0 CBC-MAC

Write the basic construction of CBC-MAC.

Note: this comes from Construction 4.11 in Katz/Lindell.

CBC-MAC Construction

Let F be a pseudorandom function, and fix a length function $\ell > 0$. Then basic CBC-MAC construction is as follows:

1. $\mathsf{Mac}_k(m)$: On input key $k \in \{0,1\}^n$, message m such that $len(m) = \ell(n) \cdot n$
   (a) Parse $m = m_1, \ldots, m_\ell$ where $len(m_i) = n$.
   (b) Set $t_0 := 0^n$. Then for $i \in 1, \ldots, \ell$: set $t_i := F_k(t_{i-1} \oplus m_i)$

   Output t_ℓ as the tag.
2. $\mathsf{Vrfy}_k(m,t)$: On input key $k \in \{0,1\}^n$, message m, tag t:
   (a) Check $len(m) = \ell(n) \cdot n$. If not output 0.
   (b) Output 1 iff $t \stackrel{?}{=} \mathsf{Mac}_k(m)$.

1 Merkle-Damgård

Let $h : \{0,1\}^{n+t} \to \{0,1\}^n$ be a fixed-length compression function. Suppose we forgot a few features of Merkle-Damgård and construct H as follows:

- Value x is input.
- Split x into $y_0, x_1, \ldots, x_k$. Where y_0 is n bits and x_i (for $i = 1, \ldots, k$) is t bits.. The last piece x_k may be padded with zeroes.
- For $i = 1, \ldots, k$, set $y_i = h(y_{i-1}||x_i)$.
- Output y_k.

It's similar to Merkle-Damgård except no IV and the final padding block is missing.

1. Describe an easy way to find two messages that are broken up into the same number of pieces, which have the same hash value under H.
2. Describe an easy way to find two messages that are broken up into a different number of pieces, which have the same hash value under H. *Hint: Pick any string of length $n + 2t$, and find a shorter string that collides with it.*

Neither of your collisions above should involve finding a collision in h!

1. Same number of blocks

We denote this transform as MD_{bad}. Suppose x is of length $N = n + kt - (t-1)$. Then we parse x as $y_0, x_1, \dots, x_k$ where $x_k = x[N]||0^{t-1}$. That is, x has only one bit in the final block, so we pad with $t-1$ 0s. Then say $x' = x||0^{k-1}$.
There is no length checking, so it is clear that $\mathsf{MD}_{bad}(x) = \mathsf{MD}_{bad}(x')$ although $x \neq x'$. Also, x, x' break up into the same number of blocks.

2. Different number of blocks

First, run MD_{bad} on $w = t_0\, w_1$. This yields $t_1 = h(t_0||w_1)$
Then:

$$x = t_1\, x_1$$
$$x' = t_0\, w_1\, x_1$$

Now note that for x:

$$y_0 = t_1$$
$$y_1 = h(t_1||x_1)$$

For x':

$$y'_0 = t_0$$
$$y'_1 = h(y'_0||x'_1) = h(t_0||w_1) = t_1$$
$$y'_2 = h(y'_1||x'_2) = h(t_1||x_1)$$

Note that y_1 is the output for $H(x)$, and y'_2 is the output for $H(x')$.
We see:

$$y'_2 = h(t_1||x_1)$$
$$y_1 = h(t_1||x_1)$$

Then although $x \neq x'$, $H(x) = H(x')$.

2 Hash Functions

I designed $H : \{0,1\}^* \to \{0,1\}^n$. I make $H(x) = x$ if x is n-bit string – but assume H's behavior is more complicated on strings of other lengths. This way we know there are no collisions among n-bit strings. Is this a good design decision?

A function H is collision-resistant if it is infeasible for any PPT algorithm to find a collision in H. This means it should be hard to compute any collision $x = x'$ such that $H(x) = H(x')$. We show that H as described above is not collision-resistant using the following attack.

Attack

$\underline{\mathcal{A}_{cr}()}$

1. Pick $x \leftarrow_{\$} \{0,1\}^{n'}$ (where $n' > n$).
2. Calculate $y := H(x)$.
3. Output (y, x) as the collision pair.

Analysis of $\mathcal{A}_{cr}$'s success

Note that $len(y) = n$ because H's range is $\{0,1\}^n$. Then H is defined on y as $H(y) = y$. Also, H itself takes polynomial time to calculate, so it is feasible for $\mathcal{A}_{cr}$ to calculate $H(x) = y$. Because $len(x) > n$, we know $x \neq y$ but $H(x) = y = H(y)$. Thus $\mathcal{A}_{cr}$ has found a collision with probability 1, which is clearly non-negligible.

3 MAC

Prove that the following modifications of basic CBC-MAC do not yield a secure MAC (even for fixed-length messages).

1. Mac outputs all blocks $t_1, \ldots, t_\ell$ rather than just t_ℓ. Verification only checks if t_ℓ is correct.
2. A random initial block is used each time a message is authenticated. That is, choose a uniform $t_0 \in \{0,1\}^n$, run basic CBC-MAC over the "message" $t_0, m_1, \ldots, m_\ell$ and output tag $\langle t_0, t_\ell \rangle$. Verification is done in a natural way.

Recall the message authentication experiment (rewritten from Katz/Lindell section 4.2) where $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$

$\mathsf{Macforge}_{\mathcal{A},\Pi}(n)$

1. A key k is generated from $\mathsf{Gen}(1^n)$.
2. $\mathcal{A}$ is given 1^n and oracle access to $\mathsf{Mac}_k(\cdot)$. Let $\mathcal{Q} = \{m | \mathcal{A} \text{ queries } \mathsf{Mac}_k(m)\}$.
3. $\mathcal{A}$ outputs (m, t).
4. $\mathcal{A}$ succeeds iff :

$$\begin{aligned} \mathsf{Vrfy}_k(m,t) = 1 \text{ and} \\ m \notin \mathcal{Q} \end{aligned}$$

Attack 1

Algorithm $\mathcal{A}_{\mathsf{Mac}}$

- $\mathcal{A}_{\mathsf{Mac}}$ queries on $m = m_1||m_2$ where $m_1 = m_2 = \vec{0}$. He receives $\langle t_1, t_2 \rangle$.
- $\mathcal{A}_{\mathsf{Mac}}$ produces $m^* = m_1^*||m_2^*$ where $m_1^* = t_1$ and $m_2^* = t_2$. The tag is $\langle t_1^*, t_2^* \rangle = \langle t_2, t_1 \rangle$

Analysis of $\mathcal{A}_{\mathsf{Mac}}$'s success In the query, we have $m = \vec{0}||\vec{0}$. Then we know:

$$\begin{aligned} t_0 &= \vec{0} \\ t_1 &= F_k(t_0 \oplus \vec{0}) = F_k(\vec{0}) \\ t_2 &= F_k(t_1 \oplus \vec{0}) = F_k(t_1) \end{aligned}$$

And $\mathcal{A}_{\mathsf{Mac}}$ is given t_1 and t_2. Now by setting $m^* = t_1||t_2$ we see that

$$\begin{aligned} t_0^* &= \vec{0} \\ t_1^* &= F_k(t_0^* \oplus t_1) = F_k(t_1) = t_2 \\ t_2^* &= F_k(t_1^* \oplus t_2) = F_k(t_2 \oplus t_2) = F_k(\vec{0}) = t_1 \end{aligned}$$

We conclude that $\mathcal{A}_{\mathsf{Mac}}$ is able to break this scheme using only one query. Thus the scheme is not secure.

Attack 2

Algorithm $\mathcal{A}_{\mathsf{Mac}}$

- Queries on message $m_1 = \vec{0}$, receives $\langle t_0, t_1 \rangle = \langle r_0, F_k(r_0) \rangle$.
- Queries on message $m_2 = \vec{0}$, receives $\langle t_0', t_1' \rangle = \langle r_0', F_k(r_0') \rangle$.
- Produces $m^* = r_0 \oplus r_0'$, $\langle t_0, t_1' \rangle$.

Analysis of $\mathcal{A}_{\mathsf{Mac}}$'s success
The choice is $\langle t_0, t_1' \rangle = \langle r_0, F_k(r_0') \rangle$. $\mathcal{A}_{\mathsf{Mac}}$ can choose any input they like for t_0. In the Mac,

$$\begin{aligned} t_1' = F_k(t_0 \oplus m^*) &\implies \\ t_1' = F_k(r_0 \oplus (r_0 \oplus r_0') &\implies \\ t_1' = F_k(r_0') \end{aligned}$$

This means t_1' was chosen correctly. $\mathcal{A}_{\mathsf{Mac}}$ wins with non-negligible probability and we conclude that the scheme is not secure.

4 Digital Signature

Let (G, S, V) be a secure signature scheme with message space $\{0,1\}^n$, and security parameter λ. Let $(pk_0, sk_0) \leftarrow_\$ G(1^\lambda)$ and $(pk_1, sk_1) \leftarrow_\$ G(1^\lambda)$ be two pairs of signing/verification keys.

Which of the following are secure signature schemes? Show an attack or prove security.

1. (S_1, V_1):
   - Sign. $S_1((sk_0, sk_1), m)$: Output $(S(sk_0, m), S(sk_1, m))$.
   - Verify. $V_1((pk_0, pk_1), m, (\sigma_0, \sigma_1))$: Output 1 if $(V(pk_0, m, \sigma_0) \vee V(pk_1, m, \sigma_1))$, 0 otherwise.

     I.e., the verification accepts if one of the two signatures accepts.
2. (S_2, V_2)
   - Sign. $S_2((sk_0, sk_1), (m_L, m_R))$: Output $(S(sk_0, m_L), S(sk_1, m_R))$.
   - Verify. $V_2((pk_0, pk_1), (m_L, m_R), (\sigma_0, \sigma_1))$: Output 1 if $(V(pk_0, m_L, \sigma_0) \wedge V(pk_1, m_R, \sigma_1))$, 0 otherwise. I.e., both verifications must accept.

Recall the signature experiment scheme (from Katz/Lindell 12.2) where $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$

$\underline{\mathsf{Sigforge}_{\mathcal{A},\Pi}(n)}$

1. $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$
2. $\mathcal{A}$ gets pk and access to $\mathsf{Sign}_{sk}(\cdot)$ oracle.
3. $\mathcal{A}$ outputs (m, σ). Let $\mathcal{Q} = \{m | \mathcal{A} \text{ asked } \mathsf{Sign}_{sk}(m)\}$.
4. $\mathcal{A}$ succeeds iff

$$\begin{aligned} \mathsf{Vrfy}_{pk}(m, \sigma) = 1 \\ m \notin \mathcal{Q} \end{aligned}$$

We say that a signature scheme is existentially unforgeable under an adaptive chosen-message attack (or secure) if for all PPT adversaries $\mathcal{A}$, there is a negligible function negl such that:

$$Pr[\mathsf{Sigforge}_{\mathcal{A},\Pi}(n)] \leq \mathsf{negl}(n)$$

Signature Scheme 1 is secure: Proof by Contradiction

Theorem. If $\Pi = (G, S, V)$ is a secure signature scheme then so is $\Sigma = (S_1, V_1)$.

Proof. By contradiction. We will prove the following statement.

If Σ is not existentially unforgeable under an adaptive chosen-message attack (EUF-CMA) then Π is not EUF-CMA.

Step 1. Let $\mathcal{F}_\Sigma$ be an adversary for Π who can win the EUF-CMA experiment with non-negligible probability $p(n)$. We construct $\mathcal{F}_\Pi$.

$\mathcal{F}_\Pi(1^\lambda)$ Obtain in input the public key pk.

Reduction

1. $\mathcal{F}_\Pi$ flips a bit $b \in \{0, 1\}$. He sets $pk = pk_b$.
2. $\mathcal{F}_\Pi$ has pk_b. He calculates $(pk_{1-b}, sk_{1-b}) \leftarrow_\$ G(1^\lambda)$.
3. $\mathcal{F}_\Pi$ actives $\mathcal{F}_\Sigma$. He forwards pk_0, pk_1 to $\mathcal{F}_\Sigma$.
4. On each query of m, $\mathcal{F}_\Pi$ forwards m to his challenger to get σ_0. He calculates σ_1 himself using sk_1. He returns σ_0, σ_1 to $\mathcal{F}_\Sigma$.
5. When $\mathcal{F}_\Sigma$ produces a forgery on m^* $(\sigma_0^*, \sigma{*}_1)$.
6. If $V(pk_b, m^*, \sigma_b^*) = 1$ $\mathcal{F}_\Pi$ outputs m^*, σ_b^*. Else, output $\perp$.

analysis of $\mathcal{F}_\Pi$'s success

If $\mathcal{F}_\Sigma$ is has a successful forgery, then it must be the case that at least one of $V(pk_0, m^*, \sigma_0)$ and $V(pk_1, m^*, \sigma_1)$ verified.

Since $\mathcal{F}_\Sigma$ picks the bit b at random, probability that b corresponds to the bit of the signature that is verified is at least $\frac{1}{2}$.

Thus $\mathcal{F}_\Sigma$ outputs a valid forgery for the scheme $\Sigma = (S, V)$ with probability $\frac{1}{2}p$

Signature Scheme 2 is not secure. Forgery

Idea of the attack: "Sign Halves"
We define $\mathcal{F}$ as a forger for the Signature scheme $\Sigma_2 = (S_2, V_2)$.

$\mathcal{F}(1^\lambda)$ Obtain in input the public key pk.
Training Phase

1. Query the signing oracle with $m^0 = m_L^0 || m_R^0$, and $m^1 = m_L^1 || m_R^1$, where $m_L^0 \neq m_R^0 \neq m_L^1 \neq m_R^1$.
2. Receive $\sigma^0 = \sigma_0^0 || \sigma_1^0$ and $\sigma^0 = \sigma_0^1 || \sigma_1^1$, respectively.

Challenge Phase
Output message $m^* = m_L^0 || m_R^1$ and signature $\sigma^* = \sigma_0^0 || \sigma_1^1$.

Analysis of $\mathcal{F}$'s success
We know $\sigma_0^0 = S(sk_0, m_L^0)$ and $\sigma_1^1 = S(sk_1, m_R^1)$ (which $\mathcal{F}$ received from the sign oracle). Then $V(pk_0, m_L^0, \sigma_0^0) = 1$ and $V(pk_1, m_R^1, \sigma_1^1) = 1$. Thus the signature verifies. Furthermore, $m^* \neq m^0$ and $m^* \neq m^1$ so $m^* \notin \mathcal{Q}$. So $\mathcal{F}$ produces a successful forgery with probability 1.