CS698G: Assignment #2

Aaryan Maheshwari
maaryan23@iitk.ac.in

Indian Institute of Technology, Kanpur

Introduction

This file contains my submission for the part 1 of assignment 2.

1 RSA Encryption

A manager selects two large primes, p and q, where $p \neq q$, to set up a secure communication channel between three employees, Alice, Bob, and Carlo, using RSA encryption. He first computes $n = pq$ and corresponding $\phi(n)$. Then, he uses this $\langle p, q \rangle$ pair to generate three RSA key pairs (with different e being relatively prime to each other and corresponding d), assigns them to the staff individually, and destroys $p, q, \phi(n)$ immediately afterwards (meaning nobody knows the values). The following lists the key pairs:

Alice: $\langle n, e_A \rangle,\ \langle n, d_A \rangle$
Bob: $\langle n, e_B \rangle,\ \langle n, d_B \rangle$
Carlo: $\langle n, e_C \rangle,\ \langle n, d_C \rangle$

Answer the following questions with detailed justification.

Question (a)

Alice wants to send a message M to both Bob and Carlo, so she calculates $C_B = M^{e_B} \bmod n$ and $C_C = M^{e_C} \bmod n$. Explain how an adversary can recover the message M without knowing private keys d_B and d_C.

Solution:

Since Alice encrypts the same message M with two different exponents e_B and e_C under the same modulus n, the adversary can exploit the common modulus attack (also called the common modulus vulnerability in RSA).

The adversary has:

$$C_B = M^{e_B} \bmod n$$
$$C_C = M^{e_C} \bmod n$$

Since $\gcd(e_B, e_C) = 1$, the adversary can compute integers x and y such that

$$xe_B + ye_C = 1$$

using the extended Euclidean algorithm.

Then the adversary computes:

$$M = C_B^x \cdot C_C^y \bmod n$$

If $x < 0$, compute C_B^{-x} as the modular inverse of $C_B^{|x|}$ modulo n.
If $y < 0$, compute C_C^{-y} as the modular inverse of $C_C^{|y|}$ modulo n.
Thus, without knowing d_B or d_C, the adversary can recover M.

Question (b)

Carlo is interested in messages sent to Alice. Outline a strategy that may help Carlo recover an alternative private key d'_A that can perform the same function as d_A to decrypt messages sent to Alice.

Solution:

1. Carlo knows his own key pair $\langle n, e_C \rangle$ and $\langle n, d_C \rangle$. Since
$$e_C d_C \equiv 1 \pmod{\phi(n)},$$
he computes
$$K = e_C d_C - 1$$
which satisfies $K = m\phi(n)$ for some integer m.
2. Carlo solves
$$e_A d'_A \equiv 1 \pmod{K}$$
by finding the modular inverse of e_A modulo K (e.g. via the extended Euclidean algorithm). This d'_A satisfies
$$e_A d'_A = jK + 1 = (jm)\phi(n) + 1$$
so
$$e_A d'_A \equiv 1 \pmod{\phi(n)}.$$

Note that there is a small caveat in this method: The attack succeeds iff $\gcd(e_A, K) = 1$, but this usually holds because e_A is chosen coprime to $\phi(n)$.

2 Hash Functions

Question (a)

Consider the following hash function based on RSA function. The key $\langle n, e \rangle$ is known to the public. A message M is represented by blocks of predefined fixed size $\{M_1, M_2, M_3, \ldots, M_m\}$ such that $0 \leq M_i < n$. We can assume that n is large enough to hold the RSA assumptions. The hash value is calculated by:
$$H(M) = ((M_1 \oplus M_2 \oplus \cdots \oplus M_m)^e) \bmod n$$
Does this hash function satisfy each of the following requirements? Justify your answers. You can give examples, if necessary, to support your arguments.

i. Fixed output size
ii. Efficiency (easy to calculate)
iii. Preimage resistant
iv. Second preimage resistant
v. Collision resistant

Solution:

i. **Fixed output size**
Yes, the output is fixed size. The result of $H(M)$ is an integer in the range $0 \leq H(M) < n$. Since n is fixed and public, the output size is fixed (determined by the bit length of n).

ii. **Efficiency (easy to calculate)**

No, it is inefficient compared to standard hash functions. The calculation involves modular exponentiation:

$$(M_1 \oplus M_2 \oplus \cdots \oplus M_m)^e \bmod n$$

which is computationally expensive, especially when e is large, as in typical RSA settings. Standard hash functions use lightweight operations (bitwise, additions, etc.).

iii. **Preimage resistant**

No, not secure. To find a preimage M for a given hash value h, it is sufficient to compute

$$X = h^d \bmod n$$

where d is the private RSA key. If d is known, preimages can be computed efficiently. Since n, e are public, an attacker with d (or with ability to compute RSA inverses) can break preimage resistance. Even without d, since the domain is just an XOR of message blocks, so it's easy to search.

iv. **Second preimage resistant**

No, not secure. Given M producing hash h, an attacker can choose M' such that

$$M_1 \oplus M_2 \oplus \cdots \oplus M_m = M'_1 \oplus M'_2 \oplus \cdots \oplus M'_m$$

This leads to the same hash value:

$$H(M) = H(M')$$

Hence, second preimages can be easily found.

v. **Collision resistant**

No, not collision resistant. Because the hash input is reduced to the XOR of all blocks before exponentiation, different combinations of message blocks yielding the same XOR produce collisions:

$$M_1 \oplus \cdots \oplus M_m = M'_1 \oplus \cdots \oplus M'_m \Rightarrow H(M) = H(M')$$

Finding two different sequences with the same XOR is trivial.

Question (b)

Explain how to efficiently find collisions in the following hash functions:

i. The function $H_a : \{0,1\}^{512} \to \{0,1\}^{256}$ is defined as follows:

$$H_a(x, y) = F(y, x \oplus y) \oplus y$$

Let the pair F, F^{-1} be a public secure symmetric key block cipher with block size and key length 256. That is, y is interpreted as the symmetric key, and $x \oplus y$ is the plaintext for F. To compute H_a, we first XOR y with x, then apply the block cipher to the result, and finally XOR the block cipher output with y one more time to get the final output.

ii.

$$H_b = F(y \oplus x, x)$$

where F, F^{-1} is as in the last question (i). $y \oplus x$ is the key and x is the plaintext.

iii. $H_c : \{0,1\}^{257} \to \{0,1\}^{256}$ is defined as follows: Let $H : \{0,1\}^* \to \{0,1\}^{256}$ be a collision-resistant hash function for arbitrary-length messages. Then for $x||b \in \{0,1\}^{257}$,

$$H_c(x, b) = \begin{cases} H(x) & \text{if } b = 0 \\ H(H(x)) & \text{if } b = 1 \end{cases}$$

where $a||b$ refers to concatenating a and b together as a single string.

Solution:

i. Given

$$H_a(x, y) = F(y, x \oplus y) \oplus y$$

Choose any y and any x. Let

$$H_a(x, y) = h$$

Pick $x' = x \oplus y \oplus y'$:

$$H_a(x', y') = F(y', x' \oplus y') \oplus y' = F(y', x \oplus y) \oplus y'$$

Since $F(y', x \oplus y) = F(y, x \oplus y)$ if $y' = y$, so setting $y' = y$ and changing x accordingly yields collision.

ii. Given

$$H_b = F(y \oplus x, x)$$

Choose arbitrary x, y. Let

$$H_b(x, y) = h$$

Pick x' and y' such that

$$y' \oplus x' = y \oplus x \quad \text{and} \quad x' = x$$

or more generally, choose (x', y') where $y' \oplus x' = y \oplus x$ and $x' \neq x$.

iii. Given

$$H_c(x, b) = \begin{cases} H(x) & b = 0 \\ H(H(x)) & b = 1 \end{cases}$$

Find x_1, x_2 such that

$$H(x_1) = H(x_2)$$

Then

$$H_c(x_1, 0) = H(x_1) = H(x_2) = H_c(x_2, 0)$$

or

$$H_c(x_1, 1) = H(H(x_1)) = H(H(x_2)) = H_c(x_2, 1)$$

If no such x_1, x_2 exist, no collision. But if any collision for H is found, use it here.

3 ElGamal

Question (a)

ElGamal Encryption:

A variant of ElGamal crypto system over the prime field $GF(q)$ is given as follows. Assume the parameters are as discussed in the lectures. Let $y_A = a^{x_A} \bmod q$, be the public address of Alice, where x_A, $1 < x_A < q - 1$, is Alice's private key. The encryption function is defined as follows:

$$E(M) = [C_1, C_2],$$

where $C_1 = a^k \bmod q$, k is a random integer $1 \leq k \leq q - 1$ and $C_2 = K \oplus M$, where $K = y_A^k \bmod q$ and $\oplus$ is binary XOR applied to binary representation of K and M.

i. Describe the Decryption Function $D(C_1, C_2)$ that Alice can use to recover the message.

ii. Show how the security of the encryption function is based on Computational Diffie-Hellman (CDH) problem.
CDH Problem: Let q be a prime number and a be a generator of the multiplicative cyclic group of modulo q. Given a^x, a^y , the CDH problem is to compute a^{xy}. .

Solution:

i. Given the ciphertext (C_1, C_2), Alice performs the following steps:

$$K = C_1^{x_A} \bmod q$$

$$M = C_2 \oplus K$$

where $\oplus$ denotes bitwise XOR of the binary representation of C_2 and K.

ii. In ElGamal encryption:

$$C_1 = a^k \bmod q$$

$$y_A = a^{x_A} \bmod q$$

The session key is:

$$K = y_A^k \bmod q = (a^{x_A})^k \bmod q = a^{x_A k} \bmod q$$

An attacker, given $C_1 = a^k$ and $y_A = a^{x_A}$, must compute:

$$a^{x_A k} \bmod q$$

This is equivalent to solving the Computational Diffie-Hellman (CDH) problem:

$$\text{Given } a^x, a^y, \text{ compute } a^{xy} \bmod q$$

Thus, breaking the encryption requires solving CDH, which is assumed to be computationally hard.

Question (b)

ElGamal signature:

Let's consider a variant of ElGamal signature over the prime field $GF(q)$. Let H be a public hash function and let $y_A = a^{x_A} \bmod q$ be the public key of Alice, where $x_A, 1 < x_A < q-1$ is the private key and a is a primitive element in the field. Alice uses the following equation to define a related ElGamal signature scheme by using:

$$mS_2 + x_A S_1 = k \bmod (q-1)$$

where $m = H(M)$, M an arbitrary message, k a random number, $S_1 = a^k \bmod q$ and S_2 are signature parameters. The signature for a message M is represented as $[M, S1, S2]$.

i. What are the signing and verification equations?

ii. Is this scheme secure? Justify your answer with reasons.

Solution:

i. **Signing:**

1. Alice selects a random k such that $1 \leq k \leq q-2$ and $\gcd(k, q-1) = 1$.
2. Computes

$$S_1 = a^k \bmod q$$

3. Computes

$$S_2 = (k - x_A S_1)m^{-1} \bmod (q-1)$$

Signature:
The signature is (S_1, S_2) on message M.

Verification:
Given M, S_1, S_2, and public key y_A, the verifier computes

$$m = H(M)$$

and checks if

$$a^k \equiv S_1 \pmod q$$

where

$$k \equiv mS_2 + x_A S_1 \pmod{q-1}$$

ii. This scheme relies on the following assumptions:

(a) **Hash Function Preimage Resistance:** The security assumes H is collision-resistant and preimage-resistant, so forging signatures for arbitrary M is infeasible without knowing k or x_A.

(b) **Discrete Logarithm Hardness:** Recovering x_A from $y_A = a^{x_A} \bmod q$ is assumed hard under the Discrete Logarithm Problem (DLP). Since the verification equation effectively hides k via a modular equation, recovering it without solving the DLP is infeasible.

(c) **Reusing k Breaks Security:** If k is reused across two messages m_1 and m_2, then given:

$$m_1 S_2^{(1)} + x_A S_1 = k \mod (q-1)$$

$$m_2 S_2^{(2)} + x_A S_1 = k \mod (q-1)$$

subtracting yields:

$$(m_1 - m_2) S_2^{(1)} \equiv (m_2 S_2^{(2)} - m_1 S_2^{(1)}) \pmod{q-1}$$

allowing recovery of x_A or k. Hence, randomness and uniqueness of k are critical.

4 Needham–Schroeder protocol

We studied the Needham–Schroeder protocol in lectures. An alternative key distribution method suggested by a network vendor is illustrated in the figure below.

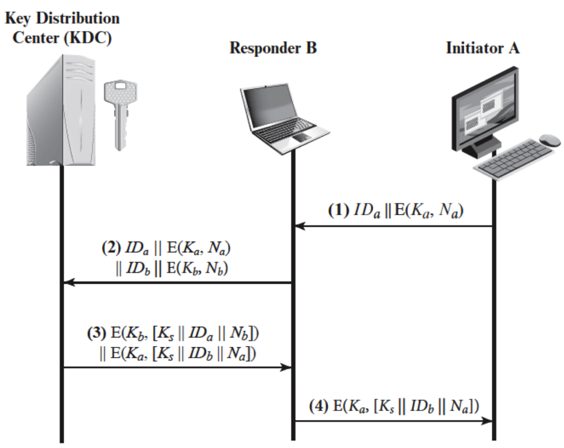

Figure 1: Key Distribution Between A and B

Question (a)

How do A and B know that the key is freshly generated?

Solution:

A and B know that the key is freshly generated because the messages they receive from the Key Distribution Center (KDC) include their respective nonces:

- The KDC encrypts the session key K_S together with the nonce N_A (for A) and N_B (for B).
- Since A and B had generated and sent these nonces in their requests, the presence of their own nonce in the KDC's response proves that the message is a fresh response to their request and not a replay.
- The encrypted message
$$E(K_A, [K_S \| ID_B \| N_A])$$
assures A that K_S was generated in response to the specific request containing N_A.
- Similarly,
$$E(K_B, [K_S \| ID_A \| N_B])$$
assures B that K_S is fresh with respect to their own request containing N_B.

Thus, inclusion of the nonces N_A and N_B ensures freshness of the key.

Question (b)

How could A and B know that the key is not available to other users in the system?

Solution:

A and B know that the key is not available to other users in the system because:

- The session key K_S is generated by the trusted Key Distribution Center (KDC), which is assumed not to disclose K_S to any unauthorized party.
- The session key K_S is transmitted to A encrypted under K_A:
$$E(K_A, [K_S \| ID_B \| N_A])$$
and to B encrypted under K_B:
$$E(K_B, [K_S \| ID_A \| N_B])$$
Since only A can decrypt messages encrypted with K_A and only B can decrypt messages encrypted with K_B, no other users can obtain K_S.
- The secrecy of K_S relies on the assumption that K_A and K_B are securely shared only between the KDC and the respective users.

Question (c)

At this stage, A and B cannot authenticate with each other. Explain why and extend the scheme with a few steps so that A and B can authenticate with each other. Your modifications should be based on symmetric key methods used in this key distribution protocol, not public key primitives.

Solution:

At this stage, A and B cannot authenticate with each other because:

- A has received K_S encrypted with K_A and B has received K_S encrypted with K_B.
- Neither A nor B has verified that the other party actually possesses K_S.
- There is no proof exchanged between A and B that they are communicating with the intended party rather than an attacker.

We can extend the protocol using a symmetric key challenge-response mechanism as follows:

(a) A generates a random nonce N_1 and sends it to B encrypted under K_S:

$$A \to B : E(K_S, N_1)$$

(b) B decrypts, generates its own random nonce N_2, and sends back:

$$B \to A : E(K_S, N_1 - 1 \| N_2)$$

where $N_1 - 1$ proves B's knowledge of N_1.

(c) A sends:

$$A \to B : E(K_S, N_2 - 1)$$

to confirm possession of K_S.

This ensures both A and B demonstrate knowledge of K_S and authenticate each other using symmetric key operations only.