Chapter 04

Mingjia Huo

February 16, 2019

Problem 4.1. *Proof.* Consider an adversary $\mathcal{A}$ for a message authentication code $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$, such that the Mac-forge$_{\mathcal{A},\Pi}(n)$ works in the following procedure:

- A key k is generated by running $\text{Gen}(1^n)$.
- The adversary $\mathcal{A}$ is given input 1^n, then randomly select a tag t with uniform distribution. (For the length of tags is $t(n)$, each tag will be selected with probability $2^{-t(n)}$.) Then $\mathcal{A}$ uniformly draws m from the message space and outputs (m, t).
- Because $\mathcal{A}$ doesn't make any query to the oracle $\text{Mac}_k(\cdot)$, $\mathcal{A}$ succeeds if and only if $\text{Vrfy}_k(m, t) = 1$, which means Mac-forge$_{\mathcal{A},\Pi}(n) = 1$.

Assume when we select m with uniform distribution, the probability of tag t is $\Pr[T = t]$. (Here T is a random variable.) Then we compute:

$$\begin{aligned}
&\Pr[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1] \\
=&\sum_t \Pr[T = t] \times \Pr[t \text{ is selected by } \mathcal{A}] \\
=&2^{-t(n)} \sum_t \Pr[T = t] \\
=&2^{-t(n)}.
\end{aligned}$$

If $t(n) = \mathcal{O}(\log n)$, we have $2^{-t(n)} = \mathcal{O}(\frac{1}{n^d})$, which is not $\text{negl}(n)$.

So t must be super-logarithmic. □

Problem 4.6. The algorithm is not secure.

Proof. Construct an adversary $\mathcal{A}$:

1. $\mathcal{A}$ generates three different messages m_1, m_2, m_3 with length $n - 1$.
2. Then $\mathcal{A}$ gets access to oracle $\text{Mac}_k(\cdot)$ and the oracle tells him $t_a = \text{Mac}_k(m_0||m_1)$ and $t_b = \text{Mac}_k(m_1||m_2)$.
3. $\mathcal{A}$ combines the first half of $\text{Mac}_k(m_0||m_1)$ with the second half of $\text{Mac}_k(m_1||m_2)$, then he get t'.

4. Output $(m_0||m_2, t')$.

Assume the length of output value is $l_{out}(n)$ in the pseudorandom function F, and $F_k(0\|m_0) = t_0, F_k(0\|m_1) = t_1, F_k(1\|m_1) = t_2, F_k(1\|m_2) = t_3$. By the definition of Mac, we have

$$t_a = \text{Mac}_k(m_0||m_1) = F_k(0\|m_0)\|F_k(1\|m_1) = t_0\|t_2,$$

$$t_b = \text{Mac}_k(m_1||m_2) = F_k(0\|m_1)\|F_k(1\|m_2) = t_1\|t_3.$$

Then

$$t' = \text{Mac}_k(m_0||m_2) = F_k(0\|m_0)\|F_k(1\|m_2) = t_0\|t_3.$$

We construct $\text{Vrfy}(m_0\|m_2, t') = 1$ with $m = m_0\|m_2$ wasn't queried by $\mathcal{A}$ with probability 1, so the algorithm is not secure. □

Problem 4.12. Advantages:

The modification changes length l to a single bit 0 or 1. The advantage is obvious:

- It shortens the length of message by $\mathcal{O}(\log n)$ bits and has better performance in practice.
- Without using the length, we can make encryption while reading the message.

Proof. Assume $\mathcal{A}$ is a probabilistic polynomial-time adversary, Π is the MAC for arbitrary-length messages, and Π' is a MAC for fixed-length messages.

Just like the proof in textbook, we define:

- Repeat: The same random identifier $r_i = r_j$ appear in two of the tags returned by the MAC oracle.
- NewBlock: At least one of the blocks $r\|j\|i\|m_i, j = 0, 1$ was never previously authenticated by the oracle.

So

$$\begin{aligned}
&\Pr[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1] \\
=&\Pr[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1 \wedge \text{Repeat}] \\
+&\Pr[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1 \wedge \overline{\text{Repeat}} \wedge \text{NewBlock}] \\
+&\Pr[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1 \wedge \overline{\text{Repeat}} \wedge \overline{\text{NewBlock}}].
\end{aligned}$$

We show the three part is negligible respectively:

1. $\Pr[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1 \wedge \text{Repeat}] \le \Pr[\text{Repeat}]$. And the probability of event Repeat is exactly the probability that $r_i = r_j$ for some $i \neq j$. Applying birthday bound in the textbook, we have $\Pr[\text{Repeat}] \le \frac{q(n)^2}{2^{n/4}}$. Since $\mathcal{A}$ makes only polynomially many queries, this value is negligible.

2. $\Pr[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1 \wedge \overline{\text{Repeat}} \wedge \text{NewBlock}] \leq \Pr[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1 \wedge \text{NewBlock}]$. It is also negligible by *Claim* 10 in the textbook.

3. Finally, we prove that

$$\Pr[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1 \wedge \overline{\text{Repeat}} \wedge \overline{\text{NewBlock}}] = 0.$$

Let $q = q(n)$ denote the number of MAC oracle queries made by $\mathcal{A}$, and r_i denote the denote the random identifier used to answer the ith oracle query. Let (m, t) be the output of $\mathcal{A}$.

Because there is no NewBlock, we have $r \in \{r_1, \cdots, r_q\}$. Assume $r = r_j$, and the jth query is about $m^{(j)}$, which is represented by d' blocks.

Consider the total block number d of (m, t) output by $\mathcal{A}$:

(a) $d = d'$: If $\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1$, then we must have $m \neq m^{(j)}$. Since m and $m^{(j)}$ have equal length, there must be at least one index i for which $m_i \neq m_i^{(j)}$.
Since the ith block is $r\|0\|i\|m_i$ with i represents the position of block, it was then never authenticated in the jth Mac query. So this is a NewBlock, a contradiction.

(b) $d \neq d'$: Consider the last block $r\|1\|d\|m_d$. In the jth Mac query, there is only one block $r\|1\|d'\|m_{d'}^{(j)}$ which has "1" to represent it's the last block. But $d \neq d'$, so $r\|1\|d\|m_d$ is a NewBlock, a contradiction.

Thus

$$\Pr[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1 \wedge \overline{\text{Repeat}} \wedge \overline{\text{NewBlock}}] = 0.$$

To sum up,

$$\begin{aligned}
&\Pr[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1] \\
\leq&\text{negl}(n) + \text{negl}(n) + 0 \\
=&\text{negl}(n).
\end{aligned}$$

So it's secure. □

Problem 4.25. Firstly, since F is a permutation, so $\mathcal{A}$ just randomly chooses $c \in \{0,1\}^n$, there is always a r and m, such that $\text{Enc}(r\|m) = c$, and $\text{Dec}(c) = m$. So

$$\Pr[\text{Enc-Forge}_{\mathcal{A},\Pi}(n) = 1] = 1,$$

which means this scheme Π is not unforgeable and is not a authenticated encryption scheme.

Next we prove it's CCA-secure.

Proof. Without loss of generality, assume the permutation is fixed length Given any adversary $\mathcal{A}$, we can construct a distinguisher D which access an oracle $\mathcal{O}$:$\{0,1\}^n \to \{0,1\}^n$ and $\mathcal{O}^{-1}$. In detail:

1. Run $\mathcal{A}(1^{2n})$.

2. When $\mathcal{A}$ queries its encryption oracle on a message $m \in \{0,1\}^n$, answer this query in the following way:
   (a) choose uniform $r \in \{0,1\}^n$.
   (b) Query $\mathcal{O}(r\|m)$ and obtain response y.
   (c) Return the ciphertext y to $\mathcal{A}$.
3. When $\mathcal{A}$ queries its decryption oracle on a ciphertext $c \in \{0,1\}^n$, answer this query in the following way: compute $\mathcal{O}^{-1}(c)$ and the second half is m.
4. When $\mathcal{A}$ outputs messages $m_0, m_1 \in \{0,1\}^n$, choose a uniform bit $b \in \{0,1\}$ and then:
   (a) choose uniform $r_0 \in \{0,1\}^n$.
   (b) Query $\mathcal{O}(r_0\|m_b)$ and obtain response y.
   (c) Return the ciphertext y to $\mathcal{A}$.
5. Continue answering encryption and decryption oracle queries of $\mathcal{A}$ as before until $\mathcal{A}$ outputs a bit b'. Output 1 if $b' = b$, and 0 outherwise.

Thus, D outputs 1 *if and only if* $\mathcal{A}$ succeeds. Let Π denotes our construction, and $\widetilde{\Pi}$ denotes the theme when we replace F_k with a uniform permutation $f \in \mathrm{Perm}_n$. Then:

$$\Pr[\mathrm{PrivK}^{cca}_{\mathcal{A},\Pi}(2n) = 1] = \Pr_{k \leftarrow \{0,1\}^{2n}}[D^{F_k(\cdot),F_k^{-1}(\cdot)}(1^{2n}) = 1],$$

$$\Pr[\mathrm{PrivK}^{cca}_{\mathcal{A},\widetilde{\Pi}}(2n) = 1] = \Pr_{f \leftarrow \mathrm{Perm}_{2n}}[D^{f(\cdot),f^{-1}(\cdot)}(1^{2n}) = 1].$$

By the definition of pseudorandom permutation, we have

$$|\Pr[D^{F_k(\cdot),F_k^{-1}(\cdot)}(1^{2n}) = 1] - \Pr[D^{f(\cdot),f^{-1}(\cdot)}(1^{2n}) = 1]| \leq \mathrm{negl}(2n).$$

Thus, to prove $\Pr[\mathrm{PrivK}^{cca}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n)$, we only need to prove:

$$\Pr[\mathrm{PrivK}^{cca}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n)$$

Let $\mathcal{C}$ be the set of all ciphertext that has been asked by $\mathcal{A}$ or answer by the oracle. That is, $\mathrm{Enc}(m_{ask}) = c$ or $Dec(c)$ is asked. Let **Repeat** be the event that when $\mathcal{A}$ output (m_0, m_1), the ciphertext $c^* = \mathrm{Enc}(m_b)$ is in $\mathcal{C}$. So,

$$\begin{aligned}&\Pr[\mathrm{PrivK}^{cca}_{\mathcal{A},\widetilde{\Pi}}(n) = 1]\\ =&\Pr[\mathrm{PrivK}^{cca}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathrm{Repeat}] + \Pr[\mathrm{PrivK}^{cca}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathrm{Repeat}}]\end{aligned}$$

Separately:

1. $\Pr[\mathrm{PrivK}^{cca}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathrm{Repeat}]$:

   There are two cases:

(a) There is some m asked by $\mathcal{A}$, such that $\text{Enc}(m)$ outputs c^*.

If $m \neq m_b$, the probability is 0; and if $m = m_b$, the probability is equal to the probability that their r are equal. There are at most $q(n)$ queries, so the probability is $\leq \frac{q(n)}{2^{n/2}} = \text{negl}(n)$.

(b) $\mathcal{A}$ asks c before the experiment outputs (m_0, m_1). Since there are $2^{-n/2}$ values that $\text{Enc}(m_b)$ have, so the probability is also $\text{negl}(n)$.

To sum up, $\Pr[\text{PrivK}^{cca}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \text{Repeat}] = \text{negl}(n)$.

2. $\Pr[\text{PrivK}^{cca}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\text{Repeat}}]$:

Let's first have a deep insight to the queries.

We know that $f \in \text{Perm}_n$. Then a query can delete some impossible functions and save the possible ones. When there is no **Repeat**, the encryption and decryption can be viewed as the same: to match a ciphertext $c \in \{0,1\}^n$ to a message $m \in \{0,1\}^{n/2}$.

After $q(n)$ queries, assume there are n_0 ciphertexts which matches m_0 and n_1 ciphertexts which matches m_1. And the number of r they used is also n_0 and n_1, denotes as sets R_0, R_1.

So there are $2^{n/2} - n_0$ ciphertexts waiting to be matched to m_0, and $2^{n/2} - n_1$ ciphertexts waiting to be matched to m_1.

Since f is uniformly drawn from permutations, so given r not involved in the queries of m_0, m_1, we have

$$\Pr[f(r\|m_0) = c^*] = \Pr[f(r\|m_1) = c^*] = p.$$

The probability taken over uniform chosen of functions f which satisfy $\mathcal{A}$'s queries and the output of $c^* = \text{Enc}(m_b)$. Furthermore,

$$\sum_{r \notin R_0} \Pr[f(r\|m_0) = c^*] = (2^{n/2} - n_0) \times p$$

$$\sum_{r \notin R_1} \Pr[f(r\|m_1) = c^*] = (2^{n/2} - n_1) \times p$$

Thus, when $c = \text{Enc}(m_b)$,

$$\begin{aligned}
&\Pr[\text{Dec}(c) = m_0 \wedge \overline{\text{Repeat}}] \\
=&\Pr[\text{Dec}(c) = m_0 \mid \overline{\text{Repeat}}] \times \Pr[\overline{\text{Repeat}}]] \\
=&\frac{\sum_{r \notin R_0} \Pr[f(r\|m_0) = c^*]}{\sum_{r \notin R_0} \Pr[f(r\|m_0) = c^*] + \sum_{r \notin R_1} \Pr[f(r\|m_1) = c^*]} \times (1 - \text{negl}(n)) \\
=&\frac{2^{n/2} - n_0}{2^{n/2+1} - n_0 - n_1} \times (1 - \text{negl}(n)) \\
\leq&\frac{1}{2} + \text{negl}(n)
\end{aligned}$$

The last inequality holds because n_0, n_1 are polynomial of n.
Similarly, $\Pr[\mathrm{Dec}(c) = m_1 \wedge \overline{\mathrm{Repeat}}]] \leq \frac{1}{2} + \mathrm{negl}(n)$. So whatever b' that $\mathcal{A}$ outputs,

$$\begin{aligned}
&\Pr[\mathrm{PrivK}^{cca}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathrm{Repeat}}] \\
=&\Pr[\mathrm{Dec}(c) = m_0 \wedge \overline{\mathrm{Repeat}}] \Pr[b = 0] + \Pr[\mathrm{Dec}(c) = m_1 \wedge \overline{\mathrm{Repeat}}] \Pr[b = 1] \\
\leq&\frac{1}{2} + \mathrm{negl}(n)
\end{aligned}$$

To sum up,

$$\begin{aligned}
&\Pr[\mathrm{PrivK}^{cca}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \\
=&\Pr[\mathrm{PrivK}^{cca}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathrm{Repeat}] + \Pr[\mathrm{PrivK}^{cca}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathrm{Repeat}}] \\
\leq&\mathrm{negl}(n) + \frac{1}{2} + \mathrm{negl}(n) \\
=&\frac{1}{2} + \mathrm{negl}(n),
\end{aligned}$$

which finishes the proof of CCA-secure. □

Problem 10. First, give F'_k as:

$$F'_k(m) = \begin{cases} 0^{|m|}, & m = k \\ F_k(k), & m = F_k^{-1}(0^{|m|}) \\ F_k(m), & others \end{cases}$$

Since F_k is a permutation, then we exchange two matches in F_k to get F'_k, so F'_k **is also a permutation.**

Construct D which can access oracle $\mathcal{O}.\mathcal{O}^{-1}$:

1. Given 1^n, then $\mathcal{A}$ simply asks $\mathcal{O}^{-1}(0^n)$.
2. Assume the answer is k. Then uniformly select a message $m \in \{0,1\}^n$, and queries $\mathcal{O}(m) = c$.
3. If $c = F_k(m)$, output 1; otherwise, output 0.

If $\mathcal{O} = f$, then $\Pr[D^{f(\cdot),f^{-1}(\cdot)}(1^n) = 1] = \frac{1}{2^n} = \mathrm{negl}(n)$.
If $\mathcal{O} = F'_k$, then $\Pr[D^{F'_k(\cdot),F'^{-1}_k(\cdot)}(1^n) = 1] \leq 1 - \frac{2}{2^n} = 1 - \mathrm{negl}(n)$.
Thus,
$$|\Pr[D^{f(\cdot),f^{-1}(\cdot)}(1^n) = 1] - \Pr[D^{F'_k(\cdot),F'^{-1}_k(\cdot)}(1^n) = 1]| > \mathrm{negl}(n).$$

So the theme is not a strong pseudorandom permutation.

Next, we prove that this theme is a **pseudorandom permutation.**

Proof. Prove by contradiction: Assume D' can use oracle $\mathcal{O}'$ to distinguish F' from random function f, that is

$$|\Pr[D'^{F'_k(\cdot)}(1^n) = 1] - \Pr[D'^{f(\cdot)}(1^n) = 1]| > \text{negl}(n).$$

Without loss of generation, let

$$\Pr[D'^{F'_k(\cdot)}(1^n) = 1] - \Pr[D'^{f(\cdot)}(1^n) = 1] > \text{negl}(n).$$

Let *Bingo* be the event that **at least one of** k **and** $m_0 = F_k^{-1}(0^{|m|})$ **have been asked by** D'. Assume there are $q(n)$ queries. Compute

$$\begin{aligned}
&\Pr[D'^{F_k(\cdot)}(1^n) = 1] \\
=&\Pr[D'^{F_k(\cdot)}(1^n) = 1 \wedge \text{Bingo}] + \Pr[D'^{F_k(\cdot)}(1^n) = 1 \wedge \overline{\text{Bingo}}] \\
\leq&\Pr[D'^{F_k(\cdot)}(1^n) = 1 \mid \text{Bingo}] \times \Pr[\text{Bingo}] + \Pr[D'^{F_k(\cdot)}(1^n) = 1 \wedge \overline{\text{Bingo}}] \\
\leq&1 \times \Pr[\text{Bingo}] + \Pr[D'^{F_k(\cdot)}(1^n) = 1 \wedge \overline{\text{Bingo}}]
\end{aligned}$$

First, we prove $\Pr[\text{Bingo}] = \text{negl}(n)$.

If there is a $\mathcal{PPT}$ A', such that $\Pr[\text{Bingo}] > \text{negl}(n)$, construct D'' with oracle $\mathcal{O}$:

1. Run 1^n. Run A'.
2. If A' asks m, then A compute $F_m(m) = c'$ and asks $\mathcal{O}(m) = c$.
3. If $c = 0^n$ or $c = c'$, output 1 and return. If not, give c to A'.
4. If $c = 0^n$ or $c = c'$ don't happen in all the queries and A' ends, uniformly output $b \in \{0, 1\}$.

If $\mathcal{O} = f$, $\Pr[c = 0^n \vee c = c'] \leq 2 \times \frac{q(n)}{2^n} = \text{negl}(n)$. So $\Pr[D''^{f(\cdot)}(1^n) = 1] \leq \frac{1}{2} + \text{negl}(n)$.

If $\mathcal{O} = F_k$, then when A' asks k, then $c = c'$ happens; and when A' asks m_0, $c = 0^n$ happens. (If D'' doesn't return, it means Bingo doesn't happen in A'.) So

$$\Pr[c = 0^n \vee c = c'] \geq \Pr[\text{Bingo}] > \text{negl}(n),$$

which means $\Pr[D''^{F_k(\cdot)}(1^n) = 1] > \frac{1}{2} + \text{negl}(n)$. Thus

$$\Pr[D''^{F_k(\cdot)}(1^n) = 1] - \Pr[D''^{f(\cdot)}(1^n) = 1] > \text{negl}(n),$$

which contradicts that F_k is a pseudorandom permutation.

So $\Pr[\text{Bingo}] = \text{negl}(n)$.

Use the conclusion above, if

$$\Pr[D'^{F'_k(\cdot)}(1^n) = 1] - \Pr[D'^{f(\cdot)}(1^n) = 1] > \text{negl}(n),$$

then

$$\Pr[D'^{F_k(\cdot)}(1^n) = 1 \wedge \overline{\text{Bingo}}] - \Pr[D'^{f(\cdot)}(1^n) = 1] > \text{negl}(n).$$

Construct D based on D' with oracle $\mathcal{O}$.

1. Given 1^n. Run D the same as D'.
2. When D' asks to encrypt a message m, run $\mathcal{O}(m) = c$ and give c to D'.
3. Output the same value with D.

Analysis:

1. If $\mathcal{O} = f$, D and D' behave the same. So

$$\Pr[D'^{f(\cdot)}(1^n) = 1] = \Pr[D^{f(\cdot)}(1^n) = 1]$$

2. If $\mathcal{O} = F_k$,

$$\begin{aligned}
&\Pr[D^{F_k(\cdot)}(1^n) = 1] \\
\geq &\Pr[D^{F_k(\cdot)}(1^n) = 1 \wedge \overline{\text{Bingo}}] \\
= &\Pr[D'^{F'_k(\cdot)}(1^n) = 1 \wedge \overline{\text{Bingo}}],
\end{aligned}$$

So

$$\begin{aligned}
&\Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1] \\
> &\Pr[D^{F_k(\cdot)}(1^n) = 1 \wedge \overline{\text{Bingo}}] - \Pr[D'^{f(\cdot)}(1^n) = 1] \\
= &\Pr[D'^{F_k(\cdot)}(1^n) = 1 \wedge \overline{\text{Bingo}}] - \Pr[D'^{f(\cdot)}(1^n) = 1] \\
> &\text{negl}(n),
\end{aligned}$$

a contradiction.

To sum up, F'_k is a **pseudorandom permutation**. □