CSE5014 · Cryptography and Network Security

Assignment 3

Liu Leqi / 12011327

May 8, 2023

1 Q1

1. G' is not necessarily a pseudorandom generator.
   G is pseudorandom for random input in $\{0,1\}^{2|s|}$, for which the probability is $2^{-2|s|}$, but the probability of an input of $s||0^{|s|}$ is only $2^{-|s|}$. So input of this are not random and therefore the output need not be pseudorandom.
2. G' is necessarily a pseudorandom generator.
   Suppose $|G(s)| = l(n)$. Since G is a pseudorandom generator, there is a distinguisher D such that

   $$|\Pr_{y\leftarrow U_{l(n)}}[D(y)=1] - \Pr_{s\leftarrow U_n}[D(G(s))=1]| \le \epsilon(n)$$

   Then suppose there is a distinguisher D' for G'. If the challenger provides a uniform distributed string y, the success probability is

   $$\Pr_{y\leftarrow U_{l(n)}}[D'(y)=1] = \Pr_{y\leftarrow U_{l(n)}}[D(y)=1] = \frac{1}{2}$$

   If the challenger provides a string $G'(s)$, the success probability is

   $$\Pr_{s\leftarrow U_n}[D'(G'(s))=1] = \Pr_{s\leftarrow U_n}[D'(s_1||G(s_2))=1]$$

   And since $s = s_1||s_2$ and $s_1, s_2 \in \{0,1\}^{|s|/2}$, it chooses s_1 with probability $2^{-|s|/2}$ and s_2 with probability $2^{-|s|/2}$. Thus, D' cannot distinguish s_1 with random string and can distinguish $G(s_2)$ as D can.

   $$\Pr_{s\leftarrow U_n}[D'(s_1||G(s_2))=1] = \Pr_{s\leftarrow U_n}[D(G(s))=1]$$

   Therefore,

   $$|\Pr_{y\leftarrow U_{l(n)}}[D'(y)=1] - \Pr_{s\leftarrow U_n}[D'(G'(s))=1]| \le \epsilon(n)$$

   If G is pseudorandom, then $\epsilon(n)$ is negligible. So G' is necessarily pseudorandom.

2 Q2

Suppose there is a distinguisher D with oracle O. The distinguisher D queries O with x_1, x_2 and $x_1 + x_2$, and outputs 1 if and only if $O(x_1) + O(x_2) = O(x_1 + x_2)$. Therefore,

- if $O = F$, then $\Pr[D^{F(\cdot)}(1^n) = 1] = 1$ (since F is under the field of $\mathbb{F}_2 = (\{0,1\}, \oplus, \cdot)$)
- if $O = f$ chosen uniformly from $Func_n$, then $\Pr[D^{f(\cdot)}(1^n) = 1] = 2^{-n}$

The difference is $|\Pr[D^{F(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1]| = 1 - 2^{-n}$, which is not negligible. Therefore, F is not a pseudorandom function.

3 Q3

Suppose there is a ditinguisher D and an oracle O. With input 1^n, A can query oracle many times, in the following steps.

1. $r \leftarrow_R \{0,1\}^n$
2. $y := O(r)$
3. return the ciphertext $< r, y \oplus m >$ to A

At any time, A outputs 2 message $m_x, m_2 \in \{0,1\}^n$ to D, and D does the following steps.

1. $r \leftarrow_R \{0,1\}^n, b \leftarrow \{0,1\}$
2. $y := O(r)$
3. return the ciphertext $< r, y \oplus m_b >$ to A

A can still access oracle and then finally outputs b' to D. If $b' = b$ then D outputs 1; otherwise, outputs 0. Therefore, we have

1. If $O = F_k$, then $y := F_k(r)$. So
$$\Pr[D^{F(\cdot)}(1^n) = 1] = \Pr[PrivK^{CPA}_{A,\Pi}(n) = 1]$$
2. If $O = f$, then $y := f(r)$. So
$$\Pr[D^{f(\cdot)}(1^n) = 1] = \Pr[PrivK^{CPA}_{A,\tilde{\Pi}}(n) = 1]$$

Since CTR-mode uses PRF F_k in each block, then

$$|\Pr[D^{F(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1]| \leq negl(n)$$

Combining them, we can get

$$|\Pr[PrivK^{CPA}_{A,\Pi}(n) = 1] - \Pr[PrivK^{CPA}_{A,\tilde{\Pi}}(n) = 1]| \leq negl(n)$$

4 Q4

1. It is EAV-secure but not CPA-secure.
   (a) EAV-secure
   Since F_k is a PRF, eavesdropper cannot distinguish it with uniformly random string. Thus it is EAV-secure.
   (b) not CPA-secure
   Since F_k is deterministic, $F_k(0^n)$ outputs the same at each time. Thus this scheme is not CPA-secure.
2. It is EAV-secure but not CPA-secure.
   (a) EAV-secure
   Suppose there exists an efficient attacker A such that
   $$\Pr[PrivK^{eav}_{A,\Pi}(n) = 1] > \frac{1}{2} + \epsilon(n)$$
   Since $\Pr[A(U_n) = 1] \leq \frac{1}{2}$ and $\Pr[A(m \oplus k) = 1] = \Pr[PrivK^{eav}_{A,\Pi}(n) = 1] > \frac{1}{2} + \epsilon(n)$, where $k \leftarrow_R \{0,1\}^n$, we have
   $$|\Pr[A(m \oplus k) = 1] - \Pr[A(U_n) = 1]| > \epsilon(n)$$
   Define D: $\{0,1\}^n \rightarrow \{0,1\}$ as : $D(y) = A(y \oplus m)$. D is efficient since A is efficient, but we have
   $$|\Pr[D(k)) = 1] - \Pr[D(m \oplus U_n) = 1]| > \epsilon(n)$$
   Since $m \oplus U_n \equiv U_n$, it contradicts that k is a random string. Therefore, such A does not exist and such scheme is EAV-secure.

(b) not CPA-secure
The attacker A knows that the ciphertext c is generated from either m_1 or m_2. A can query oracle O for the ciphertext c_1 and c_2 of m_1 and m_2, and output $b' = 1$ if $c_1 = c$; otherwise, output $b' = 0$. Since the scheme is deterministic, A will always win, i.e., it is not CPA-secure.

3. It is neither EAV-secure nor CPA-secure.
Since G is deterministic and the random string r can be learnt from the ciphertext, attacker can compute $G(r)$ and reveal the message. Thus this scheme is not secure.

4. It is EAV-secure but not CPA-secure.

(a) EAV-secure
Since $k \leftarrow_R \{0,1\}^n$ and F_k is a PRF, $F_k(r)$ and $F_k(r+1)$ are uniformly and independently distributed in $\{0,1\}^n$. There is no attacker that can distinguish $< r, m \oplus F_k(r), m \oplus F_k(r+1) >$ with random string. Thus, it is EAV-secure.

(b) CPA-secure
Since $k \leftarrow_R \{0,1\}^n, r \leftarrow_R \{0,1\}^n$, it will get uniformly independently random $F_k(r)$ (or $F_k(r+1)$) at each time to encrypt. So the encryption of the scheme is indeterministic. Thus, the scheme is CPA-secure.

5 Q5

1. It is not CPA-secure.
Suppose that the attacker A outputs 2 messages

$$m^1 = m_1^1 || m_2^1,$$
$$m^2 = m_1^2 || m_2^2$$

The challenger will give back the ciphertext $c = c_1 || c_2 || c_3$ with

$$c_1 = p_k(x_1),$$
$$c_2 = p_k(c_1 \oplus x_2),$$
$$c_3 = p_k(c_2 \oplus r)$$

where $x_1 \in \{m_1^1, m_1^2\}, x_2 \in \{m_2^1, m_2^2\}, r \in \{0,1\}^m$.
Then A can query the encryption oracle for the message m^1 and receive the ciphertext $c' = c_1' || c_2' || c_3'$. If $c_1 = c_1'$, then A outputs $b' = 1$; otherwise, A outputs $b' = 2$. A will always win in this construction.

2. It is CPA-secure.
Suppose that the attacker A outputs 2 messages

$$m^1 = m_1^1 || m_2^1,$$
$$m^2 = m_1^2 || m_2^2$$

The challenger will give back the ciphertext $c = c_1 || c_2 || c_3$ with

$$c_1 = p_k(r),$$
$$c_2 = p_k(c_1 \oplus x_1),$$
$$c_3 = p_k(c_2 \oplus x_2)$$

where $x_1 \in \{m_1^1, m_1^2\}, x_2 \in \{m_2^1, m_2^2\}, r \in \{0,1\}^m$.
Since p_k is a PRP and r is chosen uniformly from $\{0,1\}^m$, A cannot distinguish $c_1 = p_k(r)$ with a random string, and the same for c_2, c_3. Therefore, it is CPA-secure.

6 Q6

1. Reordering the block m_i of message does not change the tag. That is, the attacker can queries oracle $MAC(\cdot)$ for

$$m = m_1||...||m_l$$

And get the return as

$$t = F_k(m_1) \oplus ... \oplus F_k(m_l)$$

Then the attacker can output (m', t') as

$$\begin{aligned} m' &= m_2||m_1...m_l \\ t' &= F_k(m_2) \oplus F_k(m_1) \oplus ... \oplus F_k(m_l) = F_k(m_1) \oplus F_k(m_2) \oplus ... \oplus F_k(m_l) \end{aligned}$$

which satisfies $Vrfy_k(m, t) = 1$ and $m' \notin \{m\}$

2. Suppose the attacker queries oracle $MAC(\cdot)$ for the following 3 messages.

$$\begin{aligned} m^1 &= m_1||m_2, \\ m^2 &= m_3||m_2, \\ m^3 &= m_3||m_4 \end{aligned}$$

And the oracle returns corresponding 3 tags.

$$\begin{aligned} t^1 &= F_k([1]_2||m_1) \oplus F_k([2]_2||m_2), \\ t^2 &= F_k([1]_2||m_3) \oplus F_k([2]_2||m_2), \\ t^3 &= F_k([1]_2||m_3) \oplus F_k([2]_2||m_4), \end{aligned}$$

Then the attacker can outputs (m, t) as

$$\begin{aligned} m &= m_1 \oplus m_2 \oplus m_3 = m_1||m_4, \\ t &= t_1 \oplus t_2 \oplus t_3 = F_k([1]_2||m_1) \oplus F_k([2]_2||m_4) \end{aligned}$$

which satisfies $Vrfy_k(m, t) = 1$ and $m \notin \{m^1, m^2, m^3\}$

3. The uniform r might not influence the authentication code. The attacker can take $m \in \{0, 1\}^{\frac{n}{2}}$, i.e., there is only one block, and then take $r = [1]_2||m$. So the attacker can compute the tag as $t = ([1]_2||m, F_k([1]_2||m) \oplus F_k([1]_2||m)) = ([1]_2||m, 0^n)$. Therefore, the attacker can output $(m, t) = (m, ([1]_2||m, 0^n))$ where no matter what m is chosen, $Vrfy_k(m, t) = 1$.

7 Q7

Suppose the attacker queries oracle $MAC(\cdot)$ for the following 2 messages.

$$\begin{aligned} m^1 &= m_1^1||m_2^1, \\ m^2 &= m_1^2||m_2^2 \end{aligned}$$

And the oracle returns corresponding 2 tags.

$$\begin{aligned} t^1 &= t_1^1||t_2^1 = F_k(m_1^1)||F_k(F_k(m_2^1)), \\ t^2 &= t_1^2||t_2^2 = F_k(m_1^2)||F_k(F_k(m_2^2)) \end{aligned}$$

Then the attacker can outputs (m, t) as

$$\begin{aligned} m &= m_1^1||m_2^2, \\ t &= t_1^1||t_2^2 = F_k(m_1^1)||F_k(F_k(m_2^2)) \end{aligned}$$

which satisfies $Vrfy_k(m, t) = 1$ and $m \notin \{m^1, m^2\}$

8 Q8

1. Suppose there is a PPT attacker A attacking the scheme (E', D') in a chosen-ciphertext attack. Let **ValidQuery** be the event that A submits a new, valid ciphertext to its decryption oracle.

   (a) $Pr[ValidQuery] \leq negl(n)$. Proof as below.
   It is obvious that if **ValidQuery** occurs then in the **MAC-forge** experiment, the adversary has forged a new, valid pair (c, t). Let $q(\cdot)$ be the polynomial upper bound of the number of decryption-oracle queries made by A.
   Consider the adversary A_M attacking the message authentication code Π_M with A running as its subroutine. A_M is given input 1^n and has access to a MAC oracle $Mac_{k_M}(\cdot)$.

      i. $k_E \leftarrow_R \{0,1\}^n, i \leftarrow_R \{1, ..., q(n)\}$

      ii. Run A on input 1^n. A can query encryption oracle and decryption oracle at any time. Then A outputs 2 messages m_0, m_1 to A_M and A_M directly outputs to the challenger, the ciphertext from the challenger is also directly transmitted to A and A outputs $b' \in \{0,1\}$.

         - When A queries encryption oracle for the message m.

           A. $c \leftarrow Enc_{k_E}(m)$

           B. Query the MAC oracle for c and receive t in response. Return $< c, t >$ to A.

         - When A queries decryption oracle for the ciphertext $< c, t >$.

           – If this is the i-th decryption oracle query, output (c, t) and halt.

           – Otherwise,

             ∗ If $< c, t >$ was a response to a previous encryption oracle query for a message m, return m.

             ∗ Otherwise, return error.

   If A_M correctly guesses the first index i for which **ValidQuery** occurs, A_M succeeds in experiment **MAC-forge**$_{A_M,\Pi_M}(n)$. The probability that A_M guesses i correctly is $1/q(n)$. Therefore,

   $$\Pr[MAC - forge_{A_M,\Pi_M}(n) = 1] \geq \Pr[ValidQuery]/q(n)$$

   Since Π_M is a secure MAC and q is polynomial, then $Pr[ValidQuery] \leq negl(n)$.

   (b) $\Pi = (E', D')$ is CCA-secure. Proof as below.
   We have

   $$\begin{aligned}\Pr[PrivK^{cca}_{A,\Pi}(n) = 1] &= \Pr[PrivK^{cca}_{A,\Pi}(n) = 1|ValidQuery] \Pr[ValidQuery] \\ &+ \Pr[PrivK^{cca}_{A,\Pi}(n) = 1|\overline{ValidQuery}] \Pr[\overline{ValidQuery}] \\ &\leq \Pr[ValidQuery] + \Pr[PrivK^{cca}_{A,\Pi}(n) = 1 \wedge \overline{ValidQuery}]\end{aligned}$$

   Since $Pr[ValidQuery]$ is negligible, we need to show that

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

   Consider the adversary A_E attacking Π_E in a chosen-plaintext attack with A running as its subroutine. A_E is given input 1^n and has access to $Enc_{k_E}(\cdot)$.

      i. $k_E \leftarrow_R \{0,1\}^n$

      ii. Run A on input 1^n.

         - When A queries encryption oracle for the message m.

           A. Query $Enc_{k_E}(\cdot)$ and receive c in response.

           B. $t \leftarrow Mac_{k_M}(c)$. Return $< c, t >$ to A.

- When A queries decryption oracle for the ciphertext $< c, t >$.
  - If $< c, t >$ was a response to a previous encryption oracle query for a message m, return m.
  - Otherwise, return error.

iii. When A outputs the 2 messages m_0, m_1 to A_M, A_M directly outputs to the challenger and receives the challenge ciphertext c. Then compute $t \leftarrow Mac_{k_M}(c)$ and return $< c, t >$ to A as the challenge ciphertext.

iv. Output the same b' as output by A.

A running as a subroutine of A_E is distributed identically to A in experiment $PrivK^{cca}_{A,\Pi}(n)$ until **ValidQuery** occurs. Therefore,

$$\begin{aligned}\Pr[PrivK^{cpa}_{A_E,\Pi_E}(n) = 1] &\geq \Pr[PrivK^{cpa}_{A_E,\Pi_E}(n) = 1 \wedge \overline{ValidQuery}] \\ &= \Pr[PrivK^{cca}_{A,\Pi}(n) = 1 \wedge \overline{ValidQuery}]\end{aligned}$$

Since Π_E is CPA-secure, then

$$\Pr[PrivK^{cpa}_{A_E,\Pi_E}(n) = 1] \leq \frac{1}{2} + negl(n)$$

Thus, we have

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

Above all, Π is CCA-secure.

2. Suppose the encryption is $E'_k(x) = (y, t)$, where $y = E_k(m) = f_k(r||m)$ with $r \leftarrow_R \{0,1\}^n$, which is CPA-secure and even CCA-secure, and $t = f_k^{-1}(y)$, which is a secure MAC. However, the encryption yields that

$$\begin{aligned}y &= E_k(m) = f_k(r||m) \\ t &= f_k^{-1}(y) = f_k^{-1}(f_k(r||m)) = r||m\end{aligned}$$

We can see that the message is revealed in the tag. Therefore, if using the same key, the scheme is not CCA-secure, even CPA-secure.

9 Q9

1. The main idea is to modify the message but keep the final block of tag the same.
Suppose the attacker queries oracle $MAC(\cdot)$ for the following 2 messages.

$$\begin{aligned}m^1 &= m_1^1||m_2^1, \\ m^2 &= m_1^2||m_2^2\end{aligned}$$

And the oracle returns corresponding 2 tags.

$$\begin{aligned}t^1 &= t_1^1||t_2^1, \\ t^2 &= t_1^2||t_2^2\end{aligned}$$

By the property of basic CBC-MAC, we can know that $t_1^1 = F_k(m_1^1)$, $t_1^2 = F_k(m_1^2)$. Suppose the modified message is $m = m_1^1||x$, then we have

$$MAC(m) = MAC(m_1^1||x) = F_k(m_1^1)||F_k(F_k(m_1^1) \oplus x) = t_1^1||F_k(t_1^1 \oplus x)$$

Since we need to keep the last block of the tag the same, let $t_1^1 \oplus x = m_1^2$, i.e., $x = t_1^1 \oplus m_1^2$. Then we have

$$MAC(m) = t_1^1||F_k(t_1^1 \oplus x) = t_1^1||F_k(m_1^2) = t_1^1||t_1^2$$

Therefore, the attacker can outputs (m, t) as

$$m = m_1^1 || (t_1^1 \oplus m_1^2),$$
$$t = t_1^1 || t_1^2$$

which is a valid pair of message and tag.

2. Suppose the attacker queries oracle $MAC(\cdot)$ for an one-block message m and gets the corresponding tag $t =< t_0, t_l >$. Then $(m \oplus r, < t_0 \oplus r, t_l >)$, where $r \leftarrow_R \{0,1\}^n$, is a valid pair of message and tag.

10 Q10

1. Let **SameNumber** be the event that Alice and Bob receive the same number. We have

$$\begin{aligned}\Pr[SameNumber] &= 1 - \Pr[\overline{SameNumber}] \\ &= 1 - (1 - \frac{1}{10 \cdot 10 \cdot 26}) \\ &= \frac{2599}{2600}\end{aligned}$$

2. Let **A** be the event that at least 2 license plates have the same number. Suppose the number of this type of license plates that they can issue is n, then

$$\begin{aligned}\Pr[A] &= 1 - \Pr[\overline{A}] \\ &= 1 - \prod_{i=1}^{n-1}(1 - \frac{i}{10 \cdot 10 \cdot 26})\end{aligned}$$

By Taylor expansion, we have

$$e^x \approx 1 + x$$

Thus, $e^{-\frac{i}{2600}} \approx 1 - \frac{i}{2600}$. So we have

$$\begin{aligned}\Pr[A] &\approx 1 - \prod_{i=1}^{n-1} e^{-\frac{i}{2600}} \\ &= 1 - e^{\sum_{i=1}^{n-1} -\frac{i}{2600}} \\ &= 1 - e^{\frac{n(n-1)}{2} \cdot \frac{-1}{2600}} \\ &\approx 1 - e^{\frac{-n^2}{5200}}\end{aligned}$$

Since we want $\Pr[A] < 1\%$, then we have $n < \sqrt{2600 \ln(\frac{100}{99})} \approx 5.11$. Therefore, the maximum number of this type of license plates that they can issue is 5.

3. Let **A** be the event that at least 2 license plates have the same number. Suppose n more digits are needed, then

$$\begin{aligned}\Pr[A] &= 1 - \Pr[\overline{A}] \\ &= 1 - \prod_{i=1}^{49}(1 - \frac{i}{10 \cdot 10 \cdot 26 \cdot 10^n})\end{aligned}$$

Still by Taylor expansion, we have

$$\begin{aligned}\Pr[A] &\approx 1 - \prod_{i=1}^{49} e^{-\frac{i}{2600 \cdot 10^n}} \\ &= 1 - e^{\sum_{i=1}^{49} -\frac{i}{2600 \cdot 10^n}} \\ &= 1 - e^{\frac{-1176}{2600 \cdot 10^n}}\end{aligned}$$

Since we want $\Pr[A] < 1\%$, then we have $n > \log_{10}(\frac{1176}{26\ln(\frac{100}{99})}) - 2 \approx 1.65$. Therefore, 2 more digits are needed at least.

11 Q11

Assume $\tilde{H}$ is not a collision resistant hash function, i.e.,

$$\exists x \neq y, \tilde{H}^s(x) = \tilde{H}^s(y)$$

Therefore, we have $H^s(H^s(x)) = H^s(H^s(y))$.

- If $H^s(x) = H^s(y)$, then (x, y) is a pair of collision of H.
- If $H^s(x) \neq H^s(y)$, we can let $x' = H^s(x), y' = H^s(y)$ such that (x', y') is a pair of collision of H since $H^s(H^s(x)) = H^s(H^s(y))$.

Therefore, H is not collision resistant, which contradicts the prerequisite. Thus, $\tilde{H}$ is a collision resistant hash function.