CSCI971/CSCI471 Modern Cryptography Assignment 2

Name: Karan Goel
Student Number: 7836685

October 20, 2024

1 Task A: Ring Signatures in Privacy-Preserving Cryptocurrencies

Ring signatures are cryptographic signatures designed for privacy preservation, allowing a signer (the sender) to generate a signature on behalf of a group of potential signers without disclosing the actual identity of the member who signed.

In the realm of cryptocurrencies, ring signatures are utilized by platforms like Monero. They obscure the sender's identity by mixing their transaction with those of other users, making it challenging to trace the transaction back to a specific individual.

1.1 How Ring Signature Works

The signature scheme assumes that all parties agree on a cyclic group G of order q, a generator g of G, and a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$.

- **KeyGen(λ)**: Taking as input a security parameter λ, the probabilistic polynomial-time (P.P.T.) algorithm:
  1. Chooses a uniform $x \in \mathbb{Z}_q$ and computes $h = g^x$.
  2. The public key is h, and the private key is x.
- **Sign(sk, M, $(h_1, \ldots, h_l)$)**: Taking as input a message M, a set of public keys $(h_1, \ldots, h_l)$, and a secret key $sk = x_i$, the P.P.T. algorithm:
  1. Chooses a random number r_i.
  2. Computes $R_i = g^{r_i}$.
  3. For each $j \in [1, l]$ where $j \neq i$:
     - Chooses random c_j and z_j.
     - Computes $R_j = \frac{g^{z_j}}{h_j^{c_j}}$.

4. Computes $c = H(R_1, \ldots, R_l, M)$.
5. Computes $c_i = c - \sum_{j \neq i} c_j$.
6. Computes $z_i = r_i + c_i \cdot x_i \mod q$.
7. The signature is $((R_1, c_1, z_1), \ldots, (R_l, c_l, z_l))$.

- **Verify(S, M, $(h_1, \ldots, h_l)$)**: Taking as input a signed message M, a set of public keys $(h_1, \ldots, h_l)$, and a signature $((R_1, c_1, z_1), \ldots, (R_l, c_l, z_l))$, the P.P.T. algorithm:

1. Computes $c' = H(R_1, \ldots, R_l, M)$.
2. Accepts the signature if:
   - $c = \sum_{j=1}^{l} c_j \mod q$,
   - $g^{z_j} = R_j h_j^{c_j}$ for all j.

1.2 Cryptocurrency Transaction with Ring Signatures

A cryptocurrency platform utilizes ring signatures to enhance the privacy of its users during transactions.

Alice, Bob, Charlie, and David are all users of the cryptocurrency platform.

1. **Key Generation:**
   - Each user generates their own public/private key pair.
     - Alice: (PK_A, SK_A)
     - Bob: (PK_B, SK_B)
     - Charlie: (PK_C, SK_C)
     - David: (PK_D, SK_D)
2. **Making a Transaction:**
   - Alice wants to send 10 coins to a merchant but does not want anyone to know that she is the one making the transaction.
   - To preserve her privacy, Alice creates a ring signature that includes her own key (SK_A) along with the public keys of Bob, Charlie, and David, forming a ring of public keys: PK_A, PK_B, PK_C, PK_D.
3. **Creating the Ring Signature:**
   - Alice signs the transaction message, which includes details such as the amount (10 coins), the recipient's address, and the ring of public keys.
   - The ring signature proves that the transaction was authorized by one of the members in the group (Alice, Bob, Charlie, or David) without revealing which member actually signed it.
4. **Broadcasting the Transaction:**

- Alice broadcasts the transaction along with the ring signature to the cryptocurrency network.

5. **Verification by the Network:**
   - Miners and nodes in the network receive the transaction. They verify the ring signature using the public keys of Alice, Bob, Charlie, and David.
   - The network confirms that the signature is valid, meaning one of the members of the group approved the transaction, but it cannot tell which member it was.

Key Benefits of Ring Signatures in Cryptocurrency:

- **Anonymity:** The transaction obscures the sender's identity, providing privacy for users like Alice. This prevents potential tracking of her spending habits or balances by outsiders.
- **Security:** The transaction is still verifiable, ensuring that only legitimate members of the group can authorize transactions without exposing their identities.
- **Enhanced Privacy:** Users can transact without fear of exposing their financial activity, making the cryptocurrency more appealing for privacy-conscious individuals.

2 Task B: Recover the Secret Key of Schnorr Signature Given Two Signatures

2.1 Given:

- Two Schnorr signatures:
  - (R, z_1) for message M_1
  - (R, z_2) for message M_2
- The public key P corresponding to the secret key k.
- The random nonce r used to compute R:

$$R = g^r$$

where g is the generator of the elliptic curve.

2.2 Steps to Recover the Secret Key:

1. **Signature Definition:** In the Schnorr signature scheme, the signature (R, s) for a message M is defined as:

$$s = r + k \cdot H(M, R) \mod q$$

where H is a hash function, k is the secret key, q is the order of the group, and r is the random nonce.

2. **Set Up the Equations:** For each signature, we can write the following equations:

$$z_1 = r + k \cdot H(M_1, R) \mod q$$

$$z_2 = r + k \cdot H(M_2, R) \mod q$$

3. **Subtract the Equations:** Subtract the two equations to eliminate r:

$$z_1 - z_2 = (k \cdot H(M_1, R) - k \cdot H(M_2, R)) \mod q$$

Factor out k:

$$z_1 - z_2 = k \cdot (H(M_1, R) - H(M_2, R)) \mod q$$

Let $\Delta H = H(M_1, R) - H(M_2, R)$. The equation simplifies to:

$$z_1 - z_2 = k \cdot \Delta H \mod q$$

4. **Solve for k:** To find k, rearrange the equation and compute the modular inverse of ΔH in the finite field defined by q. This step is valid only if ΔH is non-zero.

$$k = \frac{z_1 - z_2}{\Delta H} \mod q$$

5. **Final Recovery of the Secret Key k:** The secret key k can be recovered using:

$$k = (z_1 - z_2) \cdot (\Delta H^{-1} \mod q) \mod q$$

2.3 Conditions:

- The same nonce R should be used for computing signatures. If different nonces were used, this method wouldn't work.
- The subtraction and recovery rely on ΔH not being zero (i.e., $H(M_1, R) \neq H(M_2, R)$).

3 Task C: Variant of IBS for multiple identities

3.1 Syntax for Variant IBS

This variant of IBS includes the following algorithms:

- **Setup(λ)**:
  This algorithm takes as input a security parameter λ and outputs:
  - A *master public key* MPK,
  - A *master secret key* MSK.

- **KeyGen**($MPK, MSK, \{ID_1, ID_2, \ldots, ID_n\}$):
  This is the key generation algorithm run by the Private Key Generator (PKG). Given the master public key MPK, master secret key MSK, and a set of identities $\{ID_1, ID_2, \ldots, ID_n\}$, the PKG generates a *single accumulated private key* $SK_{\{ID_1, ID_2, \ldots, ID_n\}}$ for the set of identities. The same private key can be used to sign messages for any identity in the set.

- **Sign**($M, MPK, SK_{\{ID_1, \ldots, ID_n\}}, ID$):
  The signing algorithm takes as input the master public key MPK, the accumulated private key $SK_{\{ID_1, ID_2, \ldots, ID_n\}}$, an identity $ID \in \{ID_1, \ldots, ID_n\}$, and a message M. It outputs a *signature* σ on the message M for the identity ID, provided that ID is within the set $\{ID_1, \ldots, ID_n\}$.

- **Verify**(M, σ, MPK, ID):
  The verification algorithm takes as input the master public key MPK, an identity ID, a message M, and a signature σ. It outputs a Boolean value indicating whether the signature is valid for the given identity and message.

3.2 Correctness

The correctness of the scheme requires the following condition to be satisfied:

> For any $ID \in \{ID_1, ID_2, \ldots, ID_n\}$, if a valid signature σ is generated using the accumulated private key $SK_{\{ID_1, ID_2, \ldots, ID_n\}}$ for a message M, then the verification algorithm should output **True** when verifying (MPK, ID, M, σ).

Formally, this can be expressed as:

$$\text{If } \sigma = \text{Sign}(MPK, SK_{\{ID_1, ID_2, \ldots, ID_n\}}, ID, M) \text{ and } ID \in \{ID_1, ID_2, \ldots, ID_n\},$$

then:

$$\Pr[\text{Verify}(M, \sigma, MPK, ID) = 1] = 1,$$

where $\text{Verify}(M, \sigma, MPK, ID) = 1$ indicates that the signature is valid.

4 Task D: IND-CPA of Variant of El Gamal

4.1 Scheme Overview

- **Key Generation:**
  - Choose two groups G and G_T of order q, where a pairing function $e : G \times G \to G_T$ exists.
  - Choose a generator $g \in G$ and a secret key $x \in \mathbb{Z}_q$.
  - Compute $h = g^x \in G$.
  - Public key: (G, G_T, e, q, g, h).

- Private key: (G, G_T, e, q, g, x).

- **Encryption:**
  - To encrypt a message $m \in G$:
    1. Choose a random $r \in \mathbb{Z}_q$.
    2. Compute the ciphertext $(c_1, c_2) = (g^r, h^r \cdot m) = (g^r, g^{xr} \cdot m)$.

- **Decryption:**
  - Given a ciphertext $(c_1, c_2) = (g^r, g^{xr} \cdot m)$:
    1. Compute $c_1^x = (g^r)^x = g^{xr}$.
    2. Recover the message as $m = c_2 \cdot (c_1^x)^{-1} = (g^{xr} \cdot m) \cdot (g^{xr})^{-1} = m$.

4.2 IND-CPA Attack

To show that the scheme is not IND-CPA secure, we can present an attack in which an adversary distinguishes between two chosen plaintexts m_0 and m_1.

1. **Challenge Setup:**
   - The adversary selects two messages $m_0, m_1 \in G$ and submits them to the challenger.
   - The challenger randomly chooses a bit $b \in \{0, 1\}$ and encrypts m_b using the encryption scheme. The ciphertext given to the adversary is $(c_1, c_2) = (g^r, g^{xr} \cdot m_b)$, where r is randomly chosen.

2. **Adversary's Observation:**
   - The adversary has access to the ciphertext $(c_1, c_2) = (g^r, g^{xr} \cdot m_b)$.
   - The adversary can compute the pairing $e(c_1, h) = e(g^r, g^x) = e(g, g)^{xr}$, which is the same as $e(c_1, h) = e(c_1, g^x)$.

3. **Attack:**
   - The adversary now tries to distinguish between m_0 and m_1.
   - The encryption scheme leaks information through the pairing operation.
   - The adversary can compute $e(c_1, g) = e(g^r, g) = e(g, g)^r$.
   - This pairing gives the adversary access to $e(g, g)^r$, which is independent of the encrypted message m_b.
   - The adversary can compare c_2 with $e(g, g)^r \cdot m_0$ and $e(g, g)^r \cdot m_1$ to check which one matches.

4.3 Conclusion

Since the adversary can compute pairings that allow them to distinguish between the two chosen plaintexts m_0 and m_1 with non-negligible probability, the scheme is not IND-CPA secure.

5 Task E: Message Proof

5.1 Proof Outline

To prove that the message M is either g^{100} or g^{200}, we can employ a zero-knowledge proof. This proof will convince the verifier that M is one of the two possible values, without revealing which one it is or any other information about the message.

5.2 Commitment Phase

- **Encryption:** The prover encrypts both possible values $M_1 = g^{100}$ and $M_2 = g^{200}$ using the same random value r:

$$CT_1 = (C_1^1, C_2^1) = (g^r, h^r \cdot g^{100})$$

$$CT_2 = (C_1^2, C_2^2) = (g^r, h^r \cdot g^{200})$$

- **Commitments:** The prover creates commitments for C_1^1 and C_1^2 to hide the choice of r.

5.3 Challenge Phase

The verifier selects a random challenge $b \in \{1, 2\}$. The challenge asks the prover to show that the ciphertext $CT = (C_1, C_2)$ corresponds to either CT_1 or CT_2.

5.4 Response Phase

The prover responds based on the challenge b:

- **If the challenge is** $b = 1$ (the prover must prove that $M = g^{100}$):
  - The prover reveals the randomness r used for encrypting g^{100}.
  - The prover sends r to the verifier.
  - The verifier checks the validity by verifying:

$$C_1 = g^r \quad \text{and} \quad C_2 = h^r \cdot g^{100}$$

  - If both equations hold true, then the proof is valid for $M = g^{100}$.
- **If the challenge is** $b = 2$ (the prover must prove that $M = g^{200}$):
  - The prover reveals the randomness r used for encrypting g^{200}.
  - The prover sends r to the verifier.
  - The verifier checks the validity by verifying:

$$C_1 = g^r \quad \text{and} \quad C_2 = h^r \cdot g^{200}$$

  - If both equations hold true, then the proof is valid for $M = g^{200}$.

5.5 Verification Phase

The verifier ensures that the ciphertext CT corresponds to either CT_1 or CT_2 based on the challenge. The verifier confirms that:

- If $b = 1$, then the prover has correctly demonstrated that $M = g^{100}$.
- If $b = 2$, then the prover has correctly demonstrated that $M = g^{200}$.

5.6 Zero-Knowledge Property

This proof construction has the following zero-knowledge properties:

- **No Additional Information:** The verifier learns nothing about r or which specific M was chosen, as they only validate the correctness of the ciphertext against the challenge without gaining insights into the values used in the encryption.
- **Indistinguishability:** Since the prover only reveals information pertaining to the challenge, the process ensures that the proof remains indistinguishable from one where the prover is simply guessing M.

Thus, the prover successfully demonstrates that M is either g^{100} or g^{200}, without revealing any other information.