BANK OF CANADA
BANQUE DU CANADA

Staff Working Paper/Document de travail du personnel—2023-50

Last updated: September 28, 2023

Anonymous Credentials: Secret-Free and Quantum-Safe

by Raza Ali Kazmi[1] and Cyrus Minwalla[2]

[1] Corporate Services Department
Bank of Canada
rkazmi@bankofcanada.ca

[2] Information Technology Services Department
Bank of Canada
cminwalla@bankofcanada.ca

DOI: https://doi.org/10.34989/swp-2023-50 | **ISSN** 1701-9397

Abstract

An anonymous credential mechanism is a set of protocols that allows users to obtain credentials from an organization and demonstrate ownership of these credentials without compromising users' privacy. In this work, we construct the first secret-free and quantum-safe credential mechanism. The scheme is secret-free in the sense that an organization does not need to guard a secret key. The scheme is also lightweight in construction. Security of the scheme relies on the ability of the organization to maintain the integrity of a publicly known data structure—namely, a Merkle tree—that utilizes a quantum-safe, partially homomorphic hash function as a foundational primitive. We also construct a simple, quantum-safe, zero-knowledge argument of knowledge of membership in the Merkle tree. Additionally, we explore a concrete instantiation of the scheme and show it to be practically efficient for the core functions of enrollment and verification.

Topics: Central bank research; Digital currencies and fintech; Payment clearing and settlement systems
JEL codes: E, E4, E42, G, G2, G21, O, O3, O31

Résumé

Un mécanisme d'identifiants anonymes est un ensemble de protocoles qui permet aux utilisateurs d'obtenir des identifiants d'une organisation et de prouver que ceux-ci leur appartiennent sans compromettre leurs renseignements personnels. Nous construisons le premier mécanisme d'identifiants résistant aux attaques quantiques n'utilisant pas de clé secrète. La sécurité de ce mécanisme efficient repose sur la capacité de l'organisation à maintenir l'intégrité d'une structure de données connue publiquement. Le mécanisme est basé sur un arbre de Merkle et utilise comme primitive fondamentale une fonction de hachage résistante aux attaques quantiques qui est partiellement homomorphe. Nous construisons également un argument de connaissance à divulgation nulle de connaissance simple et à l'épreuve des attaques quantiques pour vérifier la présence des identifiants dans l'arbre de Merkle. De plus, nous explorons une instanciation concrète du mécanisme et montrons son efficacité pour les fonctions de base de l'inscription et de la vérification.

Sujets : Recherches menées par les banques centrales; Monnaies numériques et technologies financières; Systèmes de compensation et de règlement des paiements

Codes JEL : E, E4, E42, G, G2, G21, O, O3, O31

1 Introduction

Digital identities enable users to authenticate themselves in online marketplace purchases or when selling goods and services. Physical forms of identity embed special protections against counterfeiting and duplication. Given the nature of digital information, copying valid credentials is trivially easy, despite protections against tampering and unauthorized creation. Users desire digital identities that they can acquire and use with the confidence that the credentials cannot be duplicated or misused by merchants and vendors or by third parties collecting information on behalf of merchants. In addition to safeguarding user information, digital identities can be configured to reveal the minimum information required to obtain services while keeping other sensitive information secret. Moreover, given the advent of quantum computing, credential mechanisms need to be hardened against quantum computing attacks that threaten to break classical cryptographic primitives based on factorization or discrete logarithms.

2 Related Work

Anonymous credentials share similarities with fully dynamic group signatures [1], ring signatures [2], and Enhanced Privacy ID (EPID) signature schemes [3, 4]. In the following sections, we compare these primitives with anonymous credential mechanisms.

2.1 Comparison with Group Signatures

Like an anonymous credential, a fully dynamic group signature scheme allows a user to prove their membership in a group anonymously [1]. However, the trace manager can always open a valid signature and identify the actual signer. This behaviour is distinct from anonymous credential systems, where even the issuing organization cannot de-anonymize a user from a verification proof. In applications where full anonymity is desired, anonymous credentials are preferable to dynamic group signatures.

2.2 Comparison with Ring Signatures

A ring signature satisfies many of the properties of an anonymous credential [2]. The primary difference is that the ring signature scheme is open to participation, such that anyone can enroll without requiring consent. This is at odds with the aims of a credential mechanism, where membership is controlled by an issuing organization. In the domain of government-issued credentials (e.g., certifying a valid driver's license or legal drinking age), ring signatures are not an appropriate choice.

2.3 Comparison with EPID Signatures

Boneh, Eskandarian, and Fisch proposed the first quantum-safe EPID signatures [4], which are similar to the group signature scheme but with the following key difference: no entity, including the organization (issuing authority), can open the signature and de-anonymize a user [3, 4]. In this mechanism, the **EPID** signatures are static in the sense that group membership is decided at the time of enrolment and new members cannot be added or removed [4] after the fact. This ultimately limits the usability of the scheme as a credential mechanism. Therefore, EPID signatures are not suitable for constructing anonymous credential systems. To the best of our knowledge, a dynamic EPID signature scheme does not currently exist.

2.4 Quantum-Safe Anonymous Credential System

Recently, Bootle et al. constructed an anonymous credential mechanism based on the presumed quantum hardness of a relatively new problem called ISIS_f (Inhomogenous Shortest Integer Solution problem) [5]. The ISIS_f problem informally states that given an integer q, a matrix $A \in \mathcal{Z}_p^{n \times m}$ (picked according to some distribution), a function $f : [N] \to \mathcal{Z}_p^n$, a $\beta \in \mathbb{R}$ and a $x \in [N]$, find an integer vector $||\vec{s}|| \leq \beta$ satisfying $A\vec{s} = f(x)$. For efficiency and other technical reasons, f is chosen from a class of simple functions such as linear or slightly higher-degree functions (as opposed to cryptographic hash functions), which admits efficient

zero-knowledge proofs [5]. However, the hardness of the ISIS_f problem when initiated with such functions is not well understood and is still an open problem. Note in [5], the issuer uses blind signatures to create credentials, whereas in our construction the issuer does not employ any type of signature scheme to create credentials and, in this sense, is secret-free. To the best of our knowledge, Bootle et al.'s scheme is the only other anonymous credential scheme that is constructed from a quantum-safe assumption.

All the schemes mentioned above possess a common security assumption: namely, that the organization must safeguard its secret key at all times. Compromising the secret key could result in an attacker issuing legitimate credentials. Moreover, it may be difficult for the organization to detect forged credentials, and the system may be rapidly flooded by such credentials before mitigation steps are taken.

2.5 Comparison with Secret-Free Electronic Cash

Sander and Ta-Shma first proposed the idea of secret-free systems in the context of electronic cash [6]. They demonstrated that an issuing bank does not require a digital signature to issue electronic cash. Their idea is best explained by considering the set membership problem, where an issuer holds a list of coins $L = \{v_1, \ldots, v_n\}$. The problem is providing a zero-knowledge argument of knowledge, **ZKAoK**, of membership in L. This is achieved by creating a Merkle tree for the data values in L, where the root of the Merkle tree is made public [7]. A prover provides a **ZKAoK** that it knows some value $v_i \in L$ and hash chain (path) from the corresponding leaf to the tree's root. The Merkle tree-based approach has an advantage over a signature-based solution, which is that the security does not depend on guarding a secret key but instead depends on the organization's ability to maintain the integrity of the Merkle tree.

Our main approach has been inspired by ideas presented in Sander and Ta-Shma [6], on which we significantly improve. First, the proof systems they present for the membership in the Merkle tree [6] are based on discrete logarithmic problems and are not quantum-safe, whereas our construction relies on a quantum-safe homomorphic hash function. Additionally, we exploit the homomorphic property of the underlying hash function to construct a much simpler and more efficient zero-knowledge argument of knowledge to prove membership in the Merkle tree. Here, the core operations are exclusive or and evaluation of a homomorphic hash function. Furthermore, Sander and Ta-Shma's [6] focus was to construct anonymous digital cash, which has different requirements from anonymous credentials. For example, to prevent double-spending in digital cash, users are de-anonymized if they try to use the same coin twice. By contrast, in anonymous credentials, users should be able to authenticate themselves arbitrarily many times without compromising their privacy. Therefore, the underlying protocols are technically different.

2.6 Our Contributions

In this work, we construct a secret-free quantum-safe anonymous credential system from any additively homomorphic quantum-safe hash function $\mathcal{H}$. In our generic construction in Section 4 we assumed without loss of generality (WLOG) that $\mathcal{H}$ is homomorphic with respect to exclusive or, $\mathcal{H}(x \oplus y) = \mathcal{H}(x) \oplus \mathcal{H}(y)$.[1] In Section 7, we estimated the cost of our system when initiated with SWIFTT [8]. Unlike previous constructions, the security of our scheme does not rely on digital signatures and is secret-free by design; that is, a secret key is not required to issue credentials. Rather, security relies on the organization's ability to maintain the integrity of a data structure—namely, the Merkle tree—and the quantum hardness of the underlying additively homomorphic hash function. One of our contributions is to provide a quantum-safe, zero-knowledge argument of knowledge for membership in the Merkle tree that is simpler compared with previously known quantum-safe proofs [9, 4]. To the best of our knowledge, this is the first anonymous credential scheme that is secret-free and quantum-safe.

3 Preliminaries and Notation

An anonymous credential system [10–16] consists of three entities: organizations, users, and verifiers. An organization **O** is responsible for creating and issuing credentials, which users can utilize to authenticate

[1] Our results are applicable for any additively homomorphic hash function.

themselves anonymously at a verifier. Credentials can be non-expiring in that an arbitrary number of verification attempts are possible. Anonymity can be guaranteed if authentications of the same user cannot be linked [15].

1. `unforgeability:` It is computationally infeasible to forge a credential or authenticate on behalf of a non-participating user even if the organization, verifiers and other users in the system collude.
2. `unlinkability:` It is computationally infeasible to link verification sessions of the same user.
3. `revocability:` The issuing organization can revoke credentials.

Let $\mathbb{N}$ be the set of natural numbers. For $n \in \mathbb{N}$, we denote the set $[n] = \{1, \cdots, n\}$. We denote the set of all binary strings of length n by $\{0,1\}^n$. An element $s \in \{0,1\}^n$ is called a bitstring, and $|s| = n$ denotes its length. Given two bitstrings, x and y, of equal length, we denote their bitwise XOR by $x \oplus y$, and **MB** denotes megabytes. We denote a positive polynomial by $poly(n)$ and a cryptographic hash function by $\mathcal{H}$. For a finite set X, the notation $x \xleftarrow{\$} X$ indicates that x is selected uniformly at random from X. For any binary tree $\mathcal{T}$, the notation (a_i, d_i) means a_i is a node in $\mathcal{T}$ and is a `left child` if $d_i = 1$ and a `right child` if $d_i = 0$. Finally, **ZKPoK** and **ZKAoK** denote zero-knowledge proof of knowledge and zero-knowledge argument of knowledge.

Lemma 1 presented and proven below is an important operation used in the **ZKAoK** construction.

Lemma 1. *For binary strings $a, b, c, d \in \{0,1\}^n$*

$$(a||b) \oplus (c||d) = (a \oplus c)||(b \oplus d) \tag{1}$$

Proof. Let $a = a_n \dots a_1$, $b = b_n \dots b_1$, $c = c_n c_{n-1} \dots c_1$ and $d = d_n \dots d_1$.

$$\begin{aligned}(a \oplus c)||(b \oplus d) &= (a_n \oplus c_n \dots a_1 \oplus c_1)||(b_n \oplus d_n \dots b_1 \oplus d_1)\\ &= (a_n \dots a_1||b_n \dots b_1) \oplus (c_n \dots c_1||d_n \dots d_1)\\ &= (a||b) \oplus (c||d).\end{aligned}$$

3.1 Additively Homomorphic Hash Function

Certain families of cryptographic hash functions are partially homomorphic. For instance, Chaum, van Heijst and Pfitzmann [17] constructed a simple hash function that is secure under the presumed hardness of the discrete logarithm problem and additively homomorphic. However, this hash function is slow for practical applications and is not quantum-safe. Below we describe a family of partially homomorphic hash functions that are efficient and provably quantum-safe; i.e., finding a collision is at least as difficult as finding the shortest vector in ideal lattices.

One practical example is **SWIFFT** [8], which is a family of hash functions that are practically efficient, highly parallelizable, provably secure against quantum-safe adversaries, and additively homomorphic. The throughput of **SWIFFT** is comparable to and even exceeds the throughput of SHA2 on modern computers, despite the fact that the parallelization of **SWIFFT** is not fully exploited to date. The input is a binary string of length mn, and the output is a binary string of length $n \log_2(p)$ (for appropriate parameters n, m, p). For parameters $m = 2^4, n = 2^6$ and $p = 257$, it maps 1024 bits to roughly 512 bits [8]. We use these parameters in the cost analysis section (Section 7.4).

3.2 Zero-Knowledge Interactive Proofs

Definition 1 *An interactive proof system with soundness error $s : \mathbb{N} \longrightarrow [0,1]$ and completeness $c : \mathbb{N} \longrightarrow [0,1]$, for a language $L \subseteq \{0,1\}^*$ is a pair of algorithms: a prover P (possibly computationally unbounded) and a probabilistic polynomial-time verifier V, with the following properties.*

- **Completeness**: *For all inputs x in L, the verifier after interacting with the prover $([P(x) \leftrightarrow V(x)])$ accepts the proof (i.e. $out_V[P(x) \leftrightarrow V(x)] = 1$) with probability at least $c|x|$*

$$\forall x \in L, \quad \mathbf{Pr}\left(out_V[P(x) \leftrightarrow V(x)] = 1\right) \geq 1 - c|x|.$$

- **Soundness**: *For every computationally unbounded P^*,*

$$\forall x \notin L, \quad \mathbf{Pr}\left(out_V[P^*(x) \leftrightarrow V(x)] = 1\right) \leq s|x|.$$

3.3 Perfect and Statistical Zero-Knowledge Proofs

Definition 2 *Suppose that we have a polynomial-time interactive proof or argument system* $[P \leftrightarrow V]$ *for a language* $L \subseteq \{0,1\}^*$. *Let* V^* *denote a (possibly cheating) verifier. Let* $\mathcal{T}(V^*, R_{V^*}, x)$ *be the set of all possible transcripts that could be produced as the result of an interactive proof* $[P \leftrightarrow V^*]$ *on input* $x \in L$ *and* R_{V^*} *be its random coins. Suppose that for every such* V^* *there exists an expected polynomial-time simulator* S, *and let* $\mathcal{T}(S, R_S, x)$ *denote the set of all possible simulated transcripts that could be produced by* S. *Let* $\mathbf{Pr}_{V^*}(\mathcal{T})$ *denote the probability distribution on* $\mathcal{T}(V^*, R_{V^*}, x)$ *as a result of* $[P \leftrightarrow V^*]$, *and* $\mathbf{Pr}_S(\mathcal{T})$ *be the probability distribution on* $\mathcal{T}(S, R_S, x)$ *induced by* S. *If the distributions* $\mathbf{Pr}_S(\mathcal{T}) = \mathbf{Pr}_{V^*}(\mathcal{T})$, *then we define the proof or argument system to be perfect zero-knowledge* (**PZKIP**), *and if the distributions* $\mathbf{Pr_S}(\mathcal{T})$ *and* $\mathbf{Pr}_{V^*}(\mathcal{T})$ *are statistically close (in the size of the input* x*), then we define the interactive proof to be statistical zero-knowledge* (**SZKIP**).

3.4 Proof of Knowledge

Let $s, c : \mathbb{N} \longrightarrow \in [0,1]$ be functions and R be a binary relationship. The language L_R associated with R is the set of all x for which there exists a witness w such that $R(x,w) = 1$,

$$L_R = \{x : \exists w \text{ such that } R(x,w) = 1\}.$$

Definition 3 *A pair of (Probabilistic Polynomial-time) Turing machines,* P *and* V, *constitute an interactive proof of knowledge for a relation* R *if the following two conditions hold:*

- **Completeness/Nontriviality** *with error* c: *For all* $(x,w) \in R$

$$\mathbf{Pr}\left(out_V[P(x,w) \leftrightarrow V(x)] = 1\right) \geq 1 - c(|x|).$$

- **Soundness/Validity** *with error* s: *For all prover* P^*, *there exists a probabilistic oracle machine* E_P^* *such that for all* x *and for all* w

$$\mathbf{Pr}((x, E_P^*(x)) = 1) \geq \mathbf{Pr}\left(out_V[P(x,w) \leftrightarrow V(x)] = 1\right) - k(|x|).$$

3.5 Zero-Knowledge Interactive Arguments

An interactive argument (or computationally sound proof system) is a relaxation of an interactive proof in which the prover is restricted to be a polynomial-time algorithm in the size of the input.

3.6 Sketch of the Zero-Knowledge Argument of Knowledge

In this section, we explain by example how one can exploit the homomorphic property of a hash function to achieve a simple **ZKAoK** that works for the Merkle tree membership. Let $\mathcal{H}$ be a quantum-safe cryptographic hash function that is an additive homomorphic with respect to exclusive or. Let $C_1, \cdots, C_n$ denote the credentials, and the leaves of the Merkle tree are $\mathcal{H}(C_1), \ldots, \mathcal{H}(C_n)$. Every internal node is a hash of the concatenation of its children. Figure 1 is an example of a Merkle tree of height $h = 2$.

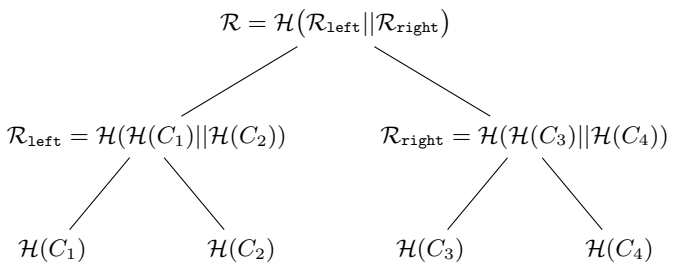

Fig. 1. Example Merkle Tree of Height 2

Consider the Merkle tree defined in Figure 1. Let $a_1 := \mathcal{H}(C_2)$ and $a_2 := \mathcal{H}(\mathcal{H}(C_3)||\mathcal{H}(C_4))$. Suppose a user possesses credentials C_1 and hash chain $((a_1, d_1), (a_2, d_2)$, where $d_1 =$ `right child` and $d_2 =$ `right child`. To prove membership, the user selects random strings $\alpha_0, \alpha_j, \beta_j,$ and γ_j for $1 \leq j \leq 2$ and sends the verifier

$$\big(C_1 \oplus \alpha_0, a_1 \oplus \mathcal{H}(\alpha_1), a_2 \oplus \mathcal{H}(\alpha_2)\big),$$

two correction words

$$\big(\beta_1, \mathcal{H}(\gamma_1)\big) \text{ and } \big(\mathcal{H}(\beta_2), \mathcal{H}(\gamma_2)\big)$$

corresponding to d_1 and d_2, and an error term $\mathcal{E}_2$ (Algorithm 2). The verifier computes the following:

1. Set $x'_1 := \mathcal{H}(C_1 \oplus \alpha_0)$.
2. Set $w := \mathcal{H}(\beta_1) \oplus \mathcal{H}(\gamma_1)$ and $x'_2 := \mathcal{H}\big((x'_1||(a_1 \oplus \mathcal{H}(\alpha_1)) \oplus (w||w)\big)$.
3. Set $w := \mathcal{H}(\beta_2) \oplus \mathcal{H}(\gamma_2)$ and $x'_3 := \mathcal{H}\big((x'_2||(a_2 \oplus \mathcal{H}(\alpha_2))) \oplus (w||w)\big)$.
4. Accepts if and only if $\mathcal{R} = x'_3 \oplus \mathcal{E}_2$.

To ensure that the prover cannot cheat by exploiting the homomorphic property, the prover is required to provide a ZKAoK of strings α_0 and $\alpha_j, \beta_j, \gamma_j$ for $1 \leq j \leq 2$, which determines the error term $\mathcal{E}_2$.

4 Anonymous Credentials

In this section, we provide a generic construction of an anonymous credential scheme from any quantum-safe additively homomorphic hash function $\mathcal{H}$. We can assume without loss of generality that $\mathcal{H}$ is homomorphic with respective to exclusive or,

$$\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^l, \quad \mathcal{H}(x \oplus y) = \mathcal{H}(x) \oplus \mathcal{H}(y),\ l \text{ is a fixed positive integer.}$$

Parameters: Let $|N|$ denote the size of each credential in the system. Let h and $\mathcal{R}$ denote the height and public root of a Merkle tree, respectively. We assume that an organization **O** can issue at most $2^{\frac{|N|}{2}}$ credentials.

4.1 Setup

The scheme consists of three protocols, `Join`, `Verification`, and `Revocation`. `Join` (enrollment) is triggered when a user requests a credential from an organization **O** and undergoes **O**'s Know Your Customer (KYC) process. For credentials equivalent to government-issued identification, the process may entail the disclosure and verification of the user's personally identifiable information (**PII**).

4.2 Join (Enrollment)

The `Join`, or enrollment, protocol (Algorithm 1) is a two-round interactive protocol between a user U and an organization **O**. At the end of the protocol, U receives: $\big(C, ((a_1, d_1), \ldots, (a_h, d_h))\big)$ and **O** receives the commitment $\mathcal{H}(C)$. The credential C is a (secret) random string chosen by the user, and $((a_1, d_1), \ldots, (a_h, d_h))$ is a unique path to the public root $\mathcal{R}$ (Section 3.6). We call $((a_1, d_1), \ldots, (a_h, d_h))$ a hash chain corresponding to the credential C.

Note that a user U does not need to prove knowledge of C to the organization. This lack of proof does not expose the system to an attack. Consider scenarios where a malicious user does not send $H(C)$ but sends up some arbitrary string y in the codomain of H. Suppose U does not know a C such $y = \mathcal{H}(C)$ or y is not in the range of H. In all cases, the user will either fail the joining process (if $H(C)$ already exists) or will not be able to authenticate at the time of verification (Algorithm 2). This is because the verification protocol is a perfect zero-knowledge argument of knowledge, and passing it without well-formed C will imply that a collision has been found. Furthermore, we note that Line 2.(a) of Algorithm 1 implies a database search, for which efficient data structures (e.g., heap) exist with a worst-case time complexity of $O(\log(N))$ and an average-case complexity of $O(1)$ (e.g., hash tables).

1. User U
   (a) Picks a uniformly random string $C \xleftarrow{\$} \{0,1\}^{|N|}$.
   (b) Computes $\mathcal{H}(C)$ and sends to **O**.
2. Organization **O**
   (a) If $\mathcal{H}(C)$ is in its private database $\mathcal{D}$, then aborts.[a]
   (b) Adds $\mathcal{H}(C)$ to an unused leaf in the tree $\mathcal{T}$.
   (c) Updates the tree $\mathcal{T}$ and adds the new root to the public root list (Section 4.5).
   (d) Computes the hash chain $((a_1, d_1), \ldots, (a_h, d_h))$.
   (e) Stores $\big(\mathcal{H}(C), ((a_1, d_1), \ldots, (a_h, d_h)), \mathbf{PII}\big)$ in its private database $\mathcal{D}$.
   (f) Sends $((a_1, d_1), \ldots, (a_h, d_h))$ to U.
3. **User**
   (a) Verifies if $((a_1, d_1), \ldots, (a_h, d_h))$ is a hash chain corresponding to C.

[a] Probability of happening this is $2^{\frac{-|H(C)|}{2}}$ [18].

Algorithm 1: `Join` Protocol

4.3 Verification

The `Verification` protocol, described in Algorithm 2, is a three-round zero-knowledge argument of knowledge **ZKAoK**. The user proves to the verifier that they know a credential C and corresponding hash chain $((a_1, d_1), \ldots, (a_h, d_h))$ to the public root $\mathcal{R}$.

Note on Non-Interactive Verification: With the exception of step (g), it is noted that every other step in the verification protocol is non-interactive (see (Algorithm 2). In theory, we could replace the proof system π in step (g) with a (quantum-safe) non-interactive zero-knowledge proof [19]. However, these proof systems are inefficient for practical applications. Moreover, the interactive zero-knowledge proofs are non-transferable.[2] In comparison, non-interactive zero-knowledge proofs are transferable [20] by design. Therefore, we believe that for anonymous credentials, interactive verification is preferable to non-interactive verification.

4.4 Revocation

To revoke the credential of a user U_i, an organization first removes the leaf $\mathcal{H}(C_i)$ and then sends all other users local to that tree a tuple containing the value to be replaced and the position of the value in their own hash chain. Each user will update their hash chain based on the tuple received. The communication and run-time costs of revocation are much higher than the cost of joining. This is because any revocation requires the organization to send messages to all affected users and requires $O(h)$ computations of the hash function. By contrast, the joining protocol requires a single message broadcast containing a root update to affected users, the cost of which is $O(1)$ hash application for the organization.

4.5 Credential Updates and Scalability

An organization needs to update and publish roots whenever a new user is on-boarded or the credentials of existing users are revoked. Additionally, certain users affected by the update may also need to update their hash chains before verification. Here, we describe how our system will handle the updates.

Adding Users As new users are enrolled in the system, the Merkle tree must be augmented with their credentials. This necessitates updating the root and, by extension, the hash chain of neighbouring users. Enrollment proceeds as follows, inspired by the construction in [6]:

1. Let us suppose there exists a tree that contains two users, has a corresponding root, r_1, is time-stamped and made publicly available, and possesses a height of 1.

[2] A verifier can't use the transcript of the proof to convince a third party that the statement is true, except with a small probability.

1. User U
   (a) Picks random strings $\alpha_0 \xleftarrow{\$} \{0,1\}^{|N|}$ and $\alpha_j, \beta_j, \gamma_j \xleftarrow{\$} \{0,1\}^{|N|}$ for $j \in [h]$.
   (b) Sets
   $$(s_j, s'_j) = \begin{cases} \big(\alpha_{j-1} \oplus \beta_j \oplus \gamma_j, \alpha_j \oplus \beta_j \oplus \gamma_j\big), & \text{if } j = 1. \\ \big(\beta_j \oplus \gamma_j, \alpha_j \oplus \beta_j \oplus \gamma_j\big), & \text{if } j \neq 1 \wedge j \in [h]. \end{cases}$$
   These values determined the error terms accumulated during verification.
   (c) Computes the final error term $\mathcal{E}_h$ recursively,
   $$\mathcal{E}_j := \begin{cases} \mathcal{H}\big(\mathcal{H}(s_j)||\mathcal{H}(s'_j)\big), & \text{if } j = 1. \\ \mathcal{H}\big((\mathcal{E}_{j-1} \oplus \mathcal{H}(s_j))||\mathcal{H}(s'_j)\big) & \text{if } j \neq 1 \wedge j \in [h]. \end{cases}$$
   (d) Sets
   $$x_j = \begin{cases} C \oplus \alpha_j, & \text{if } j = 0. \\ a_j \oplus \mathcal{H}(\alpha_j) & \text{if } j \in [h]. \end{cases}$$
   (e) Sets the correction words for all $j \in [h]$. [a]
   $$\big(w_j, w'_j\big) = \begin{cases} \big(\beta_j, \mathcal{H}(\gamma_j)\big), & \text{if } (j = 1) \wedge (d_j = \texttt{rightchild}). \\ \big(\beta_j \oplus C, \mathcal{H}(\gamma_j) \oplus a_j\big) & \text{if } (j = 1) \wedge (d_j = \texttt{leftchild}). \\ \big(\mathcal{H}(\beta_j), \mathcal{H}(\gamma_j)\big) & \text{if } (j \geq 2) \wedge (d_j = \texttt{rightchild}). \\ \big(\mathcal{H}(\beta_j) \oplus a'_j, \mathcal{H}(\gamma_j) \oplus a_j\big) & \text{if } (j \geq 2) \wedge (d_j = \texttt{leftchild}). \end{cases}$$
   where we define $a'_1 = \mathcal{H}\big(C\big)$ and
   $$a'_j = \begin{cases} \mathcal{H}\big(a'_{j-1}||a_{j-1}\big) & \text{if } (j \geq 2) \wedge (d_j = \texttt{rightchild}). \\ \mathcal{H}\big(a_{j-1}||a'_{j-1}\big) & \text{if } (j \geq 2) \wedge (d_j = \texttt{leftchild}). \end{cases}$$
   (f) Sends $\big((x_0, \ldots, x_h), ((w_1, w'_1), \ldots, (w_h, w'_h)), \mathcal{E}_h\big)$.
   (g) Provides an interactive zero-knowledge proof of knowledge π to the verifier that the user knows α_0, and $\big(\alpha_j, \beta_j, \gamma_j\big)$ for $j \in [h]$, which determines the error term $\mathcal{E}_h$. [b]
2. Verifier V
   (a) After verifying the proof π.
   (b) $x'_1 := \mathcal{H}(x_0)$.
   (c) For $j = 1$ to h
   If $j = 1$, Set $w := \mathcal{H}(w_j) \oplus w'_j$
   Else, Set $w := w_j \oplus w'_j$
   Set $x'_{j+1} := \mathcal{H}\big((x'_j||x_j) \oplus (w||w)\big)$
   (d) Accepts if and only if $\mathcal{R} = x'_{h+1} \oplus \mathcal{E}_h$.

[a] The correction words specify (in zero-knowledge) if d_j is a `left` or a `right` child.
[b] For π we can use interactive **ZKPoK** presented in [21] or any other quantum-safe **ZKAoK**.

Algorithm 2: `Verification` Protocol

2. When two additional users are enrolled, a new tree of height 1 is created with a new root, r_2, which is time-stamped and made publicly available.
3. Since the old tree and the new tree are of equal height, the two trees are merged and a new tree of height 2 with a root, $r_3 = \mathcal{H}(r_1||r_2)$, is created. After creation, r_3 is time-stamped and made publicly available.
4. When two trees are merged, the old roots r_1 and r_2 become nodes under the new tree but retain their hierarchy.

We can extend this idea to k users over the time interval T_i, where p trees exist of varying heights, each with the most recently published root $r_{p,i}$. Existing users may locally possess an old root $r_{p,q}$, where $0 \leq q \leq i$ for some positive integer q. The update process for users is described as follows:

1. For a given user, the local root, $r_{p,q}$, is $i - q$ steps distant from the latest root, $r_{p,i}$.

2. The user device requests the sequence of roots: $R = \{r_{p,q+1}, r_{p,q+2}, \cdots, r_{p,i}\}$.
3. For each root, r, from the set R, the user will update its hash chain by adding (r, d_i) to its hash chain, where r and $d_i =$ `left child` if a user is from the left sub-tree and r and $d_i =$ `right child` otherwise, thereby preserving the concatenation order.

In the proposed system, multiple Merkle trees of varying heights will exist simultaneously, with two trees merged periodically when their heights are equal. Additionally, the approach necessitates frequent root updates in the initial phases, which may be cumbersome for users and verifiers. To minimize disruption, a threshold minimum number of users, such as 2^{16}, may be required to register with the service prior to constructing the initial tree. Additional users may be enrolled in batches of equivalent size to guarantee a minimum lower-bound on the height of the tree and to preserve user privacy. Subsequent merges limit any attempts to profile users based on their local root.

5 Security Analysis

The scheme described in Section 4 is an anonymous credential scheme that can be shown to satisfy the following security properties: `unforgeability`, `unlinkability` (see Section 3). A formal treatment of the verification protocol is presented first, with mathematical proofs for the core properties of completeness, soundness, argument of knowledge, perfect zero knowledge, and efficiency. Then, a practical instantiation, taking the form of a trusted setup for joining coupled with an in-field verification, is assessed under the Dolev-Yao [22] adversarial model, where the mathematical properties are applied to prove the protocol's security under the model's assumptions.

5.1 Proof of Verification Protocol

Theorem 1. *The verification protocol (Algorithm 2) is a perfect zero-knowledge argument of knowledge (i.e., the prover has the knowledge of* $(C, ((a_1, d_1), \ldots, (a_h, d_h)))$*. Moreover, the prover runs in polynomial time.*

Proof. The verification protocol satisfies the following five conditions:

Completeness : Suppose that U is an honest user who possesses a credential C and a corresponding hash chain $((a_1, d_1), \ldots, (a_h, d_h))$ to the public root $\mathcal{R}$. The verifier receives $((x_0, \ldots, x_h), ((w_1, w_1'), \ldots, (w_h, w_h'), \mathcal{E}_h)$ and the proof π. First the verifier checks proof π, then proceeds. We will prove the completeness by mathematical induction.

1. **Base Case**: For a Merkle tree of height $h = 1$, the verifier received $(x_0, x_1))$, (w_1, w_1'), $\mathcal{E}_1$. Then computes $x_1' := \mathcal{H}(x_0)$,

$$\begin{aligned} x_2' &:= \mathcal{H}((x_1'||x_1) \oplus (w||w)) \\ &= \mathcal{H}\big((x_1' \oplus w)||(x_1 \oplus w)\big) \text{ (by } \textit{Lemma } 1) \\ &= \mathcal{H}\big(((\mathcal{H}(C \oplus \alpha_0) \oplus w)||(a_1 \oplus \mathcal{H}(\alpha_1) \oplus w)\big) \\ &= \mathcal{H}\big((\mathcal{H}(C) \oplus \mathcal{H}(\alpha_0) \oplus w)||(a_1 \oplus \mathcal{H}(\alpha_1) \oplus w)\big). \end{aligned}$$

   If $d_1 =$ `leftchild`, then $w = \mathcal{H}(C \oplus \beta_1 \oplus \gamma_1) \oplus a_1$ and we have

$$\begin{aligned} x_2' &= \mathcal{H}\big((a_1 \oplus \mathcal{H}(\alpha_0 \oplus \beta_1 \oplus \gamma_1))||(\mathcal{H}(C) \oplus \mathcal{H}(\alpha_1 \oplus \beta_1 \oplus \gamma_1)\big) \\ &= \mathcal{H}\big((a_1 \oplus \mathcal{H}(S_1))||(\mathcal{H}(C) \oplus \mathcal{H}(S_1')\big) \\ &= \mathcal{H}\big((a_1||\mathcal{H}(C)) \oplus (\mathcal{H}(S_1)||\mathcal{H}(S_1'))\big) \text{ (by } \textit{Lemma } 1) \\ &= \mathcal{H}\big(a_1||\mathcal{H}(C)\big) \oplus \mathcal{H}\big(\mathcal{H}(S_1))||\mathcal{H}(S_1')\big) \text{ (by Homomorphic Property.)} \\ &= \mathcal{R} \oplus \mathcal{E}_1. \end{aligned}$$

Therefore $x'_2 \oplus \mathcal{E}_1 = \mathcal{R}$.
If $d_1 = \texttt{rightchild}$, then $w = \mathcal{H}(\beta_1 \oplus \gamma_1)$ and

$$x'_2 := \mathcal{H}\big((\mathcal{H}(C) \oplus \mathcal{H}(\alpha_0) \oplus w)||(a_1 \oplus \mathcal{H}(\alpha_1) \oplus w)\big)$$

$$= \mathcal{H}\big((\mathcal{H}(C) \oplus \mathcal{H}(\alpha_0 \oplus \beta_1 \oplus \gamma_1))||(a_1 \oplus \mathcal{H}(\alpha_1 \oplus \beta_1 \oplus \gamma_1))\big)$$

$$= \mathcal{H}\big((\mathcal{H}(C) \oplus \mathcal{H}(S_1))||(a_1 \oplus \mathcal{H}(S'_1)\big)$$

$$= \mathcal{H}\big((\mathcal{H}(C)||a_1) \oplus (\mathcal{H}(S_1)||\mathcal{H}(S'_1))\big) (\textbf{by } \textit{Lemma } 1)$$

$$= \mathcal{H}\big((\mathcal{H}(C)||a_1\big) \oplus \mathcal{H}\big(\mathcal{H}(S_1)||\mathcal{H}(S'_1))\big) \text{ (by Homomorphic Property.)}$$

$$= \mathcal{R} \oplus \mathcal{E}_1.$$

This means $x'_2 \oplus \mathcal{E}_1 = \mathcal{R}$.

2. **Induction Step**: Suppose the verification Algorithm 2 satisfies completeness for any Merkle of height h for $h \geq 1$. We have to show that $x'_{h+1} \oplus \mathcal{E}_h$ is the public root of the Merkle tree of height h. We have

$$x_{h+1} = \mathcal{H}\big((x'_h||x_h) \oplus (w||w)\big)$$

$$x_{h+1} = \mathcal{H}\big((a'_h \oplus \mathcal{E}_{h-1} \oplus w)||(a_h \oplus \mathcal{H}(\alpha_h) \oplus w)\big)$$

if $d_h = \texttt{leftchild}$ then $w = \mathcal{H}(\beta_h \oplus \gamma_h) \oplus a'_h \oplus a_h)$. We have:

$$x_{h+1} = \mathcal{H}\big((a'_h \oplus \mathcal{E}_{h-1} \oplus \mathcal{H}(\beta_h \oplus \gamma_h) \oplus a'_h \oplus a_h))||(a_h \oplus \mathcal{H}(\alpha_h) \oplus \mathcal{H}(\beta_h \oplus \gamma_h) \oplus a'_h \oplus a_h))\big)$$

$$x_{h+1} = \mathcal{H}\big((a_h \oplus \mathcal{E}_{h-1} \oplus \mathcal{H}(s_h))||a'_h \oplus \mathcal{H}(S'_h)\big) \text{ (by } \textit{Lemma } 1)$$
$$x_{h+1} = \mathcal{H}\big((a_h||a'_h) \oplus ((\mathcal{E}_{h-1} \oplus \mathcal{H}(s_h))||\mathcal{H}(S'_h)\big) \text{ (by Homomorphic property.)}$$
$$x_{h+1} = \mathcal{R} \oplus \mathcal{E}_h.$$

A similar argument shows that if $d_h = \texttt{rightchild}$, then $w = \mathcal{H}(\beta_h \oplus \gamma_h)$, and we have $x_{h+1} = \mathcal{H}\big(a'_h||a_h\big) \oplus \mathcal{E}_h$. We conclude that completeness holds for every positive integer h.

Computational Soundness: Informally speaking, the structure of the Merkle tree ensures that a successful verification process, in the absence of possessing a credential and the corresponding hash chain, implies that an adversary has found a collision, which is considered computationally infeasible for a secure hash function.

Formally, we first note that each user picks their own credential and shares the hash of the credential only with the issuing authority (Section 4.2). Now suppose a polynomial-time adversary has learned all the hash chains in the Merkle tree. This is possible if, for example, an adversary colludes with the issuer. We show that a polynomial-time adversary who has complete knowledge of the Merkle tree cannot cheat under the assumption that the underlying hash function is collision-resistant.[3]

Suppose that an adversary knows all the paths from the leaf to the root $\mathcal{R}$. Let h be the height of the Merkle tree and $\mathcal{L}_i$ the set of all nodes at height i. For example, $\mathcal{L}_h$ contains only the root and L_0 contains all the leaves in the tree. To pass the verification process, an adversary has to construct

$$\big((x_0, \ldots, x_h), ((w_1, w'_1), \ldots, (w_h, w'_h)), \mathcal{E}_h)\big).$$

and a proof π of well-formedness of the error term $\mathcal{E}_h$ (Section 4.2). The proof π ensures that the error terms are in the range of $\mathcal{H}$. Suppose an adversary passed the verification process. We have $x'_1 := \mathcal{H}(x_0)$. Note that it is computationally infeasible to find a string x_0 and t_0 such that $x'_1 \oplus \mathcal{H}(t_0) \in L_0$. Otherwise, the adversary has found a pre-image $x_0 \oplus t_0$ of some leaf node in the Merkle. In general, given

$$(x_0, x_1, \ldots, x_i), (((w_1, w'_1), \ldots, (w_i, w'_i)), \mathcal{E}_i))).$$

[3] By complete knowledge we mean, an adversary knows all the nodes in the Merkle tree but not the preimages of leaf nodes.

for any height $0 \leq i \leq h$ and proof π, it is infeasible to find a $x'_{i+1} := \mathcal{H}(x'_i||x_i)$ and a string t_i such that $x_{i+1} \oplus \mathcal{H}(t_i) \in L_i$. Otherwise, the adversary has uncovered a pre-image $((x'_i||x_i) \oplus t_i)$ such that $\mathcal{H}((x'_i||x_i) \oplus t_i) \in L_i$. Therefore a polynomial-time adversary can't cheat except with negligible probability.

Prover/User Efficiency : Note by construction we have $h < |N|$. The user picks $O(h)$ random strings of size $|N|$, computes $O(h)$ exclusive or of strings of size $|N|$ and performs $O(h)$ hash computation on strings each of size at most $|N|$. All of these operations can be done in polynomial-time in $|N|$. The Zero-knowledge proof of knowledge π can be in polynomial-time [21]. Therefore, the prover runs in polynomial-time.

Argument of Knowledge : The extractor described as follows outputs the credential C and the hash chain $((a_1, d_1), (a_2, d_2), \ldots, (a_h, d_h))$.

1. Simulate the prover to output $((x_0, \ldots, x_h), ((w_1, w'_1), \ldots, (w_h, w'_h)), \mathcal{E}_h)$ and π, placing the prover in state Q.
2. Run the extractor for π with input $\mathcal{E}_h$ and recover α_0, $\alpha_j, \beta_j, \gamma_j, \alpha_j$ for all $j \in [h]$.
3. Recover $C, a_1, \ldots, a_h$ as follows:

$$C := x_0 \oplus \alpha_0.$$
$$a_j := x_j \oplus \mathcal{H}(\alpha_j), \text{ for } j \in [h].$$

4. Recover the path for all $j \in [h]$.

$$d_j := \begin{cases} \texttt{right child}, \text{ if } w'_j = \mathcal{H}(\gamma_j). \\ \texttt{left child}, \text{ otherwise.} \end{cases}$$

5. Output $(C, (a_1, d_1), \ldots, (a_h, d_h))$.

Algorithm 3: Extractor

Perfect Zero-knowledge : Let S denote a simulator (Algorithm 4) and V^* denote a possibly malicious verifier. Clearly, the real transcript and the simulated transcripts are identically distributed.

5.2 Security Properties

The scheme described in Section 4 satisfies the two important security properties of an anonymous credential: `Unforgeability` and `Unlinkability` (Section 3).

- `Unforgeability`: This property follows from the proof of computational soundness (Section 5.1). A cheating adversary who colludes with the issuer or other participant cannot impersonate a user U without knowing his credential C (see computational soundness Section 4.3).
- `Unlinkability`: This property follows from the fact that the verification protocol is zero-knowledge (Section 5.1). This means that if the verification sessions of the same user can be linked, then the corresponding transcripts can also be linked. However, this violates the zero-knowledge property of the verification protocol.

6 Adversarial Model Analysis

An attack surface analysis of the proposed scheme can be discussed in the context of an adversarial model. For the present scheme, we assume a Dolev-Yao adversary model [22]: The model assumes an adversary, `A`, that possesses significant computational resources and can observe all channels and can modify, inject, or remove any packets on one or more channels at any time. Additionally, the adversary can create new channels

1. Pick random strings $\alpha'_0 \xleftarrow{\$} \{0,1\}^{|N|}$ and $\alpha'_j, \beta'_j, \gamma'_j \xleftarrow{\$} \{0,1\}^{|N|}$ for $j \in [h]$.
2. Set
$$(\sigma_j, \sigma'_j) = \begin{cases} (\alpha'_{j-1} \oplus \beta'_j \oplus \gamma'_j, \alpha'_j \oplus \beta'_j \oplus \gamma'_j), & \text{if } j = 1. \\ (\beta'_j \oplus \gamma'_j, \alpha'_j \oplus \beta'_j \oplus \gamma'_j), & \text{if } j \neq 1 \wedge j \in [h]. \end{cases}$$
3. Compute the final error term $\mathcal{E}'_h$
$$\mathcal{E}'_j := \begin{cases} \mathcal{H}(\mathcal{H}(\sigma_j)||\mathcal{H}(\sigma'_j)) & \text{if } j = 1. \\ \mathcal{H}((\mathcal{E}_{j-1} \oplus \mathcal{H}(\mathcal{H}(\sigma_j))||\mathcal{H}(\sigma'_j)) & \text{if } j \neq 1 \wedge j \in [h]. \end{cases}$$
4. Pick $C', C'_1, \ldots, C'_h \xleftarrow{\$} \{0,1\}^N$ and $d_j :=$ `left child` for $j \in [h]$.
5. Set
$$x'_j = \begin{cases} C' \oplus \alpha'_j, & \text{if } j = 0. \\ \mathcal{H}(C'_j) \oplus \mathcal{H}(\alpha'_j) & \text{if } j \in [h]. \end{cases}$$
6. Set the correction words
$$(\omega_j, \omega'_j) = \begin{cases} (\beta_j, \mathcal{H}(\gamma_j)), & \text{if } j = 1. \\ (\mathcal{H}(\beta_j), \mathcal{H}(\gamma_j)), & \text{if } j \neq 1 \wedge j \in [h]. \end{cases}$$
7. Set $X' := (x'_0, \ldots, x'_h)$, $\Omega' := ((\omega_1, \omega'_1), \ldots, (\omega_h, \omega'_h))$ and $\mathcal{E}'_h$.
8. Call V^* with input $(X', \Omega, \mathcal{E}'_h)$.
9. Provide a zero-knowledge proof of knowledge π' that the error term $\mathcal{E}'$ is well-formed.
10. Output the$(X', \Omega', \mathcal{E}'_h, \pi')$ and the transcript of π'.

Algorithm 4: Simulator S - Interactive Verification

between any entities operating in the protocol. Given that the adversary has full control over the channel, it can perform a variety of attacks, including impersonating entities, tampering with packets, removing packets, and injecting packets, among others. Under this model, a protocol is considered secure if the only attack possible is a denial of service leading to a failed or aborted transaction.

6.1 Security of `Join`

We first consider the security properties of the `Join` protocol (Algorithm 1). For evaluation purposes, `Join` consists of a single channel constructed between a new User and the Issuing Authority. The information exchange constitutes a bootstrap operation, where both parties are interacting for the first time without prior interaction or knowledge. It is therefore susceptible to a Dolev-Yao attack if `Join` is conducted over an insecure channel or if both parties are malicious.

The mitigations as follows are typical for any bootstrap environment: namely, a semi-honest setting (Issuing Authority is honest) is required, the identity document validation process is trusted, and any data exchange between the User and Issuing Authority occurs over a secure channel relying on a shared, ephemeral, or long-lived secret. Additionally, enrollment relies on an established KYC process conducted by trusted entities to validate users and filter out fraudulent attempts. The trusted setup requirements for `Join` are considered to be reasonable given that `Join` occurs only once in the life cycle of a credential.

6.2 Security of `Verification`

We then consider the security properties of the `Verification` protocol Section 4.3 for an in-field authentication of a credential. The protocol consists of a single channel between two parties, the Prover, P, and the Verifier, V. The protocol is a three-round interactive sequence where P sends a message, V transmits a random challenge c to P, and P transmits a valid response consisting of vectors x, w, proof π and error term $\mathcal{E}_h$ to V (step g of the protocol). Salient attacks are as follows:

– *Replay*: The attacker, M, monitors the channel between P and V and records the transcript, $T = \{\pi, x, w, \mathcal{E}_h\}$, of a successful verification. Then M establishes a new channel with V and replays T, attempting to authenticate as P. This attack will fail as V would have a prior record of T. A variation

of the attack is as follows: M records the transcript between P and V, then establishes a channel with another Verifier, V', and replays T to authenticate as P. This attack also fails as c did not originate from V', and thus V' will reject the request.
- *Tampering and Replacement*: M can modify any number of bits in T in an attempt to impersonate another Prover P'. This is proven to be computationally infeasible by the Soundness and Completeness proofs detailed in Section 5.1. Additionally, M could randomly modify or remove some or all of the bits in T. A malformed T will be rejected by V, causing the interaction to fail. This is equivalent to a denial of service attack.
- *Reverse-engineering*: The attacker, M, could collect transcripts from multiple prover-verifier sequences and attempt to leak credential-specific bits through reverse-engineering. The proof for Perfect Zero-Knowledge in Section 5.1 suggests that this is computationally infeasible.
- *Brute Force*: M can attempt to brute-force a credential by guessing all possible combinations. Since $|C|$ is picked randomly from all possible $2^{|N|}$ strings and by design there are at most $2^{|N|/2}$ number of credentials, the probability of finding a valid credential is $2^{\frac{-|N|}{2}}$, which is negligible in $|N|$.
- *Hash Collision*: M can attempt to find pre-image C' that is equal to hash of some credential $\mathcal{H}(C)$. According to the birthday paradox, to find a collision with probability $1/2$, M is expected to perform

$$\sqrt{2^{|\mathcal{H}(C)|}}$$

computation of the hash function. To perform $\sqrt{2^{|\mathcal{H}(C)|}}$ computations for any $|H(C)| \geq 256$ is computationally infeasible. Note that $|\mathcal{H}(C)|$ denotes the size of the output of the hash function $\mathcal{H}$.

As per the analysis, *tampering* is the strongest attack possible, leading to a denial of service attack. Since denial of service is indistinguishable from a premature disconnection or other physical channel anomaly, we can conclude that the *Verification* protocol is secure, secret-free, and quantum-safe.

7 Communication and Run-time Complexity

7.1 Cost of `Join`

The *communication cost* of Algorithm 1 is at most $O(\log(N))$. The size of the messages exchanged between a user and the organization is

$$\textit{communication cost} = |\mathcal{H}(C)| + \left(\sum_{j=1}^{h} |a_j|\right) + \left(\sum_{j=1}^{h} |d_j|\right).$$

Note that $\mathcal{H}(C)$ is a constant and each d_j can be represented by a bit. We have

$$\textit{communication cost} = O(1) + h \cdot O(1) + h \in O(h).$$

Since $h \in O(log(N))$, the *communication cost* of `Join` can at most be $O(\log(N))$.

7.2 Cost of `Verification`

The *communication cost* of Algorithm 2l is at most $O(\log(N)) + |\pi|$.

$$\textit{communication cost} = \sum_{j=0}^{h} |x_j| + \sum_{j=1}^{h} \left(|w_j| + |w'_j|\right) + |\mathcal{E}| + |\pi|.$$

Note that $x_j, w_j, w'_j, \mathcal{E}_h \in O(1)$ for $j \in [h]$ and $x_0, h \in O(\log(N))$. Therefore, *communication cost* $= O(\log(N)) + |\pi|$.

The communication of the proof of knowledge given in [21] is $O(\log(N)\log(\log(N))$. If employed as π, then the communication cost of `Verification` is in $O(\log(N)\log(\log(N))$.

7.3 Cost of `Revocation`

As mentioned in Section 4.4, revocation is costly. Whenever a credential is revoked, an organization **O** needs to send $O(1)$ number of messages to every user in that tree so that they can update their hash chain. Given a tree of height h, revocation of a single credential requires **O** to send $O(2^h)$ messages and $O(h)$ hash computations. Therefore, the communication cost of the organization is at most $O(N)$.

7.4 Instantiation with SWIFFT

The computed communication cost of `Join` and `Verification` protocols, when instantiated with SWIFTT [8] (Section 3.1) and the proof of knowledge given in [21], is illustrated in Table 1. We take $|N| = 1024$ bits and calculate the communication cost for Merkle trees with different heights. Note the cost of `Join` grows linearly with the height of the Merkle tree, h, and is given by the relationship $512h + h + 512 = 513h + 512$-bits.

`Join` **Cost**	**Tree height**
0.001 **MB**	16
0.002 **MB**	32
0.04 **MB**	64
0.08 **MB**	128

`Verification` **Cost**	**Tree height**
0.68 **MB**	16
1.35 **MB**	32
2.7 **MB**	64
5.4 **MB**	128

Table 1. Communication Cost of `Join` and `Verification`.

8 Conclusion and Future Work

Proposed herein is an anonymous credential scheme that was shown to be lightweight in construction and secure against quantum computing threats. Moreover, the scheme is secret-free and does not require the issuing authority (organization) to protect a secret key. Additionally, a simple, quantum-safe, zero-knowledge argument of knowledge of membership in the Merkle tree enables verification of enrollment by multiple verifiers. The security of the scheme was proven mathematically and analyzed under a Dolev-Yao adversary model assumption. The communication costs indicate that the proposed scheme is efficient for the core functions of enrollment and verification. Future work will focus on efficient revocation mechanisms and constructing a practically efficient quantum-safe version of Sander and Ta-Shma's electronic cash system [6].

Bibliography

[1] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *Applied Cryptography and Network Security*, pages 117–136, Cham, 2016. Springer International Publishing.

[2] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 552–565, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[3] NIST. Enhanced Privacy ID. `https://csrc.nist.gov/csrc/media/events/meeting-on-privacy-enhancing-cryptography/documents/brickell.pdf`. Accessed: 2023-02-06.

[4] Dan Boneh, Saba Eskandarian, and Ben Fisch. Post-quantum epid signatures from symmetric primitives. In Mitsuru Matsui, editor, *Topics in Cryptology – CT-RSA 2019*, pages 251–271, Cham, 2019. Springer International Publishing.

[5] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Alessandro Sorniotti. A framework for practical anonymous credentials from lattices. *Cryptology ePrint Archive*, 2023.

[6] Tomas Sander and Amnon Ta-Shma. Auditable, anonymous electronic cash. In *Advances in Cryptology—CRYPTO 1999*, pages 555–572, 1999.

[7] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *Advances in Cryptology — CRYPTO '87*, pages 369–378, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[8] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. Swifft: A modest proposal for fft hashing. In *Fast Software Encryption (FSE)*, pages 54–72. Springer, 2008.

[9] David Derler, Sebastian Ramacher, and Daniel Slamanig. Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography*, pages 419–440, Cham, 2018. Springer International Publishing.

[10] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.

[11] David Chaum and Jan-Hendrik Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In A.M. Odlyzko, editor, *Advances in Cryptology – CRYPTO 1986*, pages 118–167, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[12] Lidong Chen. Access with pseudonyms. In *International Conference on Cryptography: Policy and Algorithms*, pages 232–243. Springer, 1995.

[13] Ivan Bjerre Damgård. Payment systems and credential mechanisms with provable security against abuse by individuals. In *Conference on the Theory and Application of Cryptography*, pages 328–335. Springer, 1988.

[14] Anna Lysyanskaya, Ronald L Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *International Workshop on Selected Areas in Cryptography*, pages 184–199. Springer, 1999.

[15] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, pages 93–118, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[16] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, pages 56–72, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[17] David Chaum, Eugène van Heijst, and Birgit Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 470–484, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[18] Arjen K. Lenstra. *Birthday Paradox*, pages 36–37. Springer US, Boston, MA, 2005. ISBN 978-0-387-23483-0.

[19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for np from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 89–114, Cham, 2019. Springer International Publishing.

[20] Rafael Pass. On deniability in the common reference string and random oracle model. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 316–337, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[21] Raza Ali Kazmi, Cyrus Minwala, and Duc-Phong Le. Privacy-preserving post-quantum credentials for digital payments. In Shin'ichiro Matsuo, Lewis Gudgeon, Ariah Klages-Mundt, Daniel Perez Hernandez, Werner Sam, Thomas Haines, Aleksander Essex, Andrea Bracciali, and Massimiliano Sala, editors, *Financial Cryptography and Data Security. FC 2022 International Workshops: CoDecFin, DeFi, Voting, WTSC, Grenada, Grenada, May 2-6, 2022,*. Springer Cham, 2022.

[22] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983. https://doi.org/10.1109/TIT.1983.1056650.