Anish Banerjee, Shankh Gupta **Problem Set - 2**
COL759: Cryptography September 2023

Problem 1: Cryptosystems secure against side-channel attacks

Solution: Consider the PRF $F' : \{0,1\}^{n+1} \times \{0,1\}^n \to \{0,1\}^n$

$$F'(k||b_k, x) = \begin{cases} F(k, 0^n)[1 \ldots n-1]||b_k & \text{if } x = 0^n \\ F(k, x) & \text{Otherwise} \end{cases}$$

In other words, the last bit of $F(k, 0^n)$ has been replaced with the last bit of the key.

(a) Let $\mathcal{A}$ be an adversary which breaks the PRF security of F' with non-negligible advantage ϵ. We will build a reduction $\mathcal{B}$ which breaks the PRF security of F with the same advantage.

Problem 1(a)

- Challenger picks a uniformly random bit $b \leftarrow \{0,1\}$ and a key $k \leftarrow \mathcal{K}$.
- $\mathcal{B}$ samples a random $b_k \leftarrow \{0,1\}$.
- The adversary $\mathcal{A}$ makes polynomially many queries $\{x_i\}$ to $\mathcal{B}$ who passes them to the challenger. Challenger replies as in the PRF Game.
- Upon receiving the response y_i of each query, $\mathcal{B}$ checks if $x_i = 0$. If so, it modifies y_i by exchanging its last bit with b_k. Otherwise, it just passes y_i to $\mathcal{A}$.
- After polynomially many queries, $\mathcal{B}$ forwards the response send by $\mathcal{A}$ (b') and wins if $b = b'$.

Figure 1: Reduction for Problem 1(a)

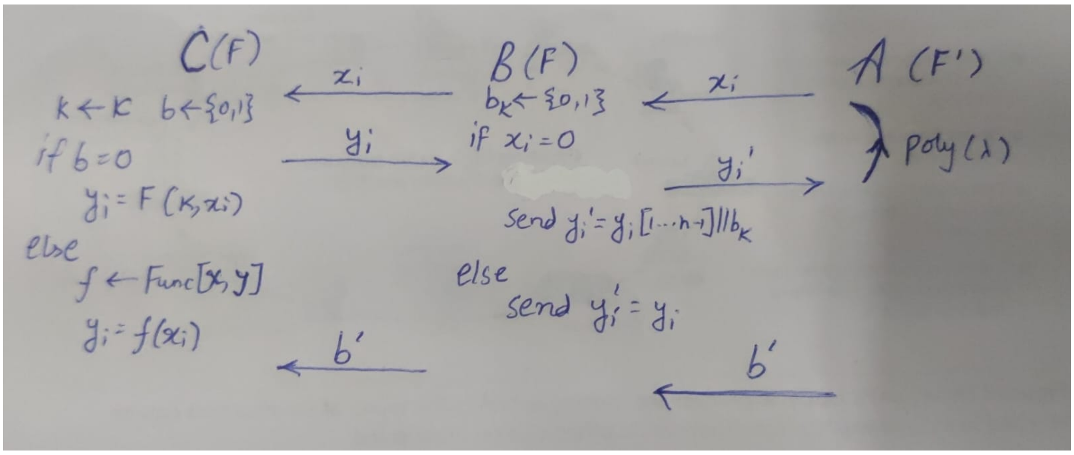

Figure 2: Image for Problem 1(a)

When the challenger chooses $b = 0$, the game is equivalent to the challenger choosing 0 in PRF game of F'.

$$\Pr[b' = 0|b = 0] = \Pr[\mathcal{A} \text{ outputs zero when the challenger chooses 0 in PRF game of } F']$$

When the challenger chooses $b = 1$, $\mathcal{A}$ receives the output of a random function for all $x_i \neq 0^n$. For $x_i = 0^n$, the output received is $r||b_k$. Since b_k is choosen randomly, this too is random.

$$\Pr[b' = 0|b = 1] = \Pr[\mathcal{A} \text{ outputs zero when the challenger chooses 1 in PRF game of } F']$$

Hence we can conclude,

$$\mathsf{PRFAdv}[\mathcal{B}, F] = \mathsf{PRFAdv}[\mathcal{A}, F']$$

(b) We will show that F' does not satisfy 1-leakage resilience by constructing an adversary $\mathcal{A}'$ who makes a leakage query for the last bit of the key and breaks F'.

- **Leakage Query:** $\mathcal{A}'$ makes a query for the last bit of the key and receives b_k from the challenger.
- **PRF Query:** $\mathcal{A}'$ queries for the $x = 0^n$ and receives y_i. He checks if the last bit of y_i is b_k. If yes it outputs $b' = 0$ (PRF), otherwise it outputs $b' = 1$ (Random Function).

From the game and definition of F', it is evident that:

$$\Pr[b' = 0 | b = 0] = 1$$

When the challenger chooses $b = 0$, the evaluation of a random function at 0^n can have its last bit as 0 or 1 with 1/2 probability. So,

$$\Pr[b' = 0 | b = 1] = \frac{1}{2}$$

And the advantage of $\mathcal{A}'$ is

$$\mathsf{PRFAdv}[\mathcal{A}, F'] = \Pr[b' = 0 | b = 0] - \Pr[b' = 0 | b = 1] = 1 - \frac{1}{2} = \frac{1}{2}$$

Which is non-negligible.

Problem 2 : MACs: unique queries vs non-unique queries

Solution:

(a) We propose the following MAC scheme $\mathcal{I}_{\text{uq}} = (\mathsf{Sign}_{\text{uq}}, \mathsf{Verify}_{\text{uq}})$ that is $\mathsf{UFCMA} - \mathsf{Unique}$ secure, but not UFCMA secure :

$$\mathsf{Sign}(k, m; r) = (r, F(k_1, r)||F(k_2, m \oplus r))$$

and

$$\mathsf{Verify}(k, m, (\sigma_0, \sigma_1)) = \begin{cases} 1 & \text{if } \sigma_1 = F(k_1, \sigma_0)||F(k_2, m \oplus \sigma_0) \\ 0 & \text{Otherwise} \end{cases}$$

where $k = (k_1, k_2) \in \{0,1\}^{2n}$, $m \in \{0,1\}^n$ and $r \in \{0,1\}^n$

(b) To prove the $\mathsf{UFCMA} - \mathsf{Unique}$ security of $\mathcal{I}_{\text{uq}}$, we first build the following sequence of hybrid games:

Game 0: This is the original security game.

- **Setup Phase :** Challenger chooses a PRF key $k = (k_1, k_2)$.
- **Query Phase :** Adversary sends polynomially many queries. For the i^{th} query m_i, the challenger chooses a random string $r_i \leftarrow \{0,1\}^n$ and sends $\sigma_i = \mathsf{Sign}(k, m_i; r_i)$ to the adversary.
- **Forgery :** Finally, the adversary outputs (m^*, σ^*) such that $(m^*, \sigma^*) \neq (m_i, \sigma_i)$ for all i.

Let the winning probability of Adversary $\mathcal{A}$ in Game 0 be p_0.

Game 1: In this game, Challenger replaces one of the PRFs with a random function.

- **Setup Phase :** Challenger chooses a PRF key $k = (k_1, k_2)$ and samples $f \leftarrow \mathsf{Func}[\mathcal{X}, \mathcal{Y}]$. (here $\mathcal{Y} = \{0,1\}^n$ and $\mathcal{X} = \{0,1\}^n$)
- **Query Phase :** Adversary sends polynomially many queries. For the i^{th} query m_i, the challenger chooses a random string $r_i \leftarrow \{0,1\}^n$ and sends $\sigma_i = (r_i, f(r_i)||F(k_2, m \oplus r_i))$ to the adversary.
- **Forgery :** Finally, the adversary outputs (m^*, σ^*) such that $(m^*, \sigma^*) \neq (m_i, \sigma_i)$ for all i.

Let the winning probability of Adversary $\mathcal{A}$ in Game 1 be p_1.

Game 2: In this game, Challenger replaces both the PRFs with random functions.

- **Setup Phase :** Challenger samples $f_1 \leftarrow \mathsf{Func}[\mathcal{X}, \mathcal{Y}]$ and $f_2 \leftarrow \mathsf{Func}[\mathcal{X}, \mathcal{Y}]$.
- **Query Phase :** Adversary sends polynomially many queries. For the i^{th} query m_i, the challenger chooses a random string $r_i \leftarrow \{0,1\}^n$ and sends $\sigma_i = (r_i, f(r_i)||f(m \oplus r_i))$ to the adversary.
- **Forgery :** Finally, the adversary outputs (m^*, σ^*) such that $(m^*, \sigma^*) \neq (m_i, \sigma_i)$ for all i.

Let the winning probability of Adversary $\mathcal{A}$ in Game 2 be p_2.

Claim 1 : $|p_0 - p_1|$ is negligible.

Proof: Let $\mathcal{A}$ be an adversary for which the claim is false. We will show a reduction algorithm $\mathcal{B}$ which uses $\mathcal{A}$ and has $|p_0 - p_1|$ advantage in the PRF game.

- The reduction algorithm receives signing queries $\{m_i\}$ from $\mathcal{A}$.
- It samples r_i and a key k and forwards it to the PRF challenger.
- It then receives y_i from the PRF challenger and calculates $\sigma_i = (r_i, y_i||F(k, m \oplus r_i))$. It sends the signature σ_i to the adversary.

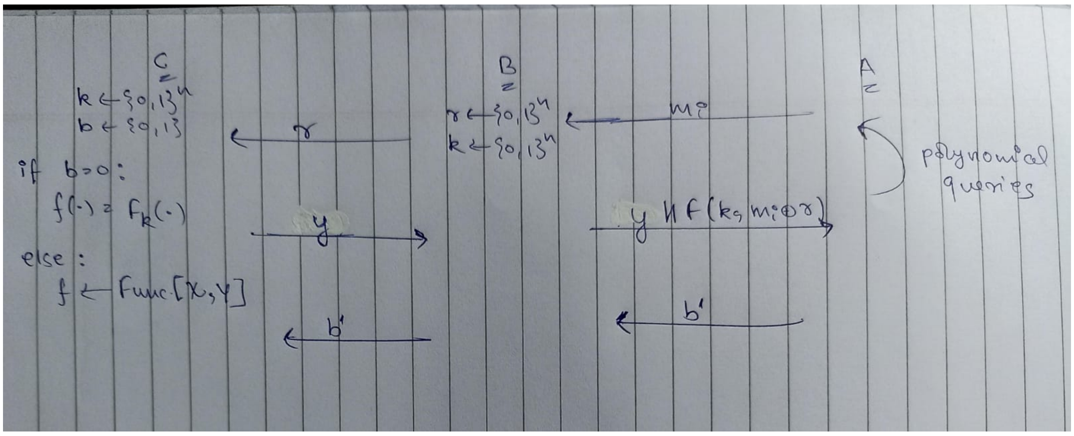

Figure 3: Reduction for Claim 1

- The adversary then guesses a bit b, where b = 0 for Game 0 and b = 1 for Game 1. The reduction $\mathcal{B}$ then forwards the bit b to the PRF challenger.

If the PRF challenger chooses PRF, this corresponds to Game 0 for the adversary $\mathcal{A}$ and it outputs 0 with a probability p_0. On the other hand, if PRF challenger chooses a random function, then it corresponds to Game 1 for the adversary and it outputs 1 with probability p_1. So

$$\mathsf{PRFAdv}[\mathcal{B}, F] = |p_0 - p_1|$$

Hence, $|p_0 - p_1|$ is negligible as it is given that F is a secure PRF.

Claim 2: $|p_1 - p_2|$ is negligible.

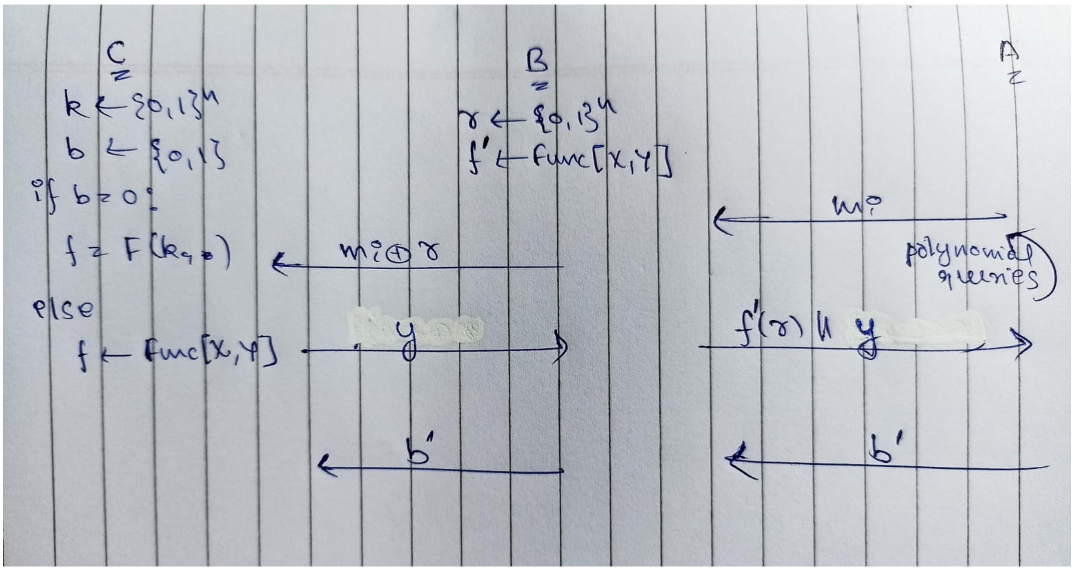

Figure 4: Reduction for Claim 2

Proof: Let $\mathcal{A}$ be an adversary for which the claim is false. We will show a reduction algorithm $\mathcal{B}$ which uses $\mathcal{A}$ and has $|p_1 - p_2|$ advantage in the PRF game..

- The reduction algorithm receives signing queries $\{m_i\}$ from $\mathcal{A}$.
- It samples r_i and a random function $f' \leftarrow \mathsf{Func}[\mathcal{X}, \mathcal{Y}]$. It then forwards $m_i \oplus r_i$ to the PRF challenger.
- It then receives y_i from the PRF challenger and calculates $\sigma_i = (r_i, f'(r_i)||y_i)$. It sends the signature σ_i to the adversary.

- The adversary then guesses a bit b, where b = 0 for Game 1 and b = 1 for Game 2. The reduction $\mathcal{B}$ then forwards the bit b to the PRF challenger.

If the PRF challenger chooses PRF, this corresponds to Game 1 for the adversary $\mathcal{A}$ and it outputs 0 with a probability p_1. On the other hand, if PRF challenger chooses a random function, then it corresponds to Game 2 for the adversary and it outputs 1 with probability p_2. So

$$\mathsf{PRFAdv}[\mathcal{B}, F] = |p_1 - p_2|$$

Hence, $|p_1 - p_2|$ is negligible as it is given that F is a secure PRF.

Claim 3: For any Adversary $\mathcal{A}$, $p_2 = \frac{1}{|\mathcal{Y}|^2}$. (note that, $|\mathcal{Y}| = 2^n$).

Proof : As, f_1 and f_2 are completely random functions, $f(x)$ for any x is equivalent to choosing a random output $y \in \mathcal{Y}$.

In order to win the game, the output by the adversary (m^*, σ^*) must satisfy $\sigma_1^* = f_1(r)$ and $\sigma_2^* = f_2(m_i \oplus r)$ (σ_1^* and σ_2^* are the first n and the last n bits of σ^* respectively). Since f_1 and f_2 are completely random (and (m^*, σ^*) can't be the same as any message-signature pair queried before), the probability of this happening would be $\frac{1}{|\mathcal{Y}|^2}$.

Thus, by all the above claims, we can prove that $\mathcal{I}_{\text{uq}}$ is a $\mathsf{UFCMA} - \mathsf{Unique\ MAC}$ scheme.

(c) We will show this using the following adversary:

- The adversary queries for m twice and receives

$$\sigma_1 = (r, F(k_1, r) || F(k_2, m \oplus r))$$

$$\sigma_2 = (r', F(k_1, r') || F(k_2, m \oplus r'))$$

- It then sends $(m \oplus r \oplus r', (r', F(k_1, r') || F(k_2, m \oplus r)))$ as the forgery.

Since $(m \oplus r \oplus r') \oplus r' = (m \oplus r)$, the above forgery is valid.

The only case where this forgery would not work is when both r and r' are same. This happens with a very small probability of $\frac{1}{2^n}$ and hence is negligible. So the adversary wins with a **probability of almost 1**. So, $\mathcal{I}_{\text{uq}}$ is not UFCMA secure scheme.

Problem 3 : A mistake in the lecture notes

Solution: According to the given flawed argument, for any (even unbounded) adversary $\mathcal{A}$ who wins the MAC game with verification queries ($\mathsf{MAC}^{\mathsf{vq}}$) with advantage ϵ, we can construct an adversary $\mathcal{B}$ who wins the MAC game without verification queries (MAC) with probability ϵ. However, we will show an adversary $\mathcal{A}'$ who wins *macvq* with advantage 1 but the reduction $\mathcal{B}$ cannot use it to win MAC.

The key observation here is that since every message has a unique signature, $\mathcal{B}$ cannot send a forgery of a message which it has already queried.

- $\mathcal{A}'$ sends verification queries $(\mathsf{Verify}, m, \sigma) \forall \sigma \in \mathcal{T}$ where $\mathcal{T}$ is the signature space.
- For the first verification query, $\mathcal{B}$ queries the challenger to obtain the signature $\sigma^\star$, and checks all the verification queries against this.

One of the queries by $\mathcal{A}'$ must be $(\mathsf{Verify}, m, \sigma^\star)$ and thus he wins the $\mathsf{MAC}^{\mathsf{vq}}$ game. However, $\mathcal{B}$ cannot use this forgery to win the MAC game since he has already queried it from the challenger.

Problem 4 : Even-Mansour instantiated with a bad permutation

Solution: The key observation here is that for any query x_i which results in an output y_i:

$$(y_i - k_2)(x_i + k_1) = 1 \mod p$$

So, we query the oracle at 3 points 0,1,2 and form three equations:

$$(y_0 - k_2)(0 + k_1) = 1 \mod p$$

$$(y_1 - k_2)(1 + k_1) = 1 \mod p$$

$$(y_2 - k_2)(2 + k_1) = 1 \mod p$$

On solving,

$$k_1 = 2(y_1 - y_2)(y_0 + y_2 - 2y_1)^{-1}$$

$$k_2 = y_1 - k_1(y_0 - y_1)$$

Now, we can just query π and check if these (k_1, k_2) satisfy $y_i = \pi(x_i + k_1) + k_2$ to distinguish it from a random permutation.

Note: all the additions and multiplications are modulo p

Problem 5 : 3-round Luby-Rackoff with inversion queries

Solution: To launch an attack against the 3-round Luby-Rackoff permutation, we will need 3 oracle queries.

- permute(x, y)
- permute(x, z)
- inverse_permute(α, β)

where x, y and z are some 16 bytes array and $y \neq z$. Let permute(x, y) = (γ_1, δ_1) and permute(x, z) = (γ_2, δ_2).

We will choose $\alpha = \gamma_2$ and $\beta = \delta_2 \oplus y \oplus z$. Let inverse_permute($\alpha$, β) = (m, n).

If $m = x \oplus \gamma_1 \oplus \gamma_2$ then the adversary outputs 0 (indicating PRP) else the adversary attack outputs 1 (indicating Completely random).

This attack works because for 3-round Luby-Rackoff

$$PRP(x, y) = (x \oplus F(k_2, y \oplus F(k_1, x)), y \oplus F(k_1, x) \oplus F(k_3, x \oplus F(k_2, y \oplus F(k_1, x))$$

and

$$PRP^{-1}(u, v) = (u \oplus F(k_2, v \oplus F(k_3, u), ...)$$

on substituting $(u, v) = (\alpha, \beta)$ we get $m = x \oplus F(k_2, y \oplus F(k_1, x)) \oplus F(k_2, z \oplus F(k_1, x))$ which is same as $x \oplus \gamma_1 \oplus \gamma_2$.

Problem 6 : CBC mode with bad initialization

Solution: Suppose the given ciphertext is (ct_0, ct_1, ct_2). Then

$$ct_0 = \mathsf{AES}(k, k \oplus m_0)$$

$$ct_1 = \mathsf{AES}(k, ct_0 \oplus m_1)$$

$$ct_2 = \mathsf{AES}(k, ct_1 \oplus m_2)$$

The attacker passes (ct_0, ct_0, ct_0) to the decrypt query and recieves (m'_0, m'_1, m'_2). Note that $m'_0 = m_0$ He can simply recover the key by

$$m'_1 = \mathsf{AES}^{-1}(ct_0) \oplus ct_0 = k \oplus m'_0 \oplus ct_0$$

$$\implies k = m'_1 \oplus m'_0 \oplus ct_0$$

Problem Part B : Coding Problem-Padding Oracle Attack

The concept here is similar to the padding oracle attack discussed in class. First, we find the padding of the message corresponding to the given ciphertext. For this, we begin from the end of the first 16 byte block of the ciphertext (after the initialization vector) and manipulate each byte till we get a padding error. For instance, in the example given in the problem statement:

$$m' = (10, 10, 10, 10, 10, 10, 10, 10, 10, \mathbf{10}, 11, 42, 33, 01, 89, 12)$$

If the highlighted byte is altered, then it will result in a padding error. But if the bytes after the highlighted one are altered, there will be no padding error.

Let the number of padding bytes be p. Now, to decipher the $(p+1)^{\text{th}}$ byte, we increase the padding bits by 1. This will result in a padding error. So, we check for which k

$$m'_{p+1} \oplus k = p + 1$$

When this happens, the padding error will stop since the first $p+1$ bytes have the value $p+1$. The message byte can be recovered simply as

$$m'_{p+1} = (p + 1) \oplus k$$

This process is continued till we get the entire message.