **subroutine** $\text{TagSim}(M)$
  
  $T \leftarrow F_n(M)$
  
  $S \leftarrow S \cup \{M\}$
  
  return $T$

$B$ runs $A$, simulating $A$’s Tag oracle via a subroutine $\text{TagSim}$ that in turn invokes $B$’s own $F_n$ oracle. When $A$ returns a forgery $(M, T)$, adversary $B$ returns $1$ if $A$ would win in game $\text{UFCMA}_F$, and $0$ otherwise.
Real game analysis

When $B$ is executed in game $\text{Real}_F$, subroutine $\text{TagSim}(M)$ will return $\text{Fn}(M)$, which equals $F_k(M)$.

So $A$ is getting the same responses to its oracle queries as it would in game $\text{UFCMA}_F$.

So

$$\Pr \left[ \text{Real}^B \Rightarrow 1 \right] = \operatorname{Adv}^\text{uf-cma}(A).$$

Rand game analysis

Here $B$ is executed in game $\text{Rand}_{\{0,1\}^n}$.

Let $(M,T)$ be the forgery that $A$ outputs. For $B$ to return 1, it must be that $M \notin S$. So $T' \leftarrow \text{Fn}(M)$ is a random $n$-bit string, independent of anything else, and has probability $2^{-n}$ of being equal to $T$.

So

$$\Pr \left[ \text{Rand}^B \Rightarrow 1 \right] \leq 2^{-n}.$$

UF-CMA $\not\Rightarrow$ PRF

We show that the converse of the Theorem is false, meaning there exist function families that are UF-CMA-secure but not PRF-secure.

We seek $F : \{0,1\}^k \times D \to \{0,1\}^\ell$ such that

1. $F$ is UF-CMA-secure, but
2. $F$ is not PRF-secure

**Approach:** We assume we are given $F' : \{0,1\}^k \times D \to \{0,1\}^n$ which is UF-CMA-secure. We modify it to construct $F$ as above.

**Alg** $F(K,M)$

$T' \leftarrow F'(K,M); T \leftarrow T'||0; \text{Return } T$

So $\ell = n + 1$. 

Putting it together

$$\operatorname{Adv}^F(B) = \Pr \left[ \text{Real}^B \Rightarrow 1 \right] - \Pr \left[ \text{Rand}^B_{\{0,1\}^n} \Rightarrow 1 \right] \geq \operatorname{Adv}^\text{uf-cma}(A) - \frac{1}{2^n}.$$ 

This yields the equation in the Theorem statement.
**F is not PRF-secure**

adversary $A$

Pick some $M \in D$

$T \leftarrow F_n(M); b \leftarrow \text{lsb}(T)$

If $(b = 0)$ then return 1 else return 0

$$\Pr\left[\text{Real}_F^A \Rightarrow 1\right] = 1$$

$$\Pr\left[\text{Rand}_F^{A,\{0,1\}^n} \Rightarrow 1\right] = \frac{1}{2}$$

So

$$\text{Adv}_F^\text{prf}(B) = \Pr\left[\text{Real}_F^A \Rightarrow 1\right] - \Pr\left[\text{Rand}_F^{A,\{0,1\}^n} \Rightarrow 1\right] = 1 - \frac{1}{2} = \frac{1}{2}$$

**PRF domain extension**

A family of functions $F$: Keys $\times D \rightarrow R$ 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
  
  $D = \{ M \in \{0, 1\}^* : 0 < |M| < n2^n \text{ and } |M| \text{ mod } n = 0 \}$

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.

**F is UF-CMA-secure**

The following says that the UF-CMA security of $F'$ is inherited by $F$.

**Claim:** Let $A$ be a uf-cma adversary making $q$ Tag queries and having running time $t$. Then there is a uf-cma adversary $A'$ such that

$$\text{Adv}_F^{\text{uf-cma}}(A) \leq \text{Adv}_F^{\text{uf-cma}}(A')$$

Adversary $A'$ makes $q$ queries to its Tag oracle and has running time $t$ plus some overhead.

adversary $A'$(M, T) $\leftarrow A^{\text{TagSim}}$

subroutine $\text{TagSim}(M)$

T' $\leftarrow T[1..n]$

T $\leftarrow T'||0$

Return $(M, T')$

Return $T$

Above, $T[1..n]$ denotes the first $n$ bits of $T$.

**PRF domain extension**

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

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
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

$\text{Alg } T_{K_{in}}(M)
\begin{align*}
C[0] &\leftarrow 0^n \\
\text{for } i = 1, ..., m \text{ do } \\
C[i] &\leftarrow E_{K_{in}}(C[i - 1] \oplus M[i]) \\
T &\leftarrow E_{K_{out}}(C[m]) \\
\text{return } T
\end{align*}$

Birthday attacks on MACs

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.

Exercise

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

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

$\text{Adv}^{\text{prf-cma}}_T(A) = \Omega\left(\frac{q^2}{2^n}\right)$

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

Security of ECBC

Birthday attack is best possible:

$\textbf{Theorem:}$ 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

$\text{Alg } F_{K_{in}}(M)
\begin{align*}
C[0] &\leftarrow 0^n \\
\text{for } i = 1, ..., m \text{ do } \\
C[i] &\leftarrow E_{K_{in}}(C[i - 1] \oplus M[i]) \\
T &\leftarrow E_{K_{out}}(C[m]) \\
\text{return } T
\end{align*}$

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 $D$ against $E$ such that

$\text{Adv}^{\text{prf}}_F(A) \leq \text{Adv}^{\text{prf}}_E(D) + \frac{\sigma^2}{2^n}$

and $D$ makes at most $\sigma$ oracle queries and has running time about $t$. 
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 \( n = 128 \).

Non-full messages

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?

\[
\begin{array}{c|c|c}
\end{array}
\]

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

Padding with 0*:

\[
\begin{array}{c|c|c}
\end{array}
\]

adversary \( A \)

\( T \leftarrow \text{Tag}(1^n1^n0); \) Return \((1^n1^n00, T)\)

This adversary has uf-cma advantage 1.
Non-full messages

Padding with 10*: For a non-full message

\[
M[1] \quad M[2] \quad M[3] \parallel 10^*
\]

For a full message

\[
M[1] \quad M[2] \quad M[3] \quad 10^*
\]

This works, but if \( M \) was full, an extra block is needed leading to an extra blockcipher operation.

Costs

Handling length-variability 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[BIRo], OMAC/OMAC1[IW]

CMAC Components and Setup

\[ E : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n \] 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

\[
\text{Alg} \quad \text{CMAC-KEYGEN}(K)
\]

\[
K_0 \leftarrow E_K(0)
\]

if msb(\( K_0 \)) = 0 then \( K_1 \leftarrow (K_0 \ll 1) \) else \( K_1 \leftarrow (K_0 \ll 1) \oplus J \)

if 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.
CMAC Algorithm

**Alg** CMAC\(_K(M)\)

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

\(M[1] \ldots M[m-1]M[m] \leftarrow M\) \quad // \quad \(|M[m]| \leq n\)

\(\ell \leftarrow |M[m]|\) \quad // \quad \ell \leq n\)

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

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

\(M \leftarrow M[1] \ldots M[m-1]M[m]\)

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

return \(T\)

In an implementation, CMAC-KEYGEN\(_K) is run once, meaning \(K_1, K_2\)
are pre-computed, stored and re-used. Performance is then optimal.

MACing with hash functions

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

\[ T_K(M) = H(K|\langle M \rangle) \]

Is this UF-CMA / PRF secure?

Extension attack

Let \(M' = M|\langle m+1 \rangle\). Then

\[ H(K|M') = h(\langle m+2 \rangle|H(K|M)) \]

so given the MAC \(H(K|M)\) of \(M\) we can easily forge the MAC of \(M'\).

**Exercise:** Specify in pseudocode an adversary mounting the above attack
to achieve uf-cma advantage 1 using 1 Tag query.
HMAC [BCK96]

Suppose \( H: D \rightarrow \{0, 1\}^n \) is the hash function, built from an underlying compression function via the MD transform.

Let \( B \geq n/8 \) denote the byte-length of a message block (\( B = 64 \) for MD5, SHA1, SHA256)

Define the following constants
- ipad : The byte 36 repeated \( B \) times
- opad : The byte 5C repeated \( B \) times

HMAC: \( \{0, 1\}^n \times D \rightarrow \{0, 1\}^n \) is defined as follows:

**Alg** HMAC\((K, M)\)

\[ K_i \leftarrow \text{ipad} \oplus K||0^{6B-n} \ ; K_o \leftarrow \text{opad} \oplus K||0^{6B-n} \]

\[ X \leftarrow H(K_i||M) \ ; Y \leftarrow H(K_o||X) \]

Return \( Y \)

HMAC Security

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

But attacks show MD5 and SHA1 are not CR.

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 result may explain why HMAC-MD5 and HMAC-SHA1 are standing even though the hash functions are broken with regard to collision resistance.

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, ...
HMAC Recommendations

- Don’t use HMAC-MD5
- No immediate need to remove HMAC-SHA1
- Use HMAC-SHA256, HMAC-SHA512 for new applications