Study of ID-based Ring Signatures and Implementation of Lattice-based Merkle Tree Accumulator

Thesis to be submitted in partial fulfillment of the requirements for the degree

of

Master of Science

by

Pritam Mondal
23MA40024

Under the guidance of

Prof. Sourav Mukhopadhyay

DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Department of Department of
Mathematics
Indian Institute of Technology,
Kharagpur
India - 721302

CERTIFICATE

This is to certify that we have examined the thesis entitled **Study of ID-based Ring Signatures and Implementation of Lattice-based Merkle Tree Accumulator**, submitted by **Pritam Mondal**(Roll Number: *23MA40024*) a postgraduate student of **Department of Department of Mathematics** in partial fulfillment for the award of degree of Master of Science. We hereby accord our approval of it as a study carried out and presented in a manner required for its acceptance in partial fulfillment for the Post Graduate Degree for which it has been submitted. The thesis has fulfilled all the requirements as per the regulations of the Institute and has reached the standard needed for submission.

Supervisor

Department of Department of Mathematics
Indian Institute of Technology,
Kharagpur

Place: Kharagpur
Date:

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor, **Prof. Sourav Mukhopadhyay**, for his unwavering guidance, insightful advice, and constant encouragement throughout this project. His expertise and mentorship have been invaluable in shaping my understanding and fostering my interest in this area of research.

I extend my heartfelt thanks to **Ms. Pratima Jana**, a research scholar in the Cryptography Group of the Department of Mathematics at IIT Kharagpur, for her generous support, invaluable suggestions, and constant assistance, which greatly contributed to the progress of this work.

I am also grateful to the **Department of Mathematics, IIT Kharagpur** for providing the necessary facilities and a conducive environment to carry out this project successfully.

Pritam Mondal
IIT Kharagpur
Date: 02.05.2025

ABSTRACT

This report presents my Master's Thesis Project, where I have explored the generic construction of an ID-based Ring Signature scheme. The primary focus of this work is the development of a lattice-based post-quantum Merkle Tree accumulator in C++. We used a lattice-based hash function to ensure quantum security. The implemented accumulator takes n-dimensional vectors of identities (IDs)—which also serve as the public keys of the signature scheme—as input and accumulates the entire set (ring) of IDs, allowing for efficient storage and verification. In addition to the accumulator, a crucial feature of the system is the user validation process, where a user must prove that their ID belongs to the accumulated ring before signing a message. The signer first generates a witness for their corresponding ID, then generates a proof using the message, accumulated value, ID, and witness to verify the presence of their ID in the ring to the verifier. This proof itself acts as the signature on the message. The successful implementation of these components provides a foundation for further integration into a complete post-quantum ID-based ring signature system. Being resistant to quantum attacks, this system can be used in applications such as blockchain and cryptocurrencies, secure messaging, and privacy-preserving authentication.

Contents

1 Introduction

A **digital signature** scheme is a cryptographic protocol that enables a person or entity to sign digital data, providing a secure and verifiable method to ensure the integrity, authenticity, and non-repudiation of the data. It involves two main processes: **signing** and **verification**. The signer uses their private key to generate a unique signature for the data, while the verifier uses the corresponding public key to validate the signature's authenticity.
Digital signatures are widely used for e-signing legal documents, secure financial transactions, verifying digital communications like emails, cryptocurrency transfers.

Ring signatures are currently considered one of the most valuable cryptographic primitives to ensure privacy. They allow a member of a group (i.e., ring) to anonymously sign a message on behalf of a group in a spontaneous manner. This spontaneity allows signers to form a group of their own choice and to generate an anonymous signature.

ID-based cryptography was introduced in 1984 to erase the need for certificates in public key infrastructures (PKI). ID-based cryptography utilizes a public key which is the identity of the user, for example, an identity can be an email address or a name.

Identity-Based Ring Signature(IDRS) combines the principles of ID-based cryptography and ring signatures to provide anonymous authentication. It allows a user to sign a message on behalf of a group (ring) using only their identity as a public key, without revealing which member actually signed it. The signature ensures anonymity, unforgeability, and identity binding without requiring public key certificates. The trusted Private Key Generator (PKG) generates private keys for members based on their identities. The main advantage of ID-based Ring signature (IDRS) over "traditional" ring signatures in PKI is that IDRS provides better spontaneity. PKI's ring signatures can only form a ring with users that requested certificates for their public keys while in IDRS signers can form a ring using users' identities even if they did not request their secret signing keys to PKG. Additionally, IDRS may significantly reduce the communication overhead in sending the list of ring public keys to the verifier along with the signature for PKI's ring signature. It is useful for

applications like e-voting, or blockchain privacy, where anonymity and authentication are crucial.

Post-quantum IDRS: The advancement of quantum computing necessitates a transition from the ring signature schemes currently in use to post-quantum IDRS schemes. Post-quantum IDRS offers cryptographic primitives that ensure anonymity without relying on certificates. Among all post-quantum cryptographic candidates, lattice-based cryptography has emerged as one of the most extensively studied due to its promising flexibility and strong security guarantees against quantum attacks. We have also used lattice-based hash function in our work.

2 Definitions

2.1 Digital Signature

A digital signature scheme DS is composed by the following algorithms:

- **Key generation** DS.KeyGen(1^λ) $\rightarrow$ (pk, sk) : This takes as input the security parameter λ and outputs the keypair (pk, sk).
- **Signing** DS.Sign(m, sk) $\rightarrow \sigma$: This takes as inputs a message m to be signed and a secret key sk. It outputs a valid digital signature σ.
- **Verification** DS.Verify(m, σ, pk) $\rightarrow$ 0/1 : This takes as inputs the signed message m, a digital signature σ, and the public key pk. It outputs 1 if σ is valid and 0 otherwise.

2.2 Accumulator

An accumulator is is a cryptographic primitive that allows efficient aggregation of a set of elements into a single value and is typically used in settings where we need to efficiently check whether an element belongs to a set without verifying the whole set. Accumulators are highly used in − Blockchain and Cryptography for efficient verification, zero-knowledge proofs, High-Frequency Trading for running profit and loss calculation, Machine Learning model training, etc. We define an accumulator A by the following algorithms:

- **Setup:** $\text{A.Gen}(1^\lambda) \rightarrow \text{pk}_\text{A}$: Takes as input the security parameter λ and outputs the public key pk_A.

- **Evaluation:** $\text{A.Eval}(\text{pk}_\text{A}, L) \rightarrow \text{A}_L, \text{pk}_\text{A}$: Takes as input the public key pk_A and the set L, and outputs the accumulator A_L and an updated public key A.pk for the new accumulated set L.

- **Witness Generation:** $\text{A.WitGen}(\text{pk}_\text{A}, \text{A}_L, L, x_i) \rightarrow w_{x_i}$: Takes as inputs the public key pk_A, the accumulator A_L, the set L, and an element x_i. Outputs the witness w_{x_i} if $x_i \in L$ and $\perp$ otherwise.

- **Verification** $\text{A.Verify}(\text{pk}_\text{A}, \text{A}_L, w_{x_i}, x_i) \rightarrow 0/1$: Takes as inputs the public key pk_A, the accumulator A_L, the witness w_{x_i}, and the element x_i. Returns 1 if w_{x_i} is a valid witness for $x_i \in L$ or 0 otherwise.

Use of Accumulator in IDRS: In IDRS the signer generates a signature using his ID, say $x_i \in L$. Use of accumulator permits signer to include the accumulated value A_L and a witness w_{x_i} in the signature, without revealing all the identities in L. And the verifier don't need to store or verify all the IDs to confirm that $x_i \in L$.

2.3 Cryptographic Hash Function

A **cryptographic hash function** $H : \{0,1\}^* \rightarrow \{0,1\}^m$ takes as input a message a of any length and outputs the hash value b of length m bits. A cryptographic hash function fulfills the following properties:

- **Pre-image Resistance (One-Wayness):** Given a hash value b, where $b = H(a)$ for a uniformly random $a \in \{0,1\}^*$, it is computationally infeasible (in polynomial-time) to find a such that $b = H(a)$.

- **Second Pre-image Resistance:** Knowing a pair $(a_0, H(a_0))$ for a uniformly random $a_0 \in \{0,1\}^*$, it is computationally infeasible to find another input $a_1 \in \{0,1\}^*$ such that $H(a_1) = H(a_0)$.

- **Collision Resistance:** It is computationally infeasible to find two different inputs a_0 and a_1 such that $a_0 \neq a_1$ and $H(a_0) = H(a_1)$.

3 Glimpse of the Technique

The basic idea behind our generic IDRS is that the Public key generator (PKG) possesses a digital signature key pair as master public mpk and private key msk (i.e. $\text{mpk} = \text{pk}, \text{msk} = \text{sk}$). Then, each user with an identity ID requests a signing key SID to PKG which generates a digital signature σ, taking the identity ID as the message or, in other words, PKG computes $\text{DS.Sign}(\text{ID}, \text{msk}) \rightarrow \text{SID}$. To sign a message m, a user in possession of a signing secret key generated by PKG proves (through a NIZK) the knowledge of an input (i.e. witness) w of a circuit C such that $C(w) = 1$. In an IDRS, the signer needs to demonstrate that (1) he owns a secret key generated by the central authority PKG and that (2) his identity belongs to the ring L. We designed a generic circuit C (see Fig.1) which proves the validity of both statements. C is divided into two sub-circuits: C_1 to prove $\text{ID} \in L$ and C_2 to prove $1 = \text{DS.Verify}(\text{ID}, \text{SID}, \text{mpk})$. C_1 and C_2 are associated with an "AND" gate to form the overall circuit C.

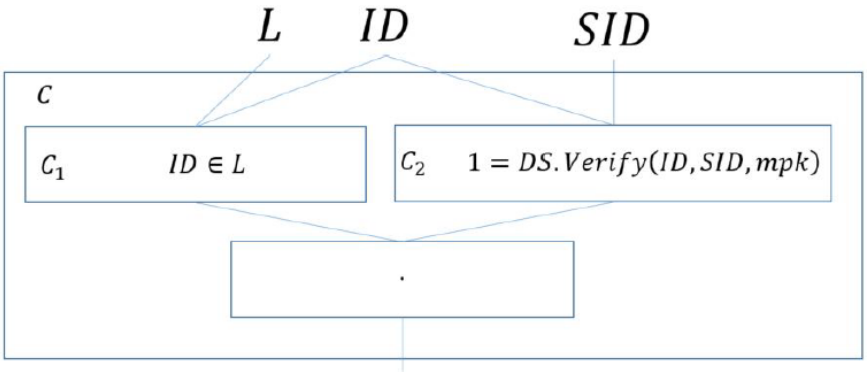


Figure 1: Circuit C

4 Generic Construction for IDRS from Symmetric-key Primitives

- $\text{IDRS.Setup}(1^\lambda) \rightarrow$ **(mpk, msk, param):** This algorithm is executed by PKG and performs the following steps:
  - $\text{DS.KeyGen}(1^\lambda) \rightarrow (\text{mpk}, \text{msk})$

- A.Gen(1^λ) $\rightarrow$ param

- **IDRS.KeyGen(ID, msk) $\rightarrow$ SID :** This algorithm is executed by PKG on the request of the user with the identity ID. It computes a digital signature σ using the ID as the message. The digital signature σ is transmitted to the user ID and becomes their signing secret key SID. This procedure executes the following steps:

  - DS.Sign(ID, msk) $\rightarrow$ SID

- IDRS.Sign(m, L, ID, SID, mpk, param) $\rightarrow$ σ**:** This procedure takes as inputs the message m, the set of N identities L (the ring), the signing secret key SID = σ, and the master public key and public parameters param, which include the initial public key of an empty accumulator. This executes the following steps:

  - A.Eval(param, L) $\rightarrow$ (A_L, pk_A) : This "accumulates" the set of identities belonging to the ring L. It returns an accumulator A and its updated public key pk_A.
  - A.WitGen(pk_A, A_L, L, ID) $\rightarrow$ w_{ID}: This returns the witness w_{ID} for the identity of the signer, which will be used to prove that their ID is included in the accumulator A_L.
  - NIZK.Prove((m, pk_A, A_L, mpk), (SID, ID, w_{ID})) $\rightarrow$ π
    The secret witness is w = (SID, ID, w_{ID}), the public statement is composed of the message m, the accumulator public key pk_A, the accumulator built on the ring A_L, and the PKG public key $mpk.x = (m, \mathrm{pk}_\mathrm{A}, A_L, mpk)$. Now the signer will prove that his Id $\in L$ and he owns a SID generated by PKG (i.e., $(x, w) \in R$) if and only if the following statements hold:
    - (1) A.Verify(pk_A, A_L, w_{ID}, ID)= 1: This is equivalent to proving ID $\in L$, so it will be proven through sub-circuit $C1$.
    - (2) DS.Verify(ID, SID, mpk) = 1: This will be proven through sub-circuit $C2$.
  - $\pi \rightarrow \sigma$

- **IDRS.Verify(**m**,** L**,** σ**,** mpk**, param) $\rightarrow$ 0/1 :** This algorithm takes as inputs the message m, the list of identities L, and a ring signature σ. It verifies the validity of σ by executing the following steps:

- $\sigma \rightarrow \pi$
- A.Eval(param, L) $\rightarrow$ (A_L, pk_A) : The verifier constructs the accumulator for the set of identities belonging to the ring L. It returns an accumulator A_L and its updated public key pk_A.
- Return NIZK.Verify((m, pk_A, A_L, mpk), π): This returns 1 if the proof π is valid for circuit C.

How does signer prove of having a valid SID to verifier ?
Signer construct the signature π using his ID, SID. When verifier uses the algorithm NIZK.Verify((m, pk_A, A_L, mpk), π) and gets 1, he understands that signer has valid SID (that's why construction of π was possible and verify algorithm returns 1).

5 Lattice-based Merkle Tree Accumulator

Aims to prove a valid signer's ID $\in L$. This needs to be a one-way (so that the adversary can't do inverse computation to find the signer's ID) and so must have two properties (a) Efficient computation and (b) Hard to invert.

Merkle Accumulator respects the definition of an accumulator and holds the above properties. To construct sub-circuit $C1$ we use Merkle Accumulator from the paper [Zero-Knowledge Arguments for Lattice-Based Accumulators: Logarithmic-Size Ring Signatures and Group Signatures without Trapdoors] [Ref 2]

We work with positive integers n, q, k, and m, where n is the security parameter, $k = \lceil \log q \rceil$, $m = 2nk$. We identify the set $\mathbb{Z}_q$ by the set $\{\ 0, 1, ..., (q-1)\ \}$.

5.1 A Family of Lattice-based Collision-Resistant Hash Functions

We describe the specific family of lattice-based collision-resistant hash functions, upon which our Merkle hash tree will be built.

Definition: The function family $\mathcal{H}$ mapping $\{0,1\}^{nk} \times \{0,1\}^{nk}$ to $\{0,1\}^{nk}$ is defined as

$$\mathcal{H} = \{h_A \mid A \in \mathbb{Z}_q^{n \times m}\},$$

where for $A = [A_0 \mid A_1]$ with $A_0, A_1 \in \mathbb{Z}_q^{n \times nk}$, and for any $(u_0, u_1) \in \{0,1\}^{nk} \times \{0,1\}^{nk}$, we have:

$$h_A(u_0, u_1) = \text{bin}\,(A_0 \cdot u_0 + A_1 \cdot u_1 \mod q) \in \{0,1\}^{nk}.$$

Note that $h_A(u_0, u_1) = u \Leftrightarrow A_0 \cdot u_0 + A_1 \cdot u_1 = G \cdot u \mod q$.

Before explaining the security of the mentioned hash function we define,

Small Integer Solution (SIS) Problem: Given a uniformly random matrix $A \in \mathbb{Z}_q^{n \times m}$, where q is a modulus, the goal is to find a non-zero vector $x \in \mathbb{Z}^m$ such that:

$$A \cdot x \equiv 0 \pmod q \quad \text{and} \quad \|x\| \leq \beta,$$

where $\|x\|$ is the norm of x, and β is a small bound.
This problem becomes hard for appropriate choices of parameters e.g., sufficiently large q, n, m and small β.

How this hash function is Collision resistant and Pre-image resistant

- **Collision Resistance:** A collision in h_A would require finding distinct pairs (u_0, u_1) and (v_0, v_1) such that $h_A(u_0, u_1) = h_A(v_0, v_1)$.
  This implies, $A \cdot z = 0 \pmod q$ where $z = [(u_0 - v_0), (u_1 - v_1)]$
  So to find a collision we need to solve SIS problem of A which is a hard problem.
  So this hash function is collision resistant.

- **Pre-image Resistant:** Given a hash value $h_A(u_0, u_1)$ to compute the pre-image (u_0, u_1) the adversary need to invert the operation $(A_0 \cdot u_0 + A_1 \cdot u_1 \mod q)$.
  This is also difficult since search space is $\{0,1\}^{nk} \times \{0,1\}^{nk}$. So this hash function is also pre-image resistant.

5.2 Construction of Merkle-Tree Accumulator

We now construct a merkle tree accumulator with $N = 2^l$ leaves, where l is a positive integer. In our IDRS these N leaves would be the IDs from the ring L.

- **TSetup**(n): Sample $A \in \mathbb{Z}_q^{n \times m}$ randomly.

- $\mathbf{TAcc}_A(L = \{d_0 \in \{0,1\}^{nk}, \ldots, d_{N-1} \in \{0,1\}^{nk}\})$: For every $j \in [0, N-1]$, let $(j_1, \ldots, j_\ell) \in \{0,1\}^\ell$ be the binary representation of j, and let $d_j = u_{j_1,\ldots,j_\ell}$.

  Form the tree of depth $\ell = \log N$ based on the N leaves $u_{0,0,\ldots,0}, \ldots, u_{1,1,\ldots,1}$ as follows:

  1. At depth $i \in [\ell]$, the node $u_{b_1,\ldots,b_i} \in \{0,1\}^{nk}$, for all $(b_1, \ldots, b_i) \in \{0,1\}^i$, is defined as
  $$h_A(u_{b_1,\ldots,b_i,0}, u_{b_1,\ldots,b_i,1}).$$
  2. At depth 0: The root $u \in \{0,1\}^{nk}$ is defined as
  $$h_A(u_0, u_1).$$

  The algorithm outputs the accumulator value u.

- $\mathbf{TWitness}_A(L, d)$: If $d \notin R$, return $\perp$. Otherwise, $d = d_j$ for some $j \in [0, N-1]$ with binary representation $(j_1, \ldots, j_\ell)$. Output the witness w defined as:
$$w = \left((j_1, \ldots, j_\ell), (u_{j_1,\ldots,\bar{j}_l,\neg j_\ell}, \ldots, u_{j_1,\bar{j}_2}, u_{\bar{j}_1})\right) \in \{0,1\}^\ell \times \{0,1\}^{nk\times\ell},$$
  for $u_{j_1,\ldots,j_{\ell-1},\bar{j}_l}, \ldots, u_{j_1,\bar{j}_2}, u_{\bar{j}_1}$ computed by algorithm $\mathbf{TAcc}_A(L)$.

- $\mathbf{TVerify}_A(u, d, w)$: Let the given witness w be of the form:
$$w = ((j_1, \ldots, j_\ell), (w_\ell, \ldots, w_1)) \in \{0,1\}^\ell \times \{0,1\}^{nk\times\ell}.$$
  The algorithm recursively computes the path $v_\ell, v_{\ell-1}, \ldots, v_1, v_0 \in \{0,1\}^{nk}$ as follows: $v_\ell = d$ and
$$\forall i \in \{\ell-1, \ldots, 1, 0\} : v_i = \begin{cases} h_A(v_{i+1}, w_{i+1}), & \text{if } j_{i+1} = 0, \\ h_A(w_{i+1}, v_{i+1}), & \text{if } j_{i+1} = 1. \end{cases}$$
  Then it returns 1 if $v_0 = u$. Otherwise, it returns 0.

In Figure 2, we give an illustrative example of a tree with $2^3 = 8$ leaves which accumulates the data blocks $d_0, \ldots, d_7$ into the value u at the root. The bit string (101) and the gray nodes form a witness to the fact that d_5 is accumulated in u.

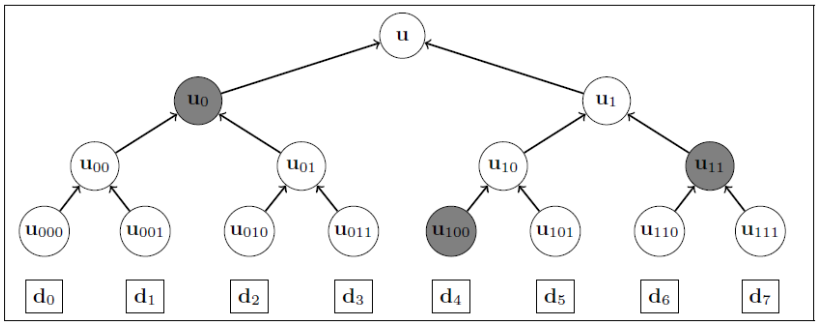


Figure 2: Merkle-tree Accumulator

5.3 Application of Lattice-based Merkle Accumulator

The lattice-based Merkle Tree accumulator combines two powerful concepts: the Merkle Tree, used for efficient and secure data verification, and lattice-based cryptography, which provides resistance to quantum attacks. This makes such an accumulator highly valuable in practical post-quantum applications. Here are several key practical use cases:

- **Post-Quantum Ring Signature Schemes:** Proving membership in a group of public keys to sign a message anonymously and quantum-securely.
- **Post-Quantum Blockchain Systems:** Efficient storage and verification of large sets of identities or transactions.
- **Privacy-Preserving Authentication:** Allows users to prove membership in a group (e.g., ID-based ring signatures) without revealing their identity.
- **Cryptocurrency Wallets and Multi-Signature Schemes:** Accumulating a set of public keys for multi-signature wallets.
- **Identity Management and Access Control:** Verifying whether a user's ID is part of an authorized set without revealing the set or requiring heavy computation.

5.4 Code Repository

The source code for the implementation of the lattice-based Merkle tree accumulator developed as part of this thesis is publicly available on GitHub:

`https://github.com/pritam01729/merkle_accumulator`

This repository contains a C++ implementation of the accumulator. It demonstrates how lattice-based cryptographic techniques can be used to construct efficient and secure membership proofs.

5.5 Inputs and Outputs

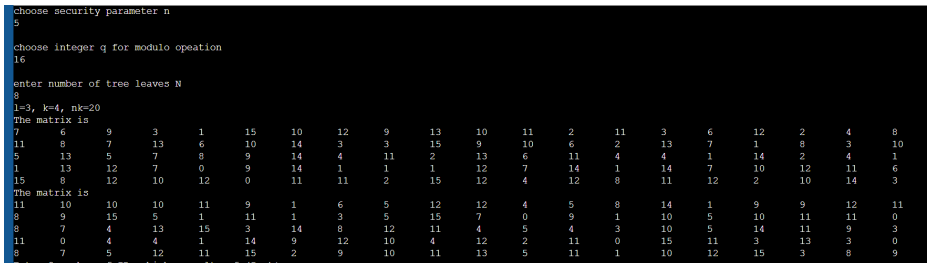
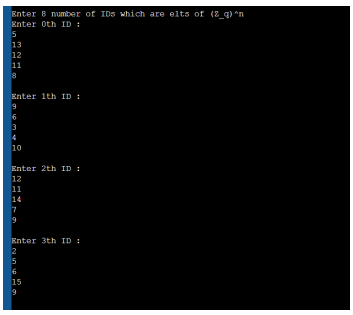


Figure 3: Input 'n','q','N' and generate the A0, A1 matrices for hash function



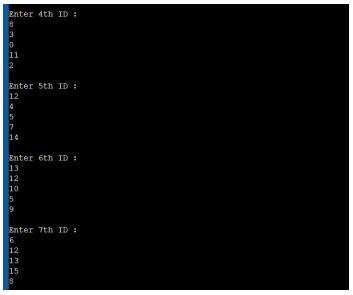


Figure 4: Input IDs

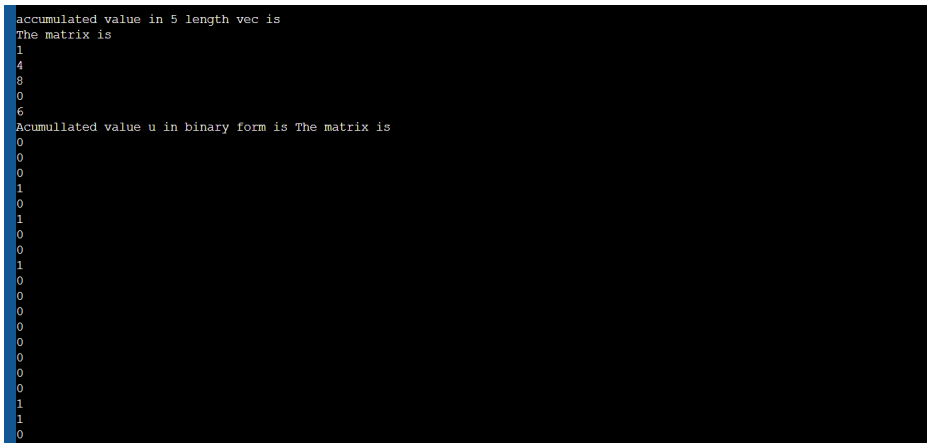


Figure 5: Accumulated value

Figure 6: Witness generation for the signing ID

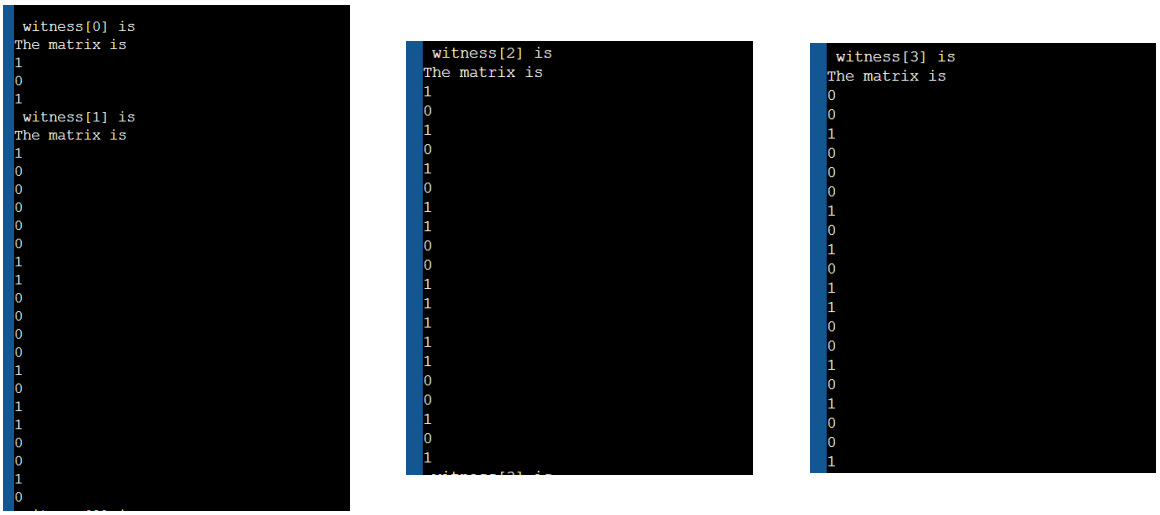


Figure 7: Witness for that ID

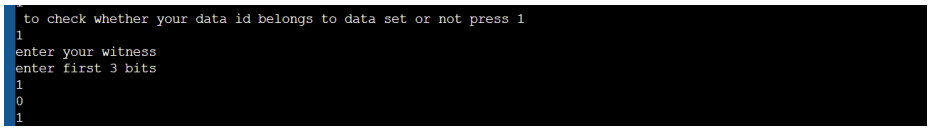


Figure 8: Give an input witness to verify

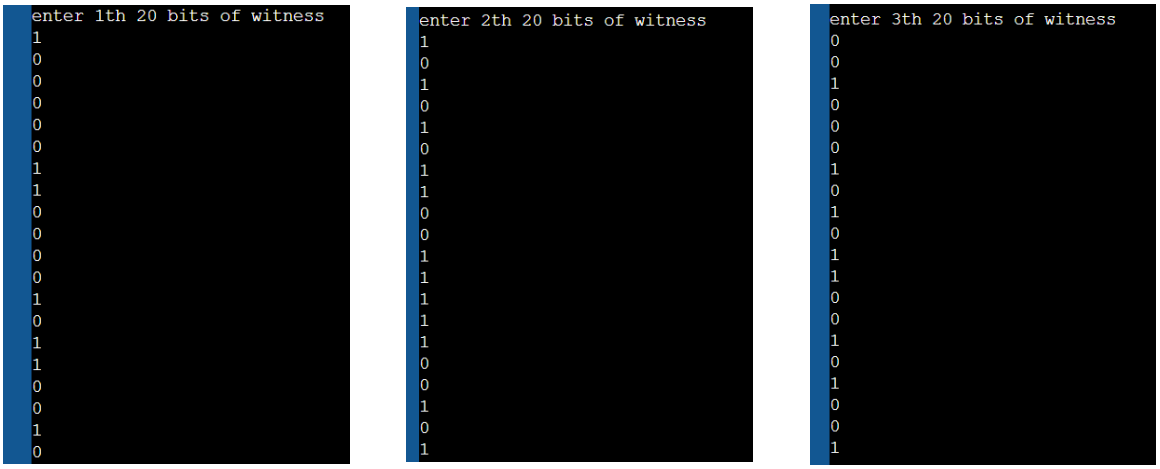


Figure 9: Input witness

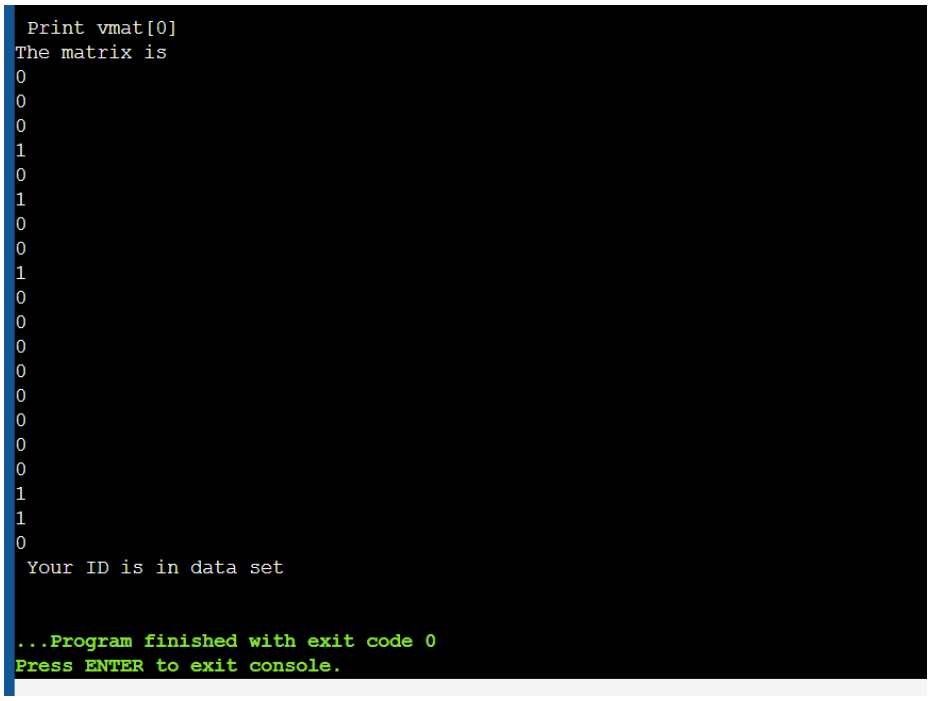


Figure 10: Output whether there is a valid ID for the witness or not

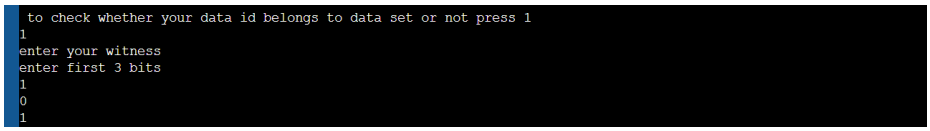


Figure 11: Taking another witness to verify

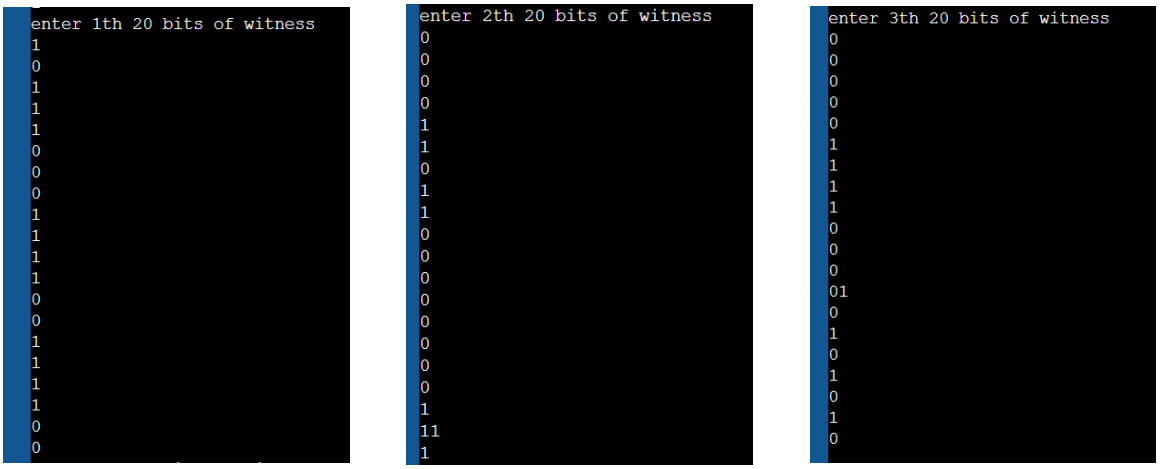


Figure 12: Input witness

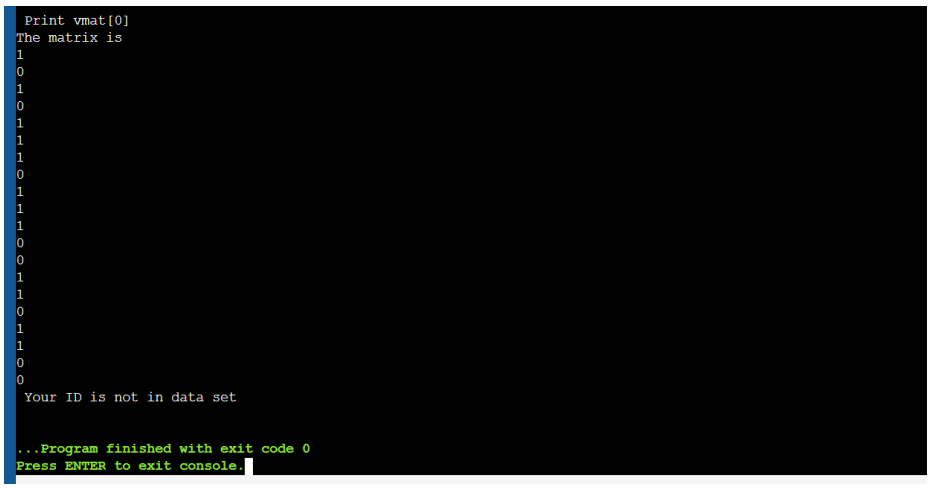


Figure 13: Output whether there is a valid ID for the witness or not

(1),(2)(3),(4),(5),(6),(7),

References

[1] M. Buser, J. K. Liu, R. Steinfeld, and A. Sakzad, "Post-quantum id-based ring signatures from symmetric-key primitives," in *International Conference on Applied Cryptography and Network Security*, pp. 892–912, Springer, 2022.

[2] B. Libert, S. Ling, K. Nguyen, and H. Wang, "Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors," *Journal of Cryptology*, vol. 36, 2023.

[3] M. H. Au, J. K. Liu, T. H. Yuen, and D. S. Wong, "Id-based ring signature scheme secure in the standard model," in *Advances in Information and Computer Security: First International Workshop on Security, IWSEC 2006, Kyoto, Japan, October 23-24, 2006. Proceedings 1*, pp. 1–16, Springer, 2006.

[4] S. Chow, R. Lui, L. Hui, and S. Yiu, "Identity based ring signature: Why, how and what next.," pp. 144–161, 01 2005.

[5] C. Gamage, B. Gras, B. Crispo, and A. Tanenbaum, "An identity-based ring signature scheme with enhanced privacy," pp. 1–5, 08 2006.

[6] J. Herranz, "Identity-based ring signatures from rsa," *Theoretical Computer Science*, vol. 389, pp. 100–117, 12 2007.

[7] D. Boneh, S. Eskandarian, and B. Fisch, *Post-quantum EPID Signatures from Symmetric Primitives: The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4–8, 2019, Proceedings*, pp. 251–271. 02 2019.