National Taiwan Normal University
CSIE Information Security

Instructor: Po-Wen Chi
Due Date: May 10, 2021, PM 11:59

Assignment 3

Policies:

- Zero tolerance for late submission.
- Please pack all your submissions in one zip file. **RAR is not allowed!!**
- I only accept **PDF**. MS Word is not allowed.
- Hand-writing is not allowed.
- Please use **Chinese.**

3.1 Authenticated Encryption (10 pts)

Authenticated encryption (AE) is a form of encryption which simultaneously assure the confidentiality and authenticity of data. That is, authenticated encryption can provide security against chosen ciphertext attack. Authenticated encryption schemes can recognize improperly-constructed ciphertexts and refuse to decrypt them. Generally speaking, an authenticated encryption can be treated as combining an encryption scheme and a message authentication code (MAC).

Let (E, D) be an AE-secure cipher. Consider the following derived ciphers:

- (E_1, D_1) are defined as follows:

$$E_1(k, m) := (E(k, m), E(k, m));$$

$$D_1(k, (c_1, c_2)) := \begin{cases} D(k, c_1), & \text{if } D(k, c_1) = D(k, c_2) \\ \text{reject}, & \text{otherwise} \end{cases}.$$

- (E_2, D_2) are defined as follows:

$$E_2(k, m) := (c \leftarrow E(k, m), \text{ output } (c, c));$$

$$D_2(k, (c_1, c_2)) := \begin{cases} D(k, c_1), & \text{if } c_1 = c_2 \\ \text{reject}, & \text{otherwise} \end{cases}.$$

Please show that (E_2, D_2) is also AE-secure but (E_1, D_1) is not.

3.2 Paillier Cryptosystem (15 pts)

The Paillier cryptosystem, invented by and named after Pascal Paillier in 1999, is a probabilistic asymmetric algorithm for public key cryptography. The scheme works as follows:

- **Key Generation**:
  - Choose two large prime numbers p and q randomly and independently of each other such that $\gcd(pq, (p-1)(q-1)) = 1$. This property is assured if both primes are of equal length.
  - Compute $n = pq$ and $\lambda = \text{lcm}(p-1, q-1)$.
  - Select random integer g where $g \in \mathbb{Z}^*_{n^2}$.
  - Ensure n divides the order of g by checking the existence of the following modular multiplicative inverse:

    $$\mu = (L(g^\lambda \mod n^2))^{-1} \mod n,$$

    where

    $$L(x) = \frac{x-1}{n}.$$

    Here the notation $\frac{a}{b}$ means the **quotient**.
  - The public key is (n, g).
  - The private key is (λ, μ).
- **Encryption**:
  - Let m be a message to be encrypted.
  - $r \xleftarrow{R} \mathbb{Z}^*_n$.
  - The ciphertext $c = g^m \cdot r^n \mod n^2$.
- **Decryption**:
  - Given a ciphertext c, the message m is

    $$m = L(c^\lambda \mod n^2) \cdot \mu \mod n.$$

There are two problems.

1. Please show that the decryption works.
2. Given a ciphertext c, please show how to re-randomize c without the private key. That is, how to generate c' from c where c' and c are encrypted from the same message.

3.3 Fugisaki-Okamoto Commitment (15 pts)

A commitment scheme is a cryptographic primitive that allows one to commit to a chosen value (or chosen statement) while keeping it hidden to others, with the ability to reveal the committed value later. Commitment schemes are designed so that a party cannot change the value or statement after they have committed to it: that is, commitment schemes are binding.

For example, you want to play an online *Rock,Paper,Scissors* with your friend. After you make your choice, you can use the scheme to generate a commitment and send the commitment to your friend. The commitment releases nothing about your choice. Now your friend give you its choice in plaintext. You also open your choice and show that this choice satisfy the previous commitment. Note that it is computationally impossible for you to change your choice that can still satisfy the existing commitment.

How to make a commitment? I will show you one commitment scheme proposed by Fugisaki and Okamoto. There are three entities here, **an authority, a prover, and a verifier**. The authority is the one who setups the environment. The prover is the one who generates the commitment and opens its message later. The verifier is the one to check if the opened message satisfies the previous commitment. The algorithms are listed as follows.

- **Setup**: Construct a group $\mathbb{G}$ with a generator g. The algorithm randomly picks k as the secret key and publishes $\{\mathbb{G}, h = g^k\}$ as the public key.
- **Prove**: Given a message m, the algorithm randomly picks r and generates the commitment c as follows:

$$c = g^m \cdot h^r.$$

- **Open**: The algorithm simply returns (m, r). It is trivial to check if (m, r) satisfies the commitment.

Note that k is only known by the authority and is kept secret to the prover and the verifier.

1. Please show that the commitment release no information about m.
2. Please show that it is computationally impossible for the prover to generate another m that can satisfy the existing commitment.

3.4 Davies-Meyer Compression Function (10 pts)

Please prove that the Davies-Meyer compression function is collision resistant.

3.5 The Cascade Problem of CMAC without Last Stage (10 pts)

In this class, I have shown you that NMAC has the cascade problem if there is no last stage. How about CMAC? If CMAC is like the operation shown in figure 3.1, what will happen? Please find a way to forge a signature in the game model.

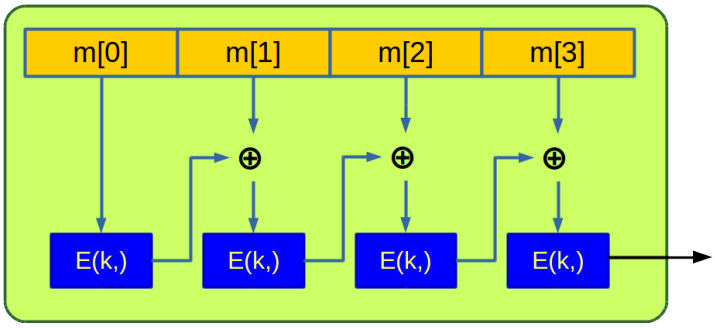

FIGURE 3.1: Wrong CMAC.

3.6 Programming: RSA Oracle Attack (15 pts)

In this class, I have introduced how RSA works. Now let's play a game. I will provide an oracle that can help you to decrypt only one bit in the message. Can you use this tool to break the ciphertext?

- http://140.122.185.210:8080/
- http://140.122.185.210:8080/oracle/xxx

PS. For your simplicity, we only use the textbook RSA.

3.7 Lab: MD5 (15 pts)

In this class, I have told you that you should not use MD5 anymore. Let's see how to make a MD5 collision pair for two executable binaries.

- Lab: https://seedsecuritylabs.org/Labs_20.04/Files/Crypto_MD5_Collision/Crypto_MD5_Collision.pdf

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits.