Message Authentication Codes

Murat Kantarcioglu

Secrecy does not Imply Authenticity/Integrity

- Encryption only provides secrecy
- In many cases we want authenticity/integrity
  - Financial transactions
- Our goal: Ensure the authenticity / integrity of the messages
- Assumptions: Shared secret key between parties

Example: CTR Mode

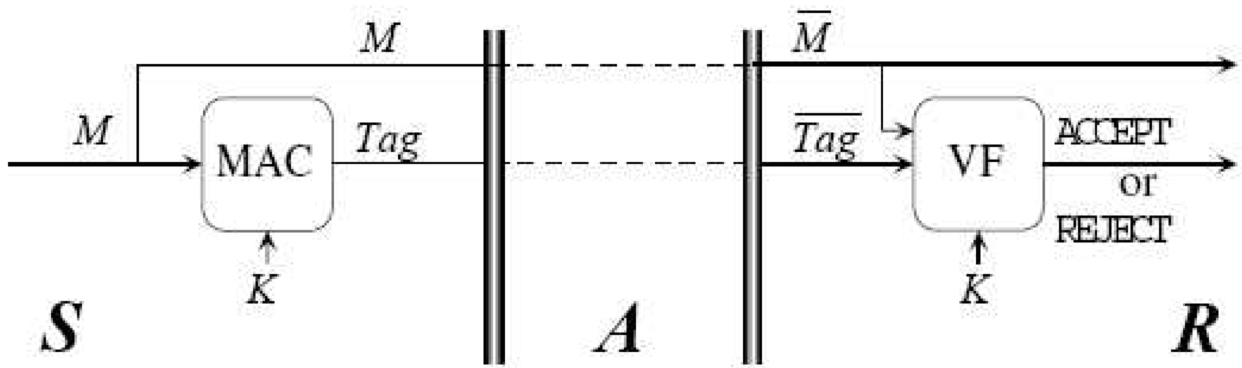
- Remember in CTR encryption

$$C_i = E_K(ctr + i) \oplus M_i$$

- An attacker can reverse any bit without being detected
- Note that

$$D_K(C_i \oplus a) = (C_i \oplus a) \oplus E_(ctr + i) = M_i \oplus a$$

Message Authentication Code



- We need three algorithms:
  - Key generation (*K*)

$$K \xleftarrow{\$} K$$

  - Message authentication code generator (*MAC*)

$$Tag \xleftarrow{\$} MAC_K(M)$$

  - Verifier (*VF*)

$$d \leftarrow VF_K(M, Tag) \text{ where } d \in \{0,1\}$$

Security for MACs

- Key generation algorithm will pick random keys from the key space.
- Deterministic MAC implies that

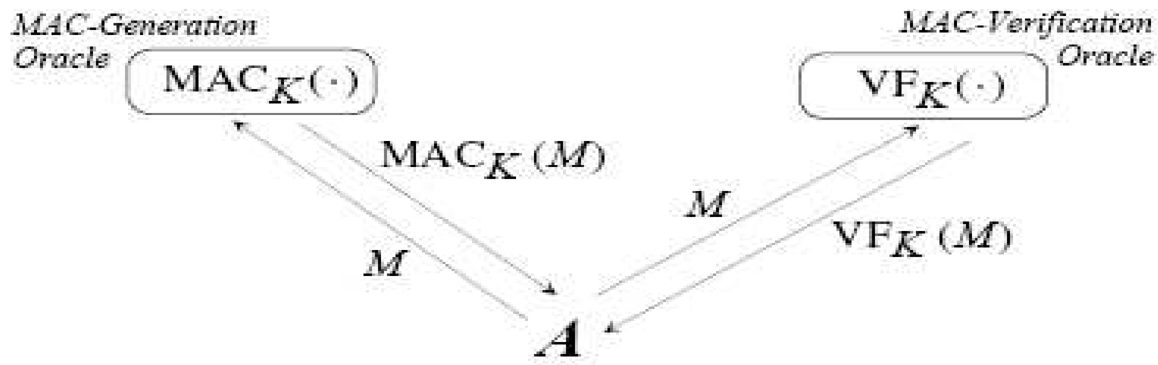
$$
\begin{array}{l}
\text{algorithm } \mathrm{VF}_K(M, Tag) \\
\quad Tag' \leftarrow \mathrm{MAC}_K(M) \\
\quad \text{if } \ (Tag = Tag' \text{ and } Tag' \neq \perp) \text{ then } \ \text{return } 1 \text{ else } \ \text{return } 0.
\end{array}
$$

- MAC and VF algorithms are assumed to be stateless

Towards defining security of MACs

- Adversary is allowed to see some message and tag pairs
- Security against key recovery is not enough
- No assumptions about the message space
- We do not consider replay attacks
  - i.e., Adversary must forge an unseen message M
- Adversary could adaptively chose messages to be tagged.
- Adversary could query whether a given tag is valid

MAC Security



- Adversary is given MAC-generation and MAC-verification oracles.
- Adversary tries to generate (M, Tag) such that M is not queried to MAC-generation oracle but (M, Tag) is accepted by MAC-verification oracle.

MAC Security

Experiment $\mathbf{Exp}_{\Pi}^{\text{uf-cma}}(A)$

$K \xleftarrow{\$} \mathcal{K}$

Run $A^{\text{MAC}_K(\cdot),\text{VF}_K(\cdot,\cdot)}$

If A made a verification query (M, Tag) such that the following are true

- The verification oracle returned 1
- A did not, prior to making verification query (M, Tag), make signing query M

Then return 1 else return 0

The *uf-cma advantage* of A is defined as

$$\mathbf{Adv}_{\Pi}^{\text{uf-cma}}(A) \;=\; \Pr\left[\mathbf{Exp}_{\Pi}^{\text{uf-cma}}(A) = 1\right] . \blacksquare$$

Examples

We fix a PRF $F : \{0,1\}^k \times \{0,1\}^l \mapsto \{0,1\}^L$

Let $\quad \Pi_1 = (K, MAC)$

algorithm $\text{MAC}_K(M)$
  if $(|M| \bmod \ell \neq 0$ or $|M| = 0)$ **then return** $\perp$
  Break M into ℓ bit blocks $M = M[1] \ldots M[n]$
  for $i = 1, \ldots, n$ **do** $y_i \leftarrow F_K(M[i])$
  $Tag \leftarrow y_1 \oplus \cdots \oplus y_n$
  return Tag

Adversary $A_1^{\text{MAC}_K(\cdot), \text{VF}_K(\cdot,\cdot)}$
  Let x be some ℓ-bit string
  $M \leftarrow x \,\|\, x$
  $Tag \leftarrow 0^L$
  $d \leftarrow \text{VF}_K(M, Tag)$

Note that $Adv_{\Pi_1}^{uf-cma}(A_1) = 1$

Example

Note that A_1 does not work for $\Pi_2 = (K, MAC)$

algorithm $\mathrm{MAC}_K(M)$
  $l \leftarrow \ell - m$
  if $(|M| \bmod l \neq 0$ or $|M| = 0$ or $|M|/l \geq 2^m)$ **then return** $\perp$
  Break M into l bit blocks $M = M[1] \ldots M[n]$
  for $i = 1, \ldots, n$ **do** $y_i \leftarrow F_K([i]_m \,\|\, M[i])$
  $Tag \leftarrow y_1 \oplus \cdots \oplus y_n$
  return Tag

Adversary $A_2^{\mathrm{MAC}_K(\cdot)}$
  Let a_1, b_1 be distinct, $\ell - m$ bit strings
  Let a_2, b_2 be distinct $\ell - m$ bit strings
  $Tag_1 \leftarrow \mathrm{MAC}_K(a_1 a_2)$; $Tag_2 \leftarrow \mathrm{MAC}_K(a_1 b_2)$; $Tag_3 \leftarrow \mathrm{MAC}_K(b_1 a_2)$
  $Tag \leftarrow Tag_1 \oplus Tag_2 \oplus Tag_3$
  $d \leftarrow \mathrm{VF}_K(b_1 b_2, Tag)$

Example

- We can prove that $Adv_{\Pi_2}^{uf-cma}(A_2) = 1$
- Note that

$$
\begin{aligned}
Tag_1 &= F_K([1]_m || a_1) \oplus F_K([2]_m || a_2) \\
Tag_2 &= F_K([1]_m || a_1) \oplus F_K([2]_m || b_2) \\
Tag_3 &= F_K([1]_m || b_1) \oplus F_K([2]_m || a_2)
\end{aligned}
$$

- We can easily find $MAC_K(b_1 b_2)$

$$
\begin{aligned}
MAC_K(b_1 b_2) &= Tag_1 \oplus Tag_2 \oplus Tag_3 \\
&= F_K([1]_m || b_1) \oplus F_K([2]_m || b_2)
\end{aligned}
$$

Security of MACs

- Our attacks do not depend on the properties of the underlying PRF
- Using even random functions would not help
- Perfect ingredients + Bad recipe => Bad Food
- Good crypto primitives + Bad design => Insecure systems

PRF as a MAC Paradigm

We fix a PRF $F : \{0,1\}^k \times \{0,1\}^l \mapsto \{0,1\}^L$

Let $\Pi = (K, MAC)$

algorithm $\mathcal{K}$	algorithm $\mathrm{MAC}_K(M)$
$K \xleftarrow{\$} \mathsf{Keys}$	**if** $(M \notin D)$ **then return** $\perp$
return K	$Tag \leftarrow F_K(M)$
	Return Tag

Proposition 6.3 Let F: $\mathsf{Keys} \times D \to \{0,1\}^\tau$ be a family of functions and let $\Pi = (\mathcal{K}, \mathrm{MAC})$ be the associated message authentication code as defined above. Let A by any adversary attacking Π, making q_s MAC-generation queries of total length μ_s, q_v MAC-verification queries of total length μ_v, and having running time t. Then there exists an adversary B attacking F such that

$$\mathbf{Adv}_{\Pi}^{\text{uf-cma}}(A) \leq \mathbf{Adv}_F^{\text{prf}}(B) + \frac{q_v}{2^\tau} . \tag{6.1}$$

Furthermore B makes $q_s + q_v$ oracle queries of total length $\mu_s + \mu_v$ and has running time t.

PRF as a MAC Paradigm

- Proof of the Proposition:

Adversary B^f
    $d \leftarrow 0$; $S \leftarrow \emptyset$
    Run A
        When A asks its signing oracle some query M:
            Answer $f(M)$ to A ; $S \leftarrow S \cup \{M\}$
        When A asks its verification oracle some query (M, Tag):
            if $f(M) = Tag$ **then**
                answer 1 to A; **if** $M \notin S$ then $d \leftarrow 1$
                else answer 0 to A
    Until A halts
    return d

$$\Pr\left[\mathbf{Exp}_F^{\mathrm{prf\text{-}1}}(B) = 1\right] = \mathbf{Adv}_{\Pi}^{\mathrm{uf\text{-}cma}}(A)$$

$$\Pr\left[\mathbf{Exp}_F^{\mathrm{prf\text{-}0}}(B) = 1\right] \leq \frac{q_{\mathrm{v}}}{2^{\tau}} .$$

Universal Hash-then-PRF Paradigm

- Candidate PRF functions (DES,AES) have fixed input length
- We want to MACs to work on arbitrary length inputs
- Idea: $MAC_{K_1||K_2}(M) = F_{K_2}(H_{K_1}(M))$
- Universal Hash Functions:

$$\forall M_1, M_2 \in D, Pr[H(K, M_1) = H(K, M_2)] = \frac{1}{R}$$

- Proof Idea: If the Hash function is universal then $F_{K_2}(H_{K_1}(M))$ is also a secure PRF. Then use prf as a MAC paradigm.

CBC MAC

- Basic Version: $\Pi = (K, MAC)$

Algorithm $\mathrm{MAC}_K(M)$
If $M \notin$ Messages then return $\perp$
Break M into n-bit blocks $M[1] \cdots M[m]$
$C[0] \leftarrow 0^n$
For $i = 1, \ldots, m$ do $C[i] \leftarrow E_K(C[i-1] \oplus M[i])$
Return $C[m]$

- Security: It is secure if we use fixed message length. It is hard to get it right in practice.

HASH Based MACs

- Remember SHA-1 $\{0,1\}^{<2^{64}} \mapsto \{0,1\}^{160}$
- SHA-1 is believed to be collision resistant but how use SHA-1 for secure MACs construction?
- Some incorrect starts:

$$\begin{aligned} MAC(K,M) &= H(K||M) \\ &= H(M||K) \\ &= H(K||M||K) \end{aligned}$$

- Provable secure version:

$$H(K \oplus a || H(K \oplus b || M))$$

Which MAC to use in practice?

- CBC-MAC is hard to get correct in practice.
- UMACs is provably secure but needs platform specific modifications for efficiency
- HMACs is provably secure and can be easily implemented using standard crypto library.