Quantum-safe Anonymous Credentials from Homomorphic Encryption

Anonymous Author(s)

ABSTRACT

The growing emphasis on privacy, driven by the abundance of electronic data and advances in analytical and artificial intelligence tools, has been reinforced by recent legislation like GDPR and CCPA. Identity management systems (IMS) are at the heart of privacy concerns, facing challenges related to traceability and overdisclosure, both online and offline. Anonymous Credentials, introduced by D. Chaum in 1985, have gained prominence, but recent research is exploring new approaches, often based on lattice-based assumptions, to address quantum computing threats.

This proposal takes a unique approach by leveraging homomorphic encryption and verifiable decryption methods to protect attributes and demonstrate specific computations over signed attributes without full disclosure. It offers advantages in versatile attribute verification, reducing the need for additional credentials, and addressing revocation concerns while maintaining privacy.

The solution is built on three key primitives, including a verifiable decryption mechanism, a method to perform inequalities and private membership tests in homomorphic encryptions, and a variant of the GPV signature suitable for homomorphic verification.

1 INTRODUCTION

The focus on privacy has increased due to the vast amount of electronic data and advances in analytical and artificial intelligence tools. Recent legislation such as GDPR (General Data Protection Regulation) [22] and CCPA (California Consumer Privacy Act) [8] emphasize the need for strict transmission of private information. Identity management systems (IMS) are at the centre of this issue, as they must prove identity traits without leaking unnecessary information. Online IMS can have traceability problems, revealing the frequency of identifications and services contacted by individuals. Offline IMS, based on signed credentials, may transmit more information than necessary and can also have traceability issues when colluding with services.

The concept of Anonymous Credentials was first proposed by D. Chaun in 1985[14], but it was not until 2001 that the first fully functional solution was proposed[11]. Since then, there have been many different proposals targeting additional features[10, 18, 23, 33], efficiency[9], standardization[20], and concrete implementations[35]. Recently, there has been a new line of research[2, 5–7, 29, 31] that aims to change the cryptographic assumptions used in these solutions, as current assumptions are believed to become obsolete with the advent of quantum computing. Most of the new proposed solutions for quantum-safe cryptography are based on lattice-based assumptions[45, 46], but different strategies exist, using: (i) lattice-based zero-knowledge proof systems[38], (ii) zkSNARKs[7], or as in the present proposal, (iii) homomorphic encryption to prove attributes without complete disclosure. Each solution should be studied, improved, and compared with others to overcome its challenges.

Although most existing solutions rely on concealing credential attributes within signed commitments, which holders then selectively demonstrate to verifiers using zero-knowledge proofs, our approach takes a different route. We employ homomorphic encryption to shield these attributes and employ verifiable decryption methods to demonstrate the result of specific computations over signed attributes to verifiers, without disclosing the attributes.

This approach offers a significant advantage: it extends beyond the conventional boundaries of attribute disclosure. For instance, it can validate statements such as "older than 18" without divulging the precise birthdate or necessitating the issuance of additional credentials. Additionally, it can verify attributes such as "address is from an EU member state" by privately inferring this information from the member state's name in the address. Moreover, our solution can verify functions across attributes in various credentials from different issuers without requiring intermediate issuers to issue new credentials, which reduces the need for extensive interaction within the overall system.

Revocation is of paramount relevance in any solution with long-lived credentials. However, revocation is notably absent in many proposals. Our solution capitalizes on the expressiveness of our attribute-proof system to build a simple revocation mechanism that still preserves privacy.

The drawback of our proposal is the size of the credentials' presentations. While the credentials themselves can be less than $5KB^*$, the presentations of the credentials (sent from the holder to the verifier) vary with the number of attributes and the depth of the function to be proved, which for the depth required by our comparison primitives (Section 6) is $\approx 734\ KB$.

Our solution is based on three primitives that may be of independent interest: (i) a special verifiable decryption primitive, first suggested in [15]; (ii) a primitive for performing simple inequalities and private membership tests in homomorphic encryptions; and (iii) a variant of the GPV signature that can be verified homomorphically.

2 RELATED WORK

At its core[36], anonymous credentials involve three main entities: issuers, holders, and verifiers. Issuers provide credentials to holders, who can present them to verifiers for access to services or acquire additional credentials. Holders must prove ownership of these credentials to verifiers, ensuring that they originate from

Proceedings on Privacy Enhancing Technologies YYYY(X), 1–15
© YYYY Copyright held by the owner/author(s).
https://doi.org/XXXXXXX.XXXXXXX

*Assuming that the holder knows the attributes the issuer only needs to send the signature.

trustworthy issuers. Privacy protection necessitates non-linkable presentations of credentials, even if all verifiers and issuers collaborate. Issuers should not be able to connect credentials issued to the same holder, but holders may present credentials from different issuers to demonstrate affiliation.

Beyond the core concept, subsequent systems for anonymous credentials introduced features such as selective disclosure (revealing specific attributes to verifiers)[10], range proofs (e.g., age > 18), and proofs involving multiple attributes. Revoking credentials while maintaining anonymity is challenging, and there are limited solutions available[33]. Pseudonyms [13] and limited-use credentials (restricting the number of times a credential can be used) are common features[10].

Most proposals adhere to the original framework[11], where holders commit to a secret value and certain attributes, which are signed by issuers and proved to verifiers using a zero-knowledge proof. Efficiently combining a digital signature system and zero-knowledge proof is the primary challenge, with various solutions proposed, including RSA-based schemes, bilinear maps [9, 12], and quantum-resistant approaches[2, 5, 29, 31].

Different research groups, such as Jeudy et al. [29], Blazy et al. [2], and Lai et al. [31], have developed their versions of quantum-resistant anonymous credentials with varying features and trade-offs. Jeudy et al. [29] focus on efficient oblivious signatures and proofs, but their proofs are relatively large ($639KB$). Blazy et al. use a group signature scheme and a lattice-based zero-knowledge framework, but lack selective disclosure and have large proofs ($3.7MB$). Lai et al. [31] build upon a new construction for commitment transferability to ensure unlinkability, but still have relatively large proofs of $500KB$.

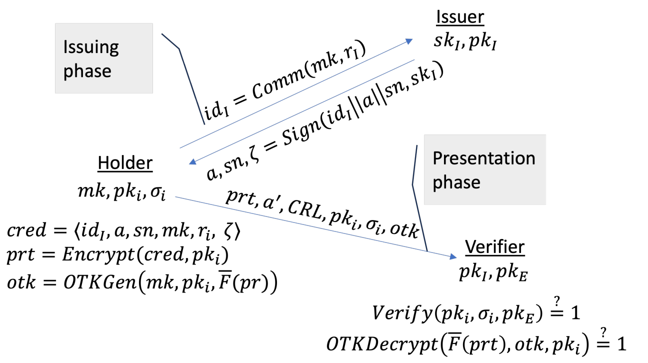
Addressing the challenge of efficiently demonstrating knowledge of random oracle preimages within zero-knowledge frameworks remains a common concern. Some proposals opt for standard-model signature frameworks. Notably, Bootle et al. [5] build a random-oracle independent signature system, albeit relying on unproven assumptions, resulting in small proof sizes ($133KB$ for 16 attributes).

Future developments in quantum-resistant anonymous credential systems are expected, exploring variants such as zkSNARKs on lattices[7], functional encryption[4], and homomorphic signatures[27].

3 PROPOSED SOLUTION

The proposed solution has three phases and uses four techniques described in detail in sections 5,6,7 and 8. The first phase is the setup phase, whose goal is to provide the holder with the necessary keys to participate in the protocol. The second phase is the issuing phase, where the Issuer issues a credential to the Holder. The third and last phase is the presentation phase, where the Holder presents one or several credentials to the Verifier.

During the setup phase, the user is required to register his secret encryption key (mk) with a key escrow service. In response, the key escrow service issues multiple public keys (pk_i) corresponding to the same secret key, with each of these public keys digitally signed (σ_i) by the key escrow service. In addition to the act of signing these public keys, the key escrow service also guarantees the provision of revocable anonymity for holders who engage in malicious or improper behavior.



Issuing phase
Issuer
sk_I, pk_I
$id_I = Comm(mk, r_I)$
$a, sn, \zeta = Sign(id_I||a||sn, sk_I)$
Holder
mk, pk_i, σ_i
Presentation phase
$prt, a', CRL, pk_i, \sigma_i, otk$
$cred = \langle id_I, a, sn, mk, r_i, \zeta \rangle$
$prt = Encrypt(cred, pk_i)$
$otk = OTKGen(mk, pk_i, \overline{F}(pr))$
Verifier
pk_I, pk_E
$Verify(pk_i, \sigma_i, pk_E) \overset{?}{=} 1$
$OTKDecrypt(\overline{F}(prt), otk, pk_i) \overset{?}{=} 1$

Figure 1: Issuing and presentation phases of the proposed solution.

The issuing phase (cf. Figure 1) starts with the holder committing to her secret key (mk) while introducing an element of randomness (r_I). Subsequently, she uses this commitment $id_I = \mathbf{Comm}(mk, r)$ as her unique identifier for a specific issuer I. The issuer, in turn, takes the commitment id_I, some attributes of the holder a and the serial number sn of the credential, signs all with the issuer's secret key sk_I and returns the signature $\varsigma = \mathbf{Sign}(id_I||a||sn, sk_I)$, the attributes a and the sn number back to the holder.

In the presentation phase (cf. Figure 1), the holder encrypts the credential $cred = \langle id_I, a, sn, mk, r_i, \varsigma \rangle$, with a homomorphic encryption algorithm and one of her public keys pk_i.

Then, the holder sends the presentation message, with:

- The resulting presentation ciphertext $prt = \mathbf{Encrypt}(cred, pk_i)$,
- The holder's attributes to be revealed a' (not necessarily the attributes a in the credential $cred$),
- The public key pk_i used to encrypt the credential and the correspondent signature from the escrow service σ_i,
- A decryption key $otk = \mathbf{OTKGen}(mk, pk_i, \overline{F}(prt))$ that can only be used to decrypt a specific homomorphic computation $\overline{F}(prt)$ over the presentation ciphertext prt, and
- A certificate revocation list CRL containing the serial numbers of revoked credentials (which could also be taken from someplace else, e.g. a distributed ledger).

The function **OTKGen** generates a special decryption key otk that allows the delegation of the decryption of a specific ciphertext to the verifier without delegating the decryption of all ciphertext encrypted with the same key. (cf. Section 5).

Upon receiving the presentation message, the verifier:

- Checks the signature σ_i on the public key pk_i of the holder,
- Recomputes and decrypts $\overline{F}(prt)$, with the otk and the holder's public key, and checks if the result is valid

$$\mathbf{OTKDecrypt}(\overline{F}(prt), otk, pk_i) \overset{?}{=} 1$$

The homomorphic computation $\overline{F}(prt)$ that both the holder and the verifier must perform on the encrypted credential prt, is the conjunction of four elements $\overline{F}(prt) = prt_{oc} \wedge prt_v \wedge prt_{CRL} \wedge prt_a$, each of them being the result of a specific homomorphic test (see Table 1). The first test prt_{oc} checks that the user owns the commitment id_I, opening the commitment with the secret key of the holder mk and the randomness r_I used to create the commitment. The second test prt_v checks the signature of the credential with the public key of the issuer pk_I. The third test prt_{CRL} checks that the credential serial number is not in the issuer's CRL, and finally the last test prt_a checks that the derived attributes a' are consistent with the signed attributes a. The homomorphic test $\overline{Check}$ depends on the specific attributes a and a', for example, if a contains an address with an EU member state and the holder wants to prove that that address is from an EU member state, it can be implemented with the "Private Set Membership" primitive $\overline{\textbf{PSM}}(prt, EU)$, which tests that the member state in the address is one of the EU member states, without revealing the member state in the address.

Element	Homo. Test	Result
prt_{oc}	$\overline{\textbf{OpenCom}}(prt)$	$\mathcal{E}_{pk_i}\,(\textbf{OpenCom}(id_I, mk, r_I))$
prt_v	$\overline{\textbf{Verify}}(prt, pk_I)$	$\mathcal{E}_{pk_i}\,(\textbf{Verify}(id_I \Vert a \Vert sn, \varsigma, pk_I))$
prt_{CRL}	$\overline{\textbf{PSM}}(prt, CRL)$	$\mathcal{E}_{pk_i}\,(sn \notin CRL)$
prt_a	$\overline{Check}(prt, a')$	$\mathcal{E}_{pk_i}\,(Check(a, a'))$

Table 1: Homomorphic tests performed over the credential presentation.

Note that the credential presentation is not traceable by any combination of verifiers and issuers because it is always a fresh encryption and the encrypted values are never revealed to the verifier.

The following section introduces some background and notation required by the remaining sections, which describe each of the previously used primitives. Section 5 describes the primitives **Encrypt**, **OTKGen**, and **OTKDecrypt**, while the primitive $\overline{\textbf{PSM}}$ is described in Section 6. Section 7 describes the primitives **Comm**, **OpenCom** and $\overline{\textbf{OpenCom}}$, and the primitives **Sign**, **Verify**, and $\overline{\textbf{Verify}}$ are described in Section 8.

4 BACKGROUND

4.1 Notation

A number is denoted by a lowercase letter, for example, $z \in \mathbb{Z}$. We denote generic sets by a capital letter S and the size of S by $|S|$. The vectors will be written in column form using the arrow notation $\vec{v}$, where v_i denotes the i^{th} element. The inner product of two vectors is denoted by $\langle \vec{x}, \vec{y} \rangle = \vec{x}^{\,T}\, \vec{y}$. Unless stated otherwise, the norm of a vector $\vec{v}$ is the l_2 norm and is denoted by $\| \vec{v} \|$. The set of residue classes modulo q is denoted by $\mathbb{Z}_q$, and the class representatives of $\mathbb{Z}_q$ are taken from the half-open interval $[-q/2, q/2)$.

Let $A \in \mathbb{Z}_q^{n \times m}$, denote

$$\Lambda_A^{\perp} = \{\vec{z} \in \mathbb{Z}^m : A\,\vec{z} = \vec{0} \bmod q\}$$

as the q-ary lattice of all elements $\vec{z}$ that satisfy the relation $A\,\vec{z} = \vec{0} \bmod q$.

A random variable x sampled from a distribution $\mathcal{D}$ is $x \leftarrow \mathcal{D}$. For a vector $\vec{v} \in \mathbb{R}^n$, a positive real s, and a lattice $\Lambda \subset \mathbb{R}^n$, let $\mathcal{D}_{\Lambda,\vec{v},s}$ denote the n-dimensional discrete Gaussian distribution over Λ, centered at $\vec{v}$, with parameter s. We use $\mathcal{U}_S$ to denote a discrete uniform distribution over a set S, $\mathcal{T}$ denotes a discrete ternary uniform distribution over $\{-1, 0, 1\}$, and χ denotes an error distribution $\mathcal{D}_{\Lambda,\vec{0},\sigma}$, with σ satisfying the Ring Learning with Errors (RLWE) security reduction [41] for the security parameter λ. The default value σ in the HELIB* and PALISADE† libraries is $\sigma = 3.2$.

We will use the following simple lemmas, for which we sketch the proof of the first.

Lemma 1. *For any odd prime number p and $0 \leq x < p$, $x(x-1) = 0 \bmod p$ iff $x = 0$ or $x = 1$.*

The proof is straightforward. Since p is prime and $x(x-1) = k.p$ for some $k \in \mathbb{Z}$ then either $x = r.p$ or $x - 1 = r.p$ for some $r \in \mathbb{Z}$, and because the only $r \in \mathbb{Z}$ satisfying $0 \leq x < p$ is $r = 0$, then $x = 0$ or $x = 1$

Lemma 2. *For any odd prime number p and $-p/2 \leq x < p/2$, $(x+1)(x-1) = 0 \bmod p$ iff $x = -1$ or $x = 1$ and $x(x+1)(x-1) = 0 \bmod p$ iff $x = 0, x = -1$ or $x = 1$.*

4.2 Cyclotomic Rings

Let $R = \mathbb{Z}[X]/\langle \Phi_m \rangle$ be the cyclotomic ring, isomorphic to the extension field $\mathbb{Q}(\xi_m)$ constructed by adjoining a primitive m^{th} root of unity, where m is a positive integer and $\Phi_m(X)$ is the m^{th} cyclotomic polynomial, and let $R_q = \mathbb{Z}_q[X]/\Phi_m(X)$ be the module q cyclotomic polynomial. Our implementation uses $\Phi_m(X) = x^d + 1$, with $m = 2d$ and d a power of 2. Whenever relevant, we denote the underlying ring with both parameters by $R_{d,q} = \mathbb{Z}_q[X]/(x^d + 1)$.

We denote by bold lowercase letters the elements of $\mathbf{a} \in R$ and by $\vec{\boldsymbol{a}}$ the vector of elements of R. The i^{th} element of a vector $\vec{\boldsymbol{a}}$ will be denoted by $\mathbf{a}_i$ and the j^{th} coefficient of polynomial $\mathbf{a}$ by a_j. Similarly, $a_{i,j}$ will denote the j^{th} coefficient of the i^{th} polynomial of a vector $\vec{\boldsymbol{a}}$. The l_2 norm $\| \vec{\boldsymbol{a}} \|$ is defined as $\| \vec{\boldsymbol{a}} \| = \sqrt{\sum a_{i,j}^2}$ and the expansion factor of R is defined as $\delta_{\mathcal{R}} = \max \|\langle \vec{\boldsymbol{a}}, \vec{\boldsymbol{b}} \rangle\|/(\| \vec{\boldsymbol{a}} \|.\| \vec{\boldsymbol{b}} \|) : \vec{\boldsymbol{a}}, \vec{\boldsymbol{b}} \in R^k$.

Let $n = \phi\,(m)$ where ϕ is Euler's totient function, and $q > 1$ be a number coprime to m, then $\Phi_m(X)$ splits modulo q into l_Φ irreducible factors of same degree d, i.e. $\Phi_m(X) = \prod_{i=1}^{l_\Phi} F_i(X) \bmod q$, where δ is the order of q modulo m, and $l_\Phi = n/\delta$. Then by Chinese Remainder Theorem (CRT), the modulo q cyclotomic polynomial $R_q = \mathbb{Z}_q[X]/\Phi_m(X)$ is congruent with $R_q \cong \prod_{i=1}^{l_\Phi} \mathbb{Z}_q[X]/\langle F_i(X) \rangle \bmod q$. Each quotient ring $\mathbb{Z}_q[X]/\langle F_i(X) \rangle$ constitutes a Single Instruction Multiple Data (SIMD) slot, isomorphic to the field $\mathbb{F}_{q^\delta}$. Therefore, the additions and multiplications of rings in R_q result in the corresponding coefficient-wise operations of the respective slots. We denote by $\vec{\sigma}(\mathbf{a})$ the canonical embedding vector of the polynomial $\mathbf{a}$ and by $\sigma(\mathbf{a})_i$ the embedded slot i^{th} in the polynomial $\mathbf{a}$. Let $\vec{\boldsymbol{a}} \in R_{d,q}^n$, we denote by $\vec{\sigma}(\vec{\boldsymbol{a}}) = [\vec{\sigma}(\mathbf{a}_0), \ldots, \vec{\sigma}(\mathbf{a}_{n-1})]$, the concatenation of the canonical embedding in every polynomial $\mathbf{a}_i$.

We will use canonical embedding extensively to pack different attributes into a polynomial, and even to pack smaller polynomials $\mathbf{a}, \mathbf{b} \in R_{d,q}$ into a larger polynomial $\mathbf{c} \in R_{d',q}$. Assume that $d|d'$ and $q \bmod 2d = 1$ both rings are fully split into d and d' slots, isomorphic to the field $\mathbb{F}_q$. We say that $\mathbf{c} \in R_{d',q}$ embeds the polynomials $\mathbf{a}, \mathbf{b} \in R_{d,q}$ if $\vec{\sigma}(\mathbf{c}) = [\vec{0}, \vec{\sigma}(\mathbf{a}), \vec{\sigma}(\mathbf{b})]$, where $\vec{0}$ is a padding vector.

*https://github.com/homenc/HElib
†https://gitlab.com/palisade/palisade-release

4.3 Hard Problems

The security of the proposed schema is based on two well-known computational problems, Ring-SIS (**RSIS**) [47][37] and Ring-LWE (RLWE) [41].

Definition 1. *(**RSIS**$_{d,q,m,\beta}$ [32]) Given $\vec{a} \in R^m_{d,q}$ chosen independently of the uniform distribution, with $m \geq \log q$, find $\vec{z} \in R^m_{d,q}$ such that $\langle a, z\rangle = 0 \mod q$ and $0 < \|\vec{z}\| \leq \beta$. An adversary $\mathcal{A}$ is said to have advantage ϵ in solving **RSIS**$_{d,q,m,\beta}$ if it is able to generate a vector $\vec{z} \in R^m_{d,q}$ with norm bounded by $0 < \|\vec{z}\| \leq \beta$ such that $\langle a, z\rangle = 0 \mod q$, with a non-negligible probability.*

Definition 2. *(RLWE$_{d,q,m,\sigma}$ [32]) Let $\mathbf{a}, \mathbf{b} \in R_{d,q}$ be two uniformly random polynomials and let $\mathbf{s}, \mathbf{e} \in R_{d,q}$ be sampled from the Gaussian distribution χ. An adversary $\mathcal{A}$ is said to have an advantage over RLWE$_{d,q,m,\sigma}$ if it is able to distinguish $(\mathbf{a}, \mathbf{b})$ from $(\mathbf{a}, \mathbf{a}.\mathbf{s} + \mathbf{e})$, with a non-negligible probability.*

4.4 Homomorphic Encryption

The presented anonymous credential solution is adaptable and can be implemented using two homomorphic encryption schemes: BGV and BFV. These schemes are quite similar, with the primary distinction being how they handle message scaling and error during encryption. Both the BGV and BFV schemes are compatible within the framework. The encryption and decryption procedures for both schemes are outlined using the adopted notation to provide a comprehensive overview*.

- $(\overrightarrow{sk}, \overrightarrow{pk}) \leftarrow$ **KeyGen**$(p, l, 1^\lambda)$: Both schemas take the plaintext module p, the multiplication depth l and the security factor λ as input for the key generation. In both cases, every element is taken from a ring $R_q = \mathbb{Z}_q/\Phi_m(X)$, with $m = m(\lambda)$ satisfying the security parameter λ, and $q = q(l, p, m)$ satisfying the multiplication depth d. The secret key $\overrightarrow{sk} = [-\mathbf{s}, \mathbf{1}]^T \in R^2_q$ is a vector of polynomials, where $\mathbf{s}$ is a random polynomial with coefficients in $\mathcal{T}$, and $\mathbf{1}$ is a constant polynomial. The public key $\overrightarrow{pk} = [\mathbf{a}, \mathbf{b}]^T$ is a pair of polynomials $\mathbf{a} \leftarrow \mathcal{U}_{R_q}$ and $\mathbf{b} = \mathbf{a}.\mathbf{s} + \tau.\mathbf{e} \mod q$, where $\mathbf{e} \leftarrow \chi$, and $\tau = \begin{cases} p & \text{if BGV} \\ 1 & \text{if BFV} \end{cases}$
- $\vec{c} \leftarrow$ **Encrypt**$(\overrightarrow{pk}, \mathbf{m})$: The encryption procedure takes the public key $\overrightarrow{pk}$ and a polynomial $\mathbf{m} \in R_p$ encoding the message and outputs a ciphertext $\vec{c} = [\mathbf{c}_0, \mathbf{c}_1]^T$ with a pair of polynomials $\mathbf{c}_0 = \mathbf{a}.\mathbf{u} + \tau.\mathbf{e}_1$ and $\mathbf{c}_1 = \mathbf{b}.\mathbf{u} + \tau.\mathbf{e}_2 + \Delta.\mathbf{m}$ where $\mathbf{u} \in R_q$ is a polynomial with coefficients taken from $\mathcal{T}$, $\mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi$ and $\Delta = \begin{cases} 1 & \text{if BGV} \\ \lfloor q/p \rfloor & \text{if BFV} \end{cases}$
- $\mathbf{m} \leftarrow$ **Decrypt**$(\overrightarrow{sk}, \vec{c})$: The decryption process takes the secret key $\overrightarrow{sk}$ and the ciphertext $\vec{c}$ and outputs the message $\mathbf{m} \in R_p$, $\mathbf{m} = \Gamma(\boldsymbol{\rho})$, with
$$\boldsymbol{\rho} = \langle \overrightarrow{sk}, \vec{c}\rangle = \Delta.\mathbf{m} + \tau.\boldsymbol{\partial} \mod q$$
and
$$\Gamma(\mathbf{x}) = \begin{cases} \mathbf{x} \mod p & \text{if BGV} \\ \lfloor p.\mathbf{x}/q \rceil \mod p & \text{if BFV} \end{cases}$$
In both schemas, the decryption algorithm holds if the ciphertext error level $\|\boldsymbol{\partial}\|_\infty < q/2p$.

We denote by $\mathcal{E}(\{x_1, \ldots, x_n\})$ the encryption of a packed vector $\{x_1, \ldots, x_n\}$ isomorphic to the field $\mathbb{F}^n_{p^\delta}$, with either BGV or BFV, and by $\mathcal{E}(\mathbf{a}) = \mathcal{E}(\vec{\sigma}(\mathbf{a}))$ the encryption of a polynomial $\mathbf{a} \in R_{d,p}$ encoding a cannonical embedding vector $\vec{\sigma}(\mathbf{a})$ isomorphic to the field $\mathbb{F}^n_{p^\delta}$. In most cases p and q denote the plaintext and ciphertext modulus of encryption, respectively; however, in some cases, we will encrypt with $\beta.p$ and $\beta.q$, which we denote by $\mathcal{E}_\beta(\mathbf{a})$. We denote with an overbar $\overline{\textbf{Function}}$ the homomorphic computation of a function with the same name **Function**.

4.5 Lattice Trapdoors

In lattice-based encryption, a "trapdoor" is a crucial tool that simplifies the solution of complex problems such as LWE or SIS and their counterparts in ring-based cryptography. These problems are inherently difficult without a trapdoor. The key idea behind lattice trapdoors is that LWE and SIS problems become computationally easy when the lattice has bases characterized by short orthogonal vectors, known as "good bases." Conversely, they become computationally hard when the lattice has bases that are not "good", referred to as "bad" bases"

The ingenious approach involves generating both "good" and "bad" bases simultaneously. The "bad" base is made public, while the "good" base is kept confidential as the trapdoor. In many cases, the "good" base is a well-known fixed base, and the trapdoor is represented by the unimodular transformation that converts one base into the other. Notably, Micciancio and Peikert's [42] solution has been one of the most efficient trapdoor generation methods, demonstrating its effectiveness in solving LWE problems and generating secure solutions for SIS problems.

We are mainly interested in sampling RSIS solutions, that is, finding a polynomial vector $\vec{s} \in R^{k+2}_q$, with a short norm such that $\vec{a}\,.\,\vec{s} \cong \mathbf{c}$, for any polynomial $\mathbf{c} \in R_q$. Let

$$\vec{a}, \vec{r} \leftarrow \textbf{TrapGen}(\lambda)$$

denote the generation of the "bad" ($\vec{a}$) and "good"($\vec{r}$) basis[†], and let

$$\vec{s} \leftarrow \textbf{TrapSampl}(\vec{a}, \vec{r}, \mathbf{c})$$

denote the sampling of a polynomial vector $\vec{s}$, such that the Euclidean norm $\|\vec{s}\| < B_t$ is bounded by some parameter B_t. The sampling algorithm of [42] requires $q = b^k$ for some $k \in \mathbb{Z}$, which is not always the most appropriate choice of q to be used in BGV or BFV, since we need q to be as small as possible for efficiency reasons. Also, we use the solution proposed by [19]. The bound $B_t = \max \|\vec{s}\|$ of the Euclidean norm for the preimage samples sampled with this procedure is

$$B_t < s\sqrt{d(k+2)} \qquad (1)$$

*We omit the remaining homomorphic operations because our solution will not impact them.

[†] $\vec{r}$ is not exactly a good base but one may be generated from it.

where $s = 1.3\sigma^2(b+1)(\sqrt{d.k} + \sqrt{2d} + 4.7)$ is the spectral bound, and $\sigma = \sqrt{\log_2 2d/\epsilon)/\pi}$ is the smoothing parameter from which the trapdoor polynomials are generated.

From [44] we know that the bound β that makes $\mathbf{RSIS}_{d,q,m,\beta}$ hard is given by

$$\beta < \min(p, 2^{\sqrt{4dm \log q \log \delta}}) \tag{2}$$

where δ is the hermite value, which depends on the security parameter λ. Whenever necessary, we will use $\delta = 1.0043$. Therefore, the parameters should be chosen such that $B_t \leq \beta$.

5 VERIFIABLE DECRYPTION

A verifiable decryption mechanism is a system that provides an individual with an encrypted value, a way to compute it, and a proof to confirm the accuracy of the decryption process. In this context, the chosen solution, as described in [15], involves a one-time self-validating key. This key is used to decrypt a specific encryption as it would be if it was decrypted with the secret key corresponding to a specific public key. However, contrary to the secret key, the one-time key cannot be used to obtain information about any other ciphertext, besides the one for which it was generated.

Definition 3. *Let A be a set of attributes of a specific user, V be a set of constant values, and a function $m = f(A, V)$. Let $\mathcal{E}(A)$ be some encryption of A with public key pk and $F(\cdot,\cdot)$ a homomorphic function such that $\mathcal{E}(m) = F(\mathcal{E}(A), V)$. A one-time key k for a tuple (A, V, f, pk) is a key of a special algorithm OTKDecrypt that, for $y = F(\mathcal{E}(A), V)$*

$$OTKDecrypt(k, y) = \begin{cases} m & \text{if } Valid(k, y, pk) \\ \perp & \text{otherwise} \end{cases}$$

5.1 OTK Encryption Schema

The proposed solution adheres to the framework presented in [15] but is tailored to the ring setting with a primary focus on optimizing efficiency. A straightforward approach is to use a Module Learning with Errors (MLWE)-based encryption schema, with a public key matrix $\boldsymbol{PK} = [\mathrm{A}, \vec{\boldsymbol{b}} = \mathrm{A}.\vec{s} + \tau\vec{e}]^T$, where $\mathrm{A}, \mathrm{R} \leftarrow \mathbf{TrapGen}(\lambda) \in R_q^{k\times 2k+2}$, $\vec{s} \leftarrow \mathcal{T}^{k.d}$ and $\vec{e} \leftarrow \chi^{k.d}$ and a ciphertext

$$\mathcal{E}(\boldsymbol{m}) = \begin{bmatrix} \vec{c_0} \\ c_1 \end{bmatrix} = \begin{bmatrix} \mathrm{A}.\vec{\boldsymbol{u}} + \tau\vec{e_1} \\ \langle \vec{\boldsymbol{b}}, \vec{\boldsymbol{u}} \rangle + \tau e_2 + \Delta.\mathrm{m} \end{bmatrix} \tag{3}$$

The trapdoor $\mathrm{R} \in R_q^{k\times k+2}$ generated with $\mathrm{A} \in R_q^{k\times k}$, enables the sampling of a polynomial $\vec{x} = \mathbf{TrapSampl}(\mathrm{A}, \mathrm{R}, \vec{c_0})$ characterized by a small norm, satisfying the equation

$$\mathrm{A}.\vec{x} = \vec{c_0} \tag{4}$$

where $\vec{c_0} \in R_q^k$ is the first element of an encryption $\mathcal{E}(\boldsymbol{m})$ (eq. (3)).

This polynomial $\vec{x}$ can then be used to recover the message $\mathbf{m} = \Gamma(c_1 - \langle \vec{x}, \vec{\boldsymbol{b}} \rangle \bmod q)$ without revealing the secret key $\mathbf{s}$, where c_1 is the second element of the encryption $\mathcal{E}(\boldsymbol{m})$ (eq. (3)).

The polynomial vector $\overrightarrow{\boldsymbol{otk}} = \vec{x}$ acts as a decryption key for that specific ciphertext [15]. This has the immediate consequence of increasing the dimensionality of the ciphertext, transitioning from $2d$ to $d.(k+1)$, and the public key, expanding from $2.d$ to $d.k.(k+1)$. Typically, when transitioning from RLWE to MLWE, the increase in dimensionality is counterbalanced by a reduction in the ring size. However, in our particular scenario, this adjustment would not only diminish the available data slots but would also exert an influence on the depth of homomorphic multiplications.

Instead, we let the public key be $\overrightarrow{\boldsymbol{pk}} = [\vec{\boldsymbol{a}}, \vec{\boldsymbol{b}} = \mathbf{s}\,\vec{\boldsymbol{a}} + \tau\,\vec{\boldsymbol{e}}]^T$, where $\vec{\boldsymbol{a}}, \vec{\boldsymbol{r}} \leftarrow \mathbf{TrapGen}(\lambda)$, and let the corresponding ciphertext be

$$\begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} \langle \vec{\boldsymbol{a}}, \vec{\boldsymbol{u}} \rangle + \tau e_1 \\ \langle \vec{\boldsymbol{b}}, \vec{\boldsymbol{u}} \rangle + \tau e_2 + \Delta.\mathrm{m} \end{bmatrix} \tag{5}$$

Then, it is possible to sample $\vec{x} \in R_q^{k+2}$ such that $\langle \vec{\boldsymbol{a}}, \vec{x} \rangle = c_0$, and recover the message $\mathbf{m} = \Gamma(c_1 - \langle \vec{x}, \vec{\boldsymbol{b}} \rangle)$. With this approach, the size of the ciphertext does not change, and the public key $\overrightarrow{\boldsymbol{pk}}$ increases only by $k+2$, which is much less than with the plain MLWE approach.

The polynomial vector $\vec{\boldsymbol{a}}$ generated from the primitive **TrapGen** has structure $\vec{\boldsymbol{a}} = [1, \boldsymbol{a_0}, \ldots, \boldsymbol{a_k}]$, where $\boldsymbol{a_0}$ is a randomly generated polynomial and $\boldsymbol{a_1}$ to $\boldsymbol{a_k}$ are statistically indistinguishable from random polynomials. As a result, all except one of the $k+2$ columns in the public key $\overrightarrow{\boldsymbol{pk}} = [\vec{\boldsymbol{a}}, \vec{\boldsymbol{b}} = \mathbf{s}\,\vec{\boldsymbol{a}} + \tau\,\vec{\boldsymbol{e}}]^T$ consist of plain RLWE samples, $\overrightarrow{\boldsymbol{pk_i}} = [\boldsymbol{a_i}, \mathbf{s}.\boldsymbol{a_i} + \tau.\boldsymbol{e_i}]$, ensuring that they do not reveal information about polynomial error $\boldsymbol{e_i}$ or secret key $\mathbf{s}$. However, the first column is represented by $\overrightarrow{\boldsymbol{pk_0}} = [1, \mathbf{s} + \tau\boldsymbol{e_0}]^T$, which can potentially expose the secret key if the distribution of the secret key is different from the distribution of $\tau.\boldsymbol{e_i}$. Aligning both distributions resolves this potential issue.

The One Time Key schema is similar to the BGV/BFV schema except for the following algorithms:

– $(\overrightarrow{\boldsymbol{sk}}, \overrightarrow{\boldsymbol{pk}}, \overrightarrow{\boldsymbol{mk}}) \leftarrow \mathbf{KeyGen}(p, d, 1^\lambda)$: The key generation procedure generates one more key than the BGV/BFV schema: the trapdoor master key $\overrightarrow{\boldsymbol{mk}}$, concretely

$$\vec{\boldsymbol{a}}, \overrightarrow{\boldsymbol{mk}} \leftarrow \mathrm{TrapGen}(\lambda)$$
$$\overrightarrow{\boldsymbol{sk}} = [-\mathrm{s}, 1]^T \quad \mathrm{s} \leftarrow \chi^n$$
$$\overrightarrow{\boldsymbol{pk}} = [\vec{\boldsymbol{a}}, \vec{\boldsymbol{b}}]^T \quad \vec{\boldsymbol{b}} = \mathrm{s}.\vec{\boldsymbol{a}} + \tau.\vec{\boldsymbol{e}} \quad \vec{\boldsymbol{e}} \leftarrow \chi^{d(k+2)}$$

– $\vec{\boldsymbol{c}} \leftarrow \mathbf{Encrypt}(\overrightarrow{\boldsymbol{pk}}, \boldsymbol{m})$: The encryption procedure is slower than the original one, given the size of the public key $\overrightarrow{\boldsymbol{pk}} \in R_q^{2(k+2)}$, but it outputs a ciphertext of the same size $\vec{\boldsymbol{c}} = [c_0, c_1]^T \in R_q^2$ (eq. (5)), where $\vec{\boldsymbol{u}} \in R_q^{k+2}$ is a vector of polynomials with coefficients sampled from $\mathcal{T}$.

– $\overrightarrow{\boldsymbol{otk}} \leftarrow \mathbf{OTKGen}(\overrightarrow{\boldsymbol{pk}}, \overrightarrow{\boldsymbol{mk}}, \vec{\boldsymbol{c}})$: The generation of the one-time key $\overrightarrow{\boldsymbol{otk}}$ for a ciphertext $\vec{\boldsymbol{c}}$ takes the master key $\overrightarrow{\boldsymbol{mk}}$, the public key $\overrightarrow{\boldsymbol{pk}}$ and the ciphertext $\vec{\boldsymbol{c}} = [c_0, c_1]^T$. Thus, a one-time key cannot be built before the computation of the ciphertext, which is not ideal. The $\overrightarrow{\boldsymbol{otk}} \in R_q^{k+2}$ is sampled from

$$\overrightarrow{\boldsymbol{otk}} = \mathbf{TrapSamp}(\vec{\boldsymbol{a}}, \overrightarrow{\boldsymbol{mk}}, c_0)$$

– $\mathbf{m} \leftarrow \mathbf{OTKDecrypt}(\overrightarrow{\boldsymbol{pk}}, \overrightarrow{\boldsymbol{otk}}, \vec{\boldsymbol{c}})$: OTK decryption takes the one-time key $\overrightarrow{\boldsymbol{otk}}$, the public key $\overrightarrow{\boldsymbol{pk}} = [\vec{\boldsymbol{a}}, \vec{\boldsymbol{b}}]^T$ and the ciphertext $\vec{\boldsymbol{c}} = [c_0, c_1]^T$ as input and outputs the message $\mathbf{m} \in R_q$ or $\perp$. The algorithm starts by checking that $\|\overrightarrow{\boldsymbol{otk}}\| < B$ is less than the maximum bound (vide eq. (1)), and that $\mathrm{c}_0 - \langle \overrightarrow{\boldsymbol{otk}}, \vec{\boldsymbol{a}} \rangle = \vec{0}$ and outputs $\mathbf{m} = \Gamma\left(c_1 - \langle \overrightarrow{\boldsymbol{otk}}, \vec{\boldsymbol{b}} \rangle \bmod q\right)$, if both conditions hold and $\perp$ otherwise.

5.2 Soundness & Security

It is easy to see that the OTK decryption algorithm is sound, provided that the public key is of the form $\overrightarrow{pk} = [\vec{a}, \vec{b} = \mathrm{s}.\vec{a} + \tau.\vec{e}]$ and the error rate ∂ of the ciphertext $\vec{c}$ does not exceed $\partial < q/2p - \delta_R.n.k.B.B_t$.

$$\mathbf{m} = \Gamma\left(c_1 - \langle \overrightarrow{otk}, \vec{b}\rangle \bmod q\right) = \Gamma(c_1 - \langle \overrightarrow{otk}, \vec{a}\rangle s + \tau.\langle \overrightarrow{otk}, \vec{e}\rangle)$$
$$= \Gamma(c_1 - c_0.s + \tau.\langle \overrightarrow{otk}, \vec{e}\rangle) = \Gamma(\Delta.\mathbf{m} + \tau.\partial + \tau.\langle \overrightarrow{otk}, \vec{e}\rangle)$$

Equality holds if

$$\|\partial + \langle \overrightarrow{otk}, \vec{e}\rangle\|_\infty < q/2p \Leftrightarrow \|\partial\|_\infty < q/2p + \|\overrightarrow{otk}\|_\infty.\|\vec{e}\|_\infty$$
$$\Leftrightarrow \|\partial\| < q/2p + s.B$$

The security of the resulting schema follows if any pair $(\overrightarrow{pk}, \vec{c})$ is still indistinguishable from random for any polynomial-time algorithm, as it is the underlying schema, and if the tuple $(\overrightarrow{pk}, \overrightarrow{otk}, \vec{c}\,')$ is indistinguishable from random for any polynomial-time algorithm for any $\vec{c}\,' \neq \vec{c}$. Both indistinguishability may be proven by hybrid arguments that we sketch below.

The public key $\overrightarrow{pk}$ comprises k+2 polynomial vectors, $[\mathbf{a}_i, \mathbf{b}_i]_{i=0}^{k+2}$, where $\mathbf{a}_0 = 1$, $\mathbf{a}_1 = \mathbf{a}' \leftarrow \mathcal{U}_{R_q}$ is a uniform random polynomial, and $\mathbf{a}_j$ for $1 < j < k$ are polynomials indinstinguishable from random. Each public key vector $[\mathbf{a}_i, \mathbf{a}_i.\mathbf{s} + \mathbf{e}_i]$, $i \neq 0$ is also an RLWE sample and may be replaced by random instances. The first column of the public key is given by $[1, \mathbf{s} + \mathbf{e}_0]$, where $\mathbf{s}, \mathbf{e}_0 \leftarrow \chi^n$ are two random Gaussian values taken from the same distribution, thus revealing nothing. Finally, by the indistinguishability of the underlying schema, we find that $(\overrightarrow{pk}, \vec{c})$ is indistinguishable from random for a polynomial-time algorithm with an advantage greater than ϵ.

The same strategy may be followed for the tuple $(\overrightarrow{pk}, \overrightarrow{otk}, \vec{c}\,')$. As before, $\overrightarrow{pk}$ may be replaced by a random polynomial vector, as, by construction, $\overrightarrow{otk}$ reveals nothing about $\overrightarrow{mk}$[42]. If it were possible to devise a distinguisher algorithm $\mathcal{D}$ that given $\overrightarrow{pk}$ and $\overrightarrow{otk}$ was able to distinguish $\vec{c}\,' = [\mathbf{c}_0', \mathbf{c}_1']^T$, where $\mathbf{c}_0' = \langle \vec{a}, \vec{u}\,'\rangle + \mathbf{e}_1$, $\mathbf{c}_1' = \langle \vec{b}, \vec{u}\,'\rangle + \mathbf{e}_2 + \Delta.\mathbf{m}'$, from $\vec{c}\,'' = [\mathbf{x}, \mathbf{y}]^T$, $\mathbf{x}, \mathbf{y} \leftarrow \mathcal{U}_{R_q}$ then it would be possible to build a distinguisher $\mathcal{D}'$ that given $\overrightarrow{pk}$ was able to distinguish $\vec{c}\,' = [\mathbf{c}_0', \mathbf{c}_1']^T$ from $\vec{c}\,'' = [\mathbf{x}, \mathbf{y}]^T$, by generating a random $\overrightarrow{otk} \leftarrow \mathcal{D}_{\Lambda,\vec{0},\sigma}$, creating a ciphertext $\vec{c} = [\langle \vec{a}, \overrightarrow{otk}\rangle, \mathrm{z}]$ and feeding $\mathcal{D}$.

5.3 Efficiency

Although the key generation and encryption algorithms in the proposed encryption scheme share a structure similar to the underlying BFV schema, they demonstrate reduced efficiency. This efficiency decrease is primarily attributed to a significant increase in the size of the public key, which becomes $k+2$ times larger. To improve operational efficiency, it is crucial to set the value of k to the smallest feasible value, denoted $k = \lceil \log_b q \rceil$.

However, this optimization faces limitations due to the "Double CRT" representation constraint, which requires k to satisfy the inequality $k \geq 2l_q + 2$. Here, l_q represents the count of pairwise coprime values q_i that make up the modulo $q = \prod_i^l q_i$, restricting the reduction of k beyond a certain threshold.

The proposed solution has been implemented in PALISADE[*], and performance results (Table 2) confirm that except for "Key Generation" and "Encryption", other BFV operations do not show a decrease in performance, i.e., SlowDow $= \frac{T_{\mathsf{OTK}} - T_{\mathsf{BFV}}}{T_{\mathsf{BFV}}} = 0$. "Key Generation" is 4.9 times slower due to the construction of the special public key, and "Encryption" is 2.3 times slower, mainly due to the larger size of the public key. The most significant drawback is the generation of $\overrightarrow{otk}$, which takes almost 18 times the elapsed time of a multiplication.

Table 2: OTK homomorphic schema performance

	OTK (μs)		BFV (μs)		Slow
	T_{OTK}	σ	T_{BFV}	σ	Down
KeyGen	12855	90	2615	26	4.9
MultKeyGen	4776	21	4794	73	0
AutomorphKeyGen	4789	36	4778	46	0
Encryption	7682	45	3407	28	2.3
Decryption	401	4	398	9	0
Add	59	6	56	4	0
AddInPlace	32	3	29	9	0
MultNoRelin	4133	241	4144	188	0
MultRelin	5456	52	5692	270	0
Automorph	755	19	760	17	0
OTKKeyGen	96569	1167	-	-	-
OTKDecrypt	616	70	-	-	-

5.4 Usage

OTK soundness requires that the holder's public key be of the form $\overrightarrow{pk} = [\vec{a}, \mathrm{s}.\vec{a} + \tau.\vec{e}]$ and that the error rate of the encryption is below some bound ∂. The former is ensured by having all the public keys of the holder signed by the key escrow service, and the latter can be checked by having the holder provide an OTK to decrypt the multiplication of the freshly encrypted message by an encryption of zero, with a large error, provided by the verifier. The zero encryption error should be such as to emulate the encryption error at level $l-1$, where l is the multiplication level of the function being evaluated. If both requirements are met, the holder may generate the encryption of the message in any way (including randomly) provided that the decryption with the secret key s matches the message she wants to prove.

6 HOMOMORPHIC COMPARISON

The primitive OTK is intended to decrypt the results of homomorphically evaluated functions (HEFs) that involve arithmetic operations, including comparisons such as equalities and inequalities. For example, in scenarios such as determining whether someone is under 18 years of age or comparing the country of residence with each European Union (EU) member state, since EU addresses usually do not explicitly contain "EU"or any distinguishing indicator, homomorphic comparisons (HC) pose a significant challenge. Bit-wise Homomorphic Encryption (HE) schemes, such as FHEW and TFHE,

[*]https://github.com/ribeirocn/palisade-sfdk

offer the fastest HC, with comparisons for two n-bit integers done in $\approx 170.n$ microseconds[16, 17].

However, when working with word-based encryption schemes (e.g., BGV, BFV, CKKS), HC performance is not as favourable. Efforts have been made to efficiently implement equality (EQ_S) and inequality (LT_S) operations within these encryption frameworks.

$$LT_S(a,b) = \begin{cases} 1, & \text{if } a < b; \\ 0, & \text{if } a \geq b, \end{cases} \qquad EQ_S(a,b) = \begin{cases} 1, & \text{if } a = b; \\ 0, & \text{if } a \neq b. \end{cases}$$

The implementation of the equality function is straightforward using Fermat's Little Theorem to implement the

$$\text{IsZero}(a-b) = EQ_S(a,b) = 1 - (a-b)^{q-1} \bmod q \tag{6}$$

function, often used in practice, although its usage can be challenging due to its sequential multiplications, particularly with large modulus values (q).

Kim et al. [30] introduced an optimization technique. When $a, b \in S = \mathbb{F}_{q^d}$ and $d > 1$, they use the multiplicative deep-free Frobenius automorphism operation $x \mapsto x^q$. This operation reduces the depth of the equality circuit $EQ(a,b) = 1 - (a-b)^{q^d-1}$ from $\lceil d \log_2(q) \rceil$ to $\lceil \log_2(d) \rceil + \lceil \log_2(q-1) \rceil$.

Iliashenko and Zucca [28] follow the bit comparison strategy, but instead of bits they split the message into digits that they encode in different slots of plaintext. They use a univariate polynomial to interpolate a function that calculates the sign of the difference between two values Tan et al. [48] went a bit further by embedding each plaintext in a ciphertext slot, encoding different digits of the plaintext into subfields of $\mathbb{F}_{q^d}$ in each slot. They compared each digit and processed the results using a lexicographic circuit.

One common optimization technique is to use SIMD techniques from certain homomorphic schemes such as BGV and BFV. This involves batching several plaintexts into a single ciphertext, allowing multiple comparisons to be performed simultaneously, which is advantageous when dealing with multiple comparisons over a dataset. Both Tan et al. and Iliashenko and Zucca's solutions use this technique to amortize the cost of each individual comparison.

However, in the "Anonymous Credentials" use case, only a single comparison is needed to decide access to an information service, and the values to compare often result from previous calculations involving additions and multiplications, which do not align with the digit decomposition in the proposed encoding. Without digit decomposition, a single comparison of a 15-bit number takes significantly longer ($\sim$ 4 m 38 s) with the Iliashenko and Zucca solution compared to the proposed solution (10s).

6.1 Private Set Membership

Our approach also takes advantage of the SIMD, but we aim to optimize a single comparison rather than amortizing the cost over several comparisons. Our comparison primitive is called "Private Set Membership" (PSM). The $\overline{\textbf{PSM}}$ primitive takes an encrypted value x and a set of values S and returns an encryption of 1 if $x \in S$ and an encryption of 0 otherwise.

$$\begin{gathered} \overline{\textbf{PSM}} : R_q^2, R_q^r \to R_q^2 \\ (\mathcal{E}(x), \vec{s}) \to \{\mathcal{E}(0), \mathcal{E}(1)\} \end{gathered} \tag{7}$$

$\overline{\textbf{PSM}}$ takes two arguments, an encrypted value $\mathcal{E}(x)$ and a set S of plaintext values, and outputs $\mathcal{E}(1)$ if $x \in S$, and $\mathcal{E}(0)$ otherwise. Encryption $\mathcal{E}(x)$ is a value of the ring R_q^2, as defined by the encryption schemas BGV or BFV, while the plaintext values $v = |S|$ in the set S are packed in a vector $\vec{s} \in R_q^r$, such that $r = \left\lceil \frac{v}{l_\Phi} \right\rceil$ where l_Φ is the number of SIMD slots of the ring R_q. The goal of such encoding is to allow parallel comparison of the encrypted value $\mathcal{E}(x)$ with every slot value of each element $\mathbf{s}_i \in R_q$ of $\vec{s}$.

Algorithm 1 Simplified PSM Algorithm

```
procedure $\overline{\textbf{PSM}}(\mathcal{E}(x), \overrightarrow{s})$
    $d \leftarrow \max |\mathbf{s}_i|$
    $\mathcal{E}(\mathbf{x}) \leftarrow$ SPREAD($\mathcal{E}(x), d$)                ▷ Copy $x$ to every slot
    $\mathcal{E}(\mathbf{o}) \leftarrow$ SUBPROD($\mathcal{E}(x), \overrightarrow{s}$)        ▷ Calculate: $\prod_{i=0}^{r-1} \mathcal{E}(\boldsymbol{x}) - \mathbf{s}_i$
    $\mathcal{E}(\mathbf{o}) \leftarrow$ ISZERO($\mathcal{E}(o)$)       ▷ Map every slot to $\mathcal{E}(0)$ or $\mathcal{E}(1)$
    $\mathcal{E}(\mathbf{o}) \leftarrow$ SUMALL($\mathcal{E}(o), d$)                ▷ Add all slots
    return $\mathcal{E}(\mathbf{o})$
end procedure
```

The simplified $\overline{\textbf{PSM}}$ algorithm (Algorithm 1) has four simple homomorphic steps. The first step copies the encrypted value $\mathcal{E}(x)$ in the first slot into all other $l_\Phi - 1$ slots. The result is an encryption of a packed vector $\mathcal{E}(\mathbf{x}) = \mathcal{E}(\{x, \ldots, x\}) \in R_q$ with l_Φ copies of x. The second step executes a SIMD subtraction of $\mathcal{E}(\mathbf{x})$ from every element of $\vec{s}$ and calculates the SIMD product of the result $\mathcal{E}(\mathbf{o}) = \prod_{i=0}^{r}(\mathcal{E}(\mathbf{x}) - \mathbf{s}_i) \bmod q$. The result $\mathcal{E}(\mathbf{o}) \in R_q^2$ is a packed encryption of a vector $\vec{o}$, with at most one zero element, the one that matches the searched value. Notice that the last element of the vector $\vec{s}$ may not have all slots filled, thus requiring padding, which is done by ciphertext stealing the result of the previous product. The third step computes the IsZero(a) (eq. (6)) function on every slot at the same time. Finally, the fourth step adds all the slots and returns the result. In general, the algorithm computes the function $\overline{\textbf{PSM}}(x, s)$ homomorphically (eq. (8)), taking advantage of the SIMD techniques as much as possible.

$$\overline{\textbf{PSM}}(x, \vec{s}) = \sum_{j=0}^{l-1} 1 - \left(\prod_{i}^{r} (x - s_{i,j}) \right) \tag{8}$$

The third step is the most expensive since it requires $\log(q-1)$ multiplications to calculate the exponentiation by iterative squaring. The first step is computed using an algorithm similar to iterative squaring, but instead of squaring the result at each iteration, it uses the automorphisms of the ring setting R_q to rotate and add $\log(l_\Phi)$ times (Appendix A). The second step requires $r-1$ multiplications. However, for large ring dimensions, the number of available slots l is often larger than $|S|$, so $r = \lceil |S|/l \rceil = 1$ and no multiplications are required. The last step also does not require multiplications. It adds all the slots and leaves the result in the first slot. It follows a strategy similar to the SPREAD function, but in the reverse direction (Appendix A). As such, it also requires $\log(l_\Phi)$ rotations and additions.

6.2 Applications and Performance

Within the "Anonymous Credential" setting, the $\overline{\textbf{PSM}}$ primitive can be used to check the following:

- if an address is within the EU, by checking if the name of the country in the address is one of the member states without revealing the address;
- if the OID of the credential is listed in a CRL without revealing the OID;
- if someone is under 18 years old at a specific date without revealing her age;
- if some encrypted value is less than another encrypted value.

However, the usage strategies differ. The most straightforward is to check the OID of the credential. It only requires packing the CRL list in the vector $\vec{s}$ and running $\overline{\textbf{PSM}}(\mathcal{E}(oid), \vec{s})$. For some RLWE settings, it is possible to pack in $\vec{s}$ more than 100K OIDs at the cost of only 2 or 3 additional multiplications (second step of algorithm 1).

6.2.1 Integer Comparison. Checking if someone is under 18 years of age or if an encrypted value a is less than another encrypted value b follows a similar strategy. We compute the difference between both values $a - b$ or between the birthdate and the current date and check if the result is in a specific set. If the result is in a set with all possible negative values ($\mathbb{F}_q^- = [-(q-1)/2, -1]$), or with all possible ages (in days) between 0 and 18 years ($S = \{0, \ldots, \sim 6570\}$) then $a < b$, or someone is under 18.

Iliashenko and Zucca [28] made available an implementation of their solution*. For comparison purposes, we have extended their implementation with our one† and tested both in the conditions required by the current setting:

- Just one comparison, thus time is not amortised over multiple comparisons;
- No digit decomposition, thus each value is contained in a single slot, and each slot may contain a single number in $\mathbb{F}_q^+$.

Table 3 contains the parameters chosen for various domain sizes with a logarithmic scale $\log_2 |S|$, for both the PSM and the univariate schemas of [28].

Table 3: Integer Comparison Settings

$\log_2 \lvert S\rvert$	p	m	l_Φ	λ	$\log_2 q$	Univ	PSM
6	131	25743	4290	161	260	4.08 s	2.75 s
9	1031	24247	898	145	400	13.29 s	4.62 s
10	2053	35443	1518	189[a]	450[b]	30.75 s	8.97 s
12	8209	39283	6480	171[c]	560[d]	77.56 s	13.00 s
15	65537	65536	32768	104	730	256.50 s	9.59 s

[a,b,c,d] differ slightly in PSM version: [a]193, [b]440, [c]175, [d]550.

We have run experiments with message bit sizes of 6 to 15. For sizes above 15 bits, the time spent initializing the cryptography context is too large. The 15-bit setting is itself a particular case. Given that the order d of q modulo m is 1, the number of available slots $l_\Phi = n/d$ is given by the size of the ring $n = |R_q| = \phi(m)$. Such a large l_Φ simplifies the comparison greatly. In particular, all negative values $\mathbb{F}_q^-$ fit into a single vector $\vec{s}$ and may be compared simultaneously.

From the results depicted in Fig. 2a, we may conclude that, for this specific purpose, the PSM solution is always faster than the univariate solution, which was expected from the number of multiplications that both need to perform. For the specific setting, the number of multiplications performed by the Univariate schema is

$$U(p) \sim \sqrt{2p-4} + 3(\log_2 2p - 4)/2 + 2$$

while the number of multiplications in PSM is

$$P(p) = (p-1)/2l_\Phi + \log_2 p - 1$$

It is easy to show that $U(p) > P(p)$ for any $l_\Phi > 0$ and $p > 2$.

6.2.2 String Comparison. $\overline{\textbf{PSM}}$ handles string comparison in two ways, resulting from different packaging strategies. Both strategies require minor changes to the $\overline{\textbf{PSM}}$ algorithm. The first strategy, called MultiSlot, is to pack each string of size e over e plaintext slots, thus setting the number of strings encoded in each plaintext at most $b = \lfloor l_\Phi / e \rfloor$. The second strategy, dubbed UniSlot, is more compact, but slightly less efficient. It takes advantage of the plaintext ring structure $\mathbb{F}_{p^\delta}$, where δ is the order of p modulo m. When $\delta > 1$, it is possible to pack several characters in a single slot.

The first strategy requires changing the function SumAll because the slots containing the result of the slot-wise comparison of the string with each copy of the pattern must first be multiplied by each other and then only added together, thus requiring additional $\log_2 e$ multiplications. The second strategy requires changing the IsZero function to implement Fermat's Little Theorem on extension fields.

$$\text{IsZero}(a) = 1 - a^{p^{\delta-1}} \bmod p \qquad (9)$$

Kim et al. [30] showed that this equation could be implemented with a multiplicative depth of $\lceil \log_2 \delta \rceil + \lceil \log_2 p - 1 \rceil$ by taking advantage of the homomorphic Frobenius automorphism $x \mapsto x^p$. Therefore, when $\delta = e$, the multiplicative depth of both methods is equal. However, this second strategy also requires $\delta - 2$ automorphisms that do not consume depth but do consume time.

Both alternatives impose limits on the search space. The UniSlot strategy forces the size of the pattern string to be no larger than p^δ bits, while the MultiSlot strategy imposes a soft limit on the search space $|S|$. The limit of the UniSlot strategy may be eliminated by combining both strategies, thus packing each string in $e.\delta$ slots. To eliminate the limit of the MultiSlot strategy, the SubProd call in Alg. 1, must be replaced by a call to a more complex SubMapAdd function (Appendix A).

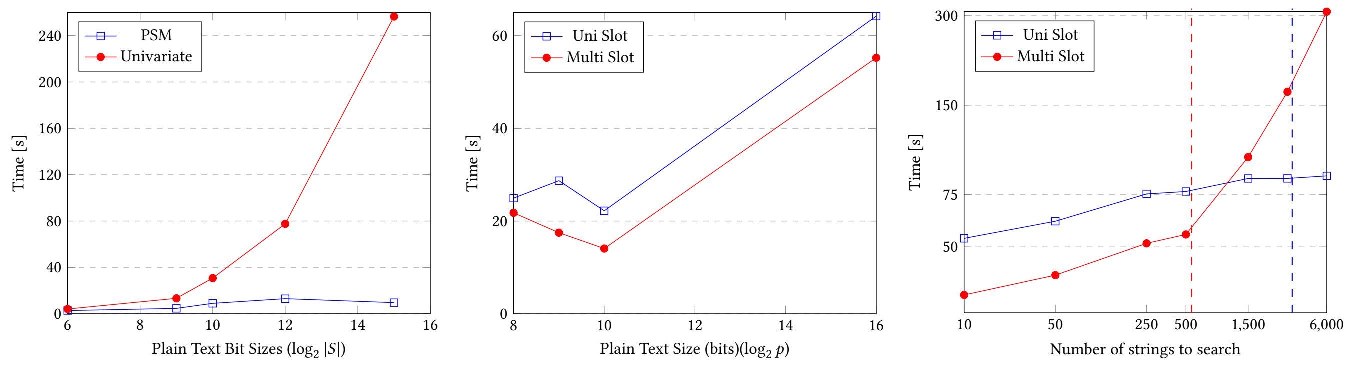
Tab. 4 compares both strategies for different plaintext modulus p. The maximum string size v in bytes is set to $v = d. \lfloor \log_2 p \rfloor / 8$ for the UniSlot strategy and $v = e. \lfloor \log_2 p \rfloor / 8$, with $e = 16$, for the MultiSlot strategy. The biggest difference between the two approaches is the number of strings that may be compared simultaneously. The UniSlot strategy can, on average, compare ten times more strings simultaneously (ss_{max}).

Regarding efficiency, the UniSlot version is slightly less efficient (Fig. 2b) due to additional $d - 2$ automorphisms.

However, if the search space $|S| > ss_{max}$, then MultiSlot is much less efficient due to the SubMapAdd algorithm. Fig. 2c depicts the time required to search for different set sizes $|S|$, for 16 character

*https://github.com/iliailia/comparison-circuit-over-fq

†https://github.com/ribeirocn/comparison-circuit-over-fq



(a) Efficiency of Integer Comparison. **(b) PSM efficiency with plaintext size.** **(c) PSM efficiency with set size.**

Figure 2: PSM efficiency

Table 4: String Equality Settings

p	λ	UniSlot			MultiSlot		
		v	ss	T(s)	v	ss	T(s)
257	163	21	1448	24.95	16	90	21.78
521	141[a]	24	1560	28.71	18	159	17.49
1031	133	21	1904	22.22	20	119	14.07
65537	164[b]	36	3480	64.22	32	504	55.24

[a,b] differ slightly in MultiSlot version: [a]135, [b]122.

UTF-16 strings (last line of Tab. 4). Notice that the slope of the MultiSlot curve increases when the set to search is larger than $ss_{max} = 504$, but that does not happen with UniSlot when the search set is larger than $ss_{max} = 3480$.

7 PROOF OF OWNERSHIP

The proof of ownership of a credential is no different from the proof of any other attribute. The holder provides the issuer with a committed value, and the issuer includes it as any other attribute. To prove ownership, the holder opens the commitment but does it homomorphically, not revealing the opener or the commitment.

The commitment can be any commitment with a linear opening, such as, for instance, the Ajtai commitment [1] adapted to the ring setting. In such a commitment scheme, one commits to a secret vector $\vec{s} \in R^l_{d,p}$ with a randomness $\vec{r} \in R^l_{d,p}$, where both $\|\vec{s}\|$ and $\|\vec{r}\|$ are small, for some public random values $\vec{a} \leftarrow R^l_{d,p}$ and $\vec{b} \leftarrow R^l_{d,p}$, by computing the commitment

$$\begin{aligned}\mathbf{Comm}(\vec{s},\vec{r})\colon R^{2l}_{d,p} &\to R_{d,p}\\ \langle\vec{a},\vec{s}\rangle + \langle\vec{b},\vec{r}\rangle &\mapsto \mathbf{c}\end{aligned} \tag{10}$$

and sending it to the issuer to be signed, together with the issuers' attributes (cf. section 3).

Opening the Ajtai commitment is a matter of revealing the committed value $\vec{s}$ and the randomness $\vec{r}$, and check

$$\mathbf{OpenCom}(\mathbf{c},\vec{s},\vec{r})\colon R^{2l+1}_{d,p} \to \{0,1\}$$

$$\langle\vec{a},\vec{s}\rangle + \langle\vec{b},\vec{r}\rangle \overset{?}{=} \mathbf{c} \tag{11a}$$

$$\|\vec{s}\| \text{ and } \|\vec{r}\| \text{ are small} \tag{11b}$$

Implementing the commitment opening homomorphically over the presentation ciphertext $\boldsymbol{prt} = \mathcal{E}(\boldsymbol{m}) \in R_q$

$$\overline{\mathbf{OpenCom}}(\boldsymbol{prt})\colon R_q \to \mathcal{E}(\{0,1\}) \tag{12}$$

can be done efficiently if the presentation $\boldsymbol{prt}$ is the encryption of the canonical embedding of the different elements of the presentation

$$\boldsymbol{prt} = \mathcal{E}\left(\{\vec{0}, sn, \vec{\sigma}(\mathbf{c}), \vec{\mu}, \vec{\sigma}(\vec{\varsigma}), \vec{\sigma}(\vec{\boldsymbol{o}})\}\right) \tag{13}$$

where $\mathbf{c}$ is the commitment, $\vec{\mu} = \{\mu_0, \ldots \mu_p\}$ is the set of attributes, $\vec{\varsigma}$ is the credential signature (cf. section 8), sn is the credential serial number, and $\vec{\boldsymbol{o}} = (\vec{s},\vec{r}) \in R^{2l}_{d,p}$ is the commitment opener. Using standard automorphism operations available in BGV and BFV, the verifier can verifier extract $\mathcal{E}(\vec{s})$, $\mathcal{E}(\vec{r})$ and $\mathcal{E}(\mathbf{c})$ from the presentation $\boldsymbol{prt} = \mathcal{E}(\boldsymbol{m})$ and verify homomorphically eqs. (11a) and (11b).

The eq. (11a) can be checked efficiently homomorphically by computing

$$\overline{\mathbf{Sum}}_{2l,d}\left((\vec{a},\vec{b}) \odot \mathcal{E}(\vec{o})\right) \ominus \mathcal{E}(\mathbf{c}) \tag{14}$$

The result will be an encryption of 0 if the commitment was correctly opened or something else otherwise. The operators $\odot$ and $\ominus$ denote the element-wise homomorphic multiplication and subtraction, respectively, and $\overline{\mathbf{Sum}}_{m,n}(\mathcal{E}(\vec{a}))$ is a homomorphic function that takes m groups of n slots of each encrypted polynomial $\mathcal{E}(\mathbf{a}_k)$ and returns an encrypted polynomial $\mathcal{E}(\mathbf{b}_k)$ with the first n slots having the group-wise homomorphic addition, and the remaining slots having zero (vide eq. (15)).

$$\begin{aligned}\mathcal{E}\left(\vec{b}\right) &= \overline{\mathbf{Sum}}_{m,n}(\mathcal{E}(\vec{a}))\\ \vec{\sigma}(\mathbf{b}_k)_i &= \begin{cases}\sum_{j=0}^{m-1} \vec{\sigma}(\mathbf{a}_k)_{i+j.n} & \text{if } 0 \le i < n\\ 0 & \text{otherwise}\end{cases}\end{aligned} \tag{15}$$

With this approach, checking eq. (11a) requires only one homomorphic multiplication, $1 + \log m$ additions, and $\log m$ automorphisms.

To simplify the process of verifying eq. (11b), we restrict $\vec{s}$ and $\vec{r}$ to have $\| \vec{\sigma}(\vec{s})\|_\infty = \| \vec{\sigma}(\vec{r})\|_\infty = 1$, by chosen two random vectors $\vec{s}, \vec{r} \leftarrow \{-1, 0, 1\}^{l.d}$ and canonical embedding them in $\vec{s}$ and $\vec{r}$, respectively. The verifier leverages the lemma 2 to check that every slot of the opener $\vec{\sigma}(\boldsymbol{o})_i \in \{-1, 0, 1\}$, and the norm inequalities[21] to ensure that the norm

$$\| \vec{o} \| \leq \sqrt{d.(l+1)}\|\boldsymbol{o}\|_\infty \leq \sqrt{d.(l+1)}\| \vec{\sigma}(\boldsymbol{o})\|_\infty \leq \sqrt{d.(l+1)}$$

The commitment is statistically hiding and computationally binding, provided that $RSIS_{d,p,l,d(l+1)}$ is hard. From eq. (2) we must set $d, l \geq \log p$ and p so that the bound

$$B = \sqrt{d.(l+1)} < \min(p, 2^{\sqrt{4d(l+1)\log p \log \delta}})$$

For reasons that will be evident in the next section, the plaintext modulus p can be set to a prime number $p \approx O(2^{16})$. Setting l to the minimum admissible value $l = 16$, we have $d \geq 1$ for a hermit value $\delta = 1.0043$. Choosing $d = 4$ ensures a large enough commitment (2^{64}) that is not easily guessable. Therefore, the commitment takes only four slots in the credential, and the opening $2.4.16 = 128$ slots in the encrypted presentation message $\mathcal{E}(\vec{\sigma}(m))$.

As usual, the commitment $\mathbf{c}$ is the holder's identifier on the issuer. The holder may be known by different commitments $\mathbf{c}' = \langle \vec{a}, \vec{s} \rangle + \langle \vec{b}, \vec{r}\,' \rangle$ to the same secret $\vec{s}$, in different issuers, by using different randomizations $\vec{r}\,'$. In this context, proving the ownership of two credentials from different issuers requires only extending the opener $\vec{o} = (\vec{s}, \vec{r}, \vec{r}\,')$ to include both randomizations.

8 HOMOMORPHIC VERIFICATION OF ENCRYPTED SIGNATURES

In an "Anonymous Credentials" setting, the credential holders should be able to present the same credential several times to the same verifier or different verifiers without revealing their identity or being traced. However, signatures are often immutable to prevent existential forgery attacks, and therefore easily traceable. The classical solution is to replace the signature with a new, fresh zero-knowledge proof that the holder knows the signature for the credential. However, quantum-resistant zero-knowledge solutions in quantum-resistant encrypted data, signed with quantum-resistant signatures, have some challenges[5]. Another approach is to encrypt the signature with a fresh new encryption each time a credential is presented. However, verifying an encrypted signature over encrypted data poses several challenges, with the primary one being the need for homomorphic computability in the verification process.

A signature solution proposed by Lyubashevki et al.[39] could potentially be used. However, it requires a large modulus ($q = O(2^{64})$), which is suboptimal for encryption. When the plaintext modulus is of order $O(2^{64})$, the ciphertext modulus must be of the order $O(2^{128})$, which is impractical for efficient encryption operations.

In the following subsection, we start by proposing a simple GPV-like signature [*], which is not as efficient as most existing lattice-based signatures but fits our homomorphic needs. Then we show how to implement the signature verification step homomorphically.

8.1 GPV-like Signature

The proposed schema is similar to the GPV schema [26], but the hash function is replaced by a collision-resistant lattice base hash function and a pseudorandom number generator. The GPV schema uses a trapdoor $\vec{r} \in R_p^k$ for a vector $\vec{a} \in R_p^{k+2}$ to sample a short vector $\vec{s}$ from a discrete Gaussian in a lattice $\Lambda^{\perp}_{H(\mu)}(\vec{a})$, such that $\langle \vec{a}, \vec{s} \rangle = H(\boldsymbol{\mu}) \bmod p$, where $H : R_q \rightarrow R_q$ is a polynomial hash function.

The proposed schema also samples a short vector $\vec{s}$ from a discrete Gaussian such that

$$\langle \vec{a}, \vec{s} \rangle = \langle \vec{b}, \vec{\mu} \rangle + \mathbf{v}_i \bmod p$$

where $\mathbf{v}_i$ is the i^{th} element of a pseudorandom number generator $\mathbf{PRGN}(seed) = [\mathbf{v}_1, \ldots, \mathbf{v}_N] \in R_q^N$ and $\vec{\mu} \in R_p^m$ is a short vector with signed data.

8.1.1 *Signature Definition.*

ParamGen$(1^\lambda) \rightarrow d, q, m, b, \sigma$: Generates the schema parameters $(d, q, m, b, \sigma) \in \mathbb{Z}^5$ for the specified security λ.

KeyGen$(d, q, m, b) \rightarrow \overrightarrow{pk}, \overrightarrow{sk}$: The key generation phase starts by generating an MP12 trapdoor $(\vec{a}, \vec{r}) \leftarrow \mathbf{TrapGen}(d, q, b, \sigma)$, with $k = \lceil \log_b q \rceil$, and d as the ring size. Then it samples $\vec{b} \leftarrow R_q^m$ and $seed \leftarrow \mathbb{Z}$ uniformly. It returns $\overrightarrow{sk} = \vec{r} \in R_q^k$ and $\overrightarrow{pk} = (\vec{a}, \vec{b}, seed) \in R_q^{k+2} \times R_q^m \times \mathbb{Z}$

Sign$(\vec{\mu}, \overrightarrow{sk}, \overrightarrow{pk}) \rightarrow \vec{\varsigma}$: The signature phase starts by checking that $\vec{\mu} \in R_q^m$ and $\| \vec{\mu} \| < \sigma_\mu$. It gets a one-time public key $\mathbf{v}_i \leftarrow \mathbf{PRGN}(seed)$, increments i, computes the syndrome $\mathbf{c} = \langle \vec{b}, \vec{\mu} \rangle + \mathbf{v}_i$, and samples a short vector $\vec{s} = \mathbf{SamplePre}(\mathbf{c}, \vec{a}, \vec{r})$. It returns $\vec{\varsigma} \leftarrow (\vec{s}, \mathbf{v}_i)$

Verify$(\vec{\mu}, \vec{\varsigma}, \overrightarrow{pk}) \rightarrow \{1, 0\}$: The signature verification phase checks that

$$\langle \vec{a}, \vec{s} \rangle = \langle \vec{b}, \vec{\mu} \rangle + \mathbf{v}_i \bmod p \tag{16}$$
$$\| \vec{s} \| < B_t \tag{17}$$
$$\| \vec{\mu} \| < B_\mu \tag{18}$$
$$\mathbf{v}_i \in \mathbf{PRGN}(seed) \tag{19}$$

Outputs 1 if all equations hold and 0 otherwise.

8.1.2 *Soundness and Security.* The signature schema is trivially correct by the definition of **TrapGen** since every signature generated by the signing procedure is verified by the verification procedure.

To prove security, we show that, if there is an adversary $\mathcal{A}$ that can generate a new valid signature $(\vec{\mu}, \vec{\varsigma})$ in an existential forgery game, we can build a simulator $\mathcal{B}$ that solves $\mathbf{RSIS}_{d,q,k+2,B_t}$, $\mathbf{RSIS}_{d,q,m,B_\mu}$ or $\mathbf{RSIS}_{d,q,m+k+2,\sqrt{B_\mu^2+B_t^2}}$ with nonnegligible probability (cf. Appendix B).

[*]The signature is straightforward and we are not sure that it has never been proposed before, but we could not find it.

8.1.3 Parameter Setting. Setting the signature parameters such that the signature is secure and yet easily encryptable is a challenge. Encrypting polynomials with large plaintext is not efficient; therefore, we choose p to be the smallest possible prime value that is still secure. From eq. (1) we know that the bound $B_t = \| \vec{s} \|$ on the trapdoor-generated vector $\vec{s} \in R_{d,p}^{k+2}$, with $k = \lceil \log_b p \rceil$, is given by

$$B_t = s\sqrt{d(k+2)} \tag{20}$$

and from eq. (2) that

$$B_t < \min(p, 2^{\sqrt{4d(k+2)\log p \log \delta}}) \tag{21}$$

For $\delta = 1.0043$ the minimum p that satisfies both equations is $p \sim O(2^{16})$, for $b = 2$ and $d = 64$.

In the previous section, we have defined the presentation message $\boldsymbol{prt}$ (eq. (13)) embedding the encryption of the signed data $\mathcal{E}(\boldsymbol{\mu}) = \mathcal{E}(\{sn, \vec{\sigma}(\mathbf{c}), \vec{\mu}\})$. However, this embedding is not possible because the signed data are embedded in a polynomial with a large norm $\boldsymbol{\mu}$, and the signature algorithm signs vectors of polynomials with a small norm $\vec{\boldsymbol{\mu}}$. Therefore, instead of embedding $\vec{\sigma}(\boldsymbol{\mu})$ we embed a b' decomposition vector $\vec{\sigma}(\vec{\boldsymbol{\mu}})$, such that $\vec{\sigma}(\boldsymbol{\mu}) = G.\vec{\sigma}(\vec{\boldsymbol{\mu}})$, where $G = \vec{\boldsymbol{g}} \otimes I$ is the so-called Gadget matrix and $\vec{\boldsymbol{g}} = [1, b', b'^2, \dots b'^{h-1}]$ the Gadget vector of polynomials of constant value.

8.2 Homomorphic Signature Verification

The purpose of this section is to describe the implementation of homomorphic signature verification.

$$\begin{aligned} \overline{\textsf{Verify}} : & R_q^2, R_q^{k+2} \times R_q^m \times \mathbb{Z} \to R_q^2 \\ & \boldsymbol{prt}, \overrightarrow{\boldsymbol{pk}} \to \{\mathcal{E}(\vec{0}), \mathcal{E}(\vec{x})\} \end{aligned} \tag{22}$$

The primitive $\overline{\textsf{Verify}}$ takes two arguments, the encrypted presentation $\boldsymbol{prt} \in R_q^2$ and the issuer's public key $\overrightarrow{\boldsymbol{pk}} \in R_q^{k+2} \times R_q^m \times \mathbb{Z}$ and outputs and encryption of zero if the verification is successful or something else if the verification is not successful.

The homomorphic verification of the signature is valid if the homomorphic verification of each of the eqs. (16) to (19) is valid. whereas eq. (16) is trivially verified by computing

$$\overline{\textsf{Sum}}_{k+2,d}\left(\mathcal{E}(\vec{s}) \odot \vec{a}\right) - \overline{\textsf{Sum}}_{h,d}\left(\mathcal{E}(\vec{\boldsymbol{\mu}}) \odot \vec{b}\right) - \mathcal{E}(\mathbf{v}_i) \tag{23}$$

and testing for zero, the others are slightly more involved. The next two subsections will address the verification of the norm-bound equations (eqs. (17) and (18)) and the **PRGN** membership test eq. (19), respectively.

8.2.1 Bounded Norm Check. The homomorphic verification of eq. (18) for some bound $B_\mu = \min(p, 2^{\sqrt{4d.l \log p \log \delta}})$ that makes $\textsf{RSIS}_{d,q,m,B_\mu}$ difficult can be achieved by checking that $\mathcal{E}(\vec{\boldsymbol{\mu}})$ encrypts a polynomial vector containing binary slot values (i.e., $b' = 2$). In fact, taking advantage of the norm inequalities [21], if $\vec{\sigma}(\vec{\boldsymbol{\mu}}) \in \{0, 1\}$ then

$$\| \vec{\boldsymbol{\mu}} \| \leq \sqrt{h.d} \| \vec{\boldsymbol{\mu}} \|_\infty \leq \sqrt{h.d} \| \vec{\sigma}(\vec{\boldsymbol{\mu}}) \|_\infty = \sqrt{h.d} \ll B_\mu$$

The homomorphic multiplication of two encrypted values is equivalent to the elementwise modular multiplication of their canonical vectors, therefore by lemma 1, $\mathcal{E}(\vec{\boldsymbol{\mu}})$ encrypts a polynomial vector containing binary slot values if and only if

$$\mathcal{E}(\vec{\boldsymbol{\mu}}) \odot (\vec{1} - \mathcal{E}(\vec{\boldsymbol{\mu}})) \overset{?}{=} \mathcal{E}(\vec{0}) \tag{24}$$

The homomorphic verification of eq. (17) is slightly more involved. We take advantage that any vector $\vec{s}$ sampled by **TrapSampl** has norm $\| \vec{s} \|_\infty < s$ w.h.p.*, and that $\| \vec{\sigma}(\vec{s}) \|_\infty \leq \sqrt{d} \| \vec{s} \|_\infty$ [21], to infer the maximum amount of bits $w = \left\lceil \log s\sqrt{d} \right\rceil$ needed to encode the modulo of each element of $\vec{\sigma}(\vec{s})$, and replace the embedding of $\vec{s}$ in the encrypted presentation $\boldsymbol{prt}$ by embedding a special decomposition

$$\vec{\sigma}(\vec{s})^* = (\vec{\sigma}(\vec{s})^-, \vec{\sigma}(\vec{s})^+)$$

The key intuition is that $\vec{\sigma}(\vec{s})^- \in \{1, -1\}^{d(k+2)}$ encodes the signal of every element of $\vec{\sigma}(\vec{s})$ and $\vec{\sigma}(\vec{s})^+ \in \{0, 1\}^{d.w(k+2)}$ encodes a binary representation of the modulo of every element in $\vec{\sigma}(\vec{s})$, such that it is possible to reconstruct $\vec{\sigma}(\vec{s})$ by the linear operation

$$\vec{\sigma}(\vec{s}) = \vec{\sigma}(\vec{s})^- \odot (\mathbf{G}.\vec{\sigma}(\vec{s})^+) \tag{25}$$

Therefore, the homomorphic verification of the norm bound of eq. (17) subsumes to the verification that each each element of $\vec{\sigma}(\vec{s})^+$ is 0 or 1, and that each element in $\vec{\sigma}(\vec{s})^-$ is either -1 or 1, by using Lemma 1 and Lemma 2, respectively, homomorphically

$$\begin{aligned} \mathcal{E}\left(\vec{\sigma}(\vec{s})^+\right).(\vec{1} - \mathcal{E}\left(\vec{\sigma}(\vec{s})^+\right)) &\overset{?}{=} \mathcal{E}(\vec{0}) \\ (\vec{1} + \mathcal{E}\left(\vec{\sigma}(\vec{s})^-\right)).(\vec{1} - \mathcal{E}\left(\vec{\sigma}(\vec{s})^-\right)) &\overset{?}{=} \mathcal{E}(\vec{0}) \end{aligned} \tag{26}$$

Proving these properties proves that $\| \vec{s} \|_\infty < 2^w$, which is usually enough (cf. Appendix C).

8.2.2 Homomorphic PRGN Computation. Homomorfically checking that eq. (19) holds can be done using the $\overline{\textsf{PSM}}$ operation (Section 6.1) or the one-out-of-many proof proposed in [40]. However, none of them scales well. $\overline{\textsf{PSM}}$ may only be used efficiently for small sets and small string sizes (Figure 2b), while the validation of the one-out-of-many proof from [40] scales linearly with the number of maximum signatures allowed for each key. The proposed schema scales logarithmically with the maximum number of signatures allowed, making this limit irrelevant.

The main insight of our approach is to implement **PRGN**(*seed*) using a homomorphically computable pseudo-random function **PRF**(*seed*, i) and recompute

$$\mathcal{E}(\mathbf{v}_i) = \overline{\textbf{PRF}}(seed, \mathcal{E}(i)) \tag{27}$$

homomorphically. If the maximum number of signatures $N = 2^m$ is set to be the biggest number that can be encoded in $\texttt{Enc}(i)$ then the computed $\mathcal{E}(\vec{v}_i) \in \overline{\textbf{PRGN}}(seed)$.

The variant RLWE of the **PRF** proposed by Boneh et al. [3] almost fits our needs. It may be described as

$$\textbf{PRF}(\text{seed}, v) = \langle \lfloor \vec{f}^* \rceil_\beta, \vec{g} \rangle \tag{28}$$

$$\vec{f}^* = \mathbf{s}.\left(\prod_{j=0}^{m-1} \mathbf{P}_0.(1 - v_j) + \mathbf{P}_1.v_j\right) \tag{29}$$

where

*If that is not the case the signature issuer should sample the vector again.

$\vec{g}$	$= [1, \beta, \ldots, \beta^{\lceil \log_\beta p \rceil}]$	is the gadget vector
$\mathbf{s}$	$= Enc(seed) \in R_{d,p}$	is the ring encoding of the seed
$\vec{v}$	$= g^{-1}(v)$	is the bit decomposition of v
$\mathbf{P}_i$	$\in R_{d,p}^{\alpha\times\alpha}$	are random binary matrixes
α	$= \lceil \log_2 p \rceil$	is the number of bits of p
$\lfloor x \rceil_\beta$	denotes rounding an element $x \in \mathbb{Z}_p$ to $\mathbb{Z}_\beta$, $\beta < p$, by multiplying it by (β/p) and rounding the result	

Homomorphically computing an encryption of $\vec{f}^*$ (eq. (29)) from $\mathcal{E}(\vec{v})$ is relatively straightforward. However, performing the rounding operation in eq. (28) homomorphically presents a greater challenge. This challenge closely resembles the rounding step found in BFV or BGV bootstrapping [25], and similar techniques can be applied. However, the substantial multiplication depth required for both the rounding operation in eq. (28) and the computation of eq. (29) requires the use of large and inefficient rings.

Instead, we provide the verifier with $\mathcal{E}(\vec{x}) = \mathcal{E}\left(\lfloor \vec{f}^* \rceil_\beta\right)$, from which the verifier can compute $\mathcal{E}(\mathbf{v}_i)$ from eq. (28), and leverage the proposal from [34] to check that $\mathcal{E}(\vec{x}) = \mathcal{E}\left(\lfloor \vec{f}^* \rceil_\beta \bmod p\right)$, by homomorphically checking that

$$p.\mathcal{E}(\vec{x}) = \beta.\mathcal{E}\left(\vec{f}^*\right) + \mathcal{E}_\beta(\vec{z}) \quad (\mathrm{mod}\,\beta.p) \tag{30}$$

$$\|\vec{\sigma}(\vec{x})\|_\infty = \beta \quad \|\vec{\sigma}(\vec{f}^*)\|_\infty < p \quad \|\vec{\sigma}(\vec{z})\|_\infty < p \tag{31}$$

where $\mathcal{E}_\beta(\vec{z})$ is an ecryption modulo $\beta.q$ of a polynomial $\vec{z} \in R^\alpha_{d,\beta.p}$, with $\|\vec{z}\|_\infty < p$ that makes eq. (30) true.

Checking eq. (30) homomorphically is not straightforward because it must be done modulo $\beta.p$, and while $\mathcal{E}_\beta(\vec{z})$ encrypts a plaintext modulo $\beta.p$, $\mathcal{E}(\vec{x})$ and $\mathcal{E}\left(\vec{f}^*\right)$ encrypt plaintext modulo p. However, it can be shown (Appendix D) that for BFV encryptions, the **OTKDecrypt** algorithm over plaintext modulo $\beta.p$ and ciphertext modulo $p.q$ (denoted **OTKDecrypt**$_\beta$) correctly decrypts eq. (30) provided that $\beta \ll p$.

Checking eq. (30) homomorphically is a question of computing

$$\mathcal{E}(\vec{y}) = \beta.\mathcal{E}\left(\vec{f}^*\right) + \mathcal{E}_\beta(\vec{z}) - p.\mathcal{E}(\vec{x}) \quad (\mathrm{mod}\,\beta.p)$$

and testing

$$\mathsf{OTKDecrypt}_\beta(\overrightarrow{\boldsymbol{pk}}, \overrightarrow{\boldsymbol{otk}}, \mathcal{E}(\boldsymbol{y_i})) \stackrel{?}{=} \mathbf{0} \qquad \forall_{0\le i<\alpha} \tag{32}$$

Finally, the verifier must check the norms in eq. (31). The first one can be check easily if $\beta = 2$, by Lemma 1

$$\mathcal{E}(\vec{x}).(1 - \mathcal{E}(\vec{x})) \stackrel{?}{=} \mathcal{E}(\vec{0}) \tag{33}$$

The second is implicitly true because $\mathcal{E}(\vec{y})$ encrypts a plaintext modulo p, but the third requires replacing the embed encryption $\mathcal{E}_\beta(\vec{z})$ in $\mathcal{E}(\boldsymbol{m})$ by the encryption of special decomposition $\vec{\sigma}(\vec{z})^* = (\vec{\sigma}(\vec{z})^-, \vec{\sigma}(\vec{z})^+)$ (see eq. (25)), and then check

$$\mathcal{E}_\beta(\vec{z}^+) \odot \left(\vec{1} - \mathcal{E}_\beta(\vec{z}^+)\right) \stackrel{?}{=} \mathcal{E}_\beta(0) \tag{34}$$

$$\left(\vec{1} + \mathcal{E}_\beta(\vec{z}^-)\right) \odot \left(\vec{1} - \mathcal{E}_\beta(\vec{z}^-)\right) \stackrel{?}{=} \mathcal{E}_\beta(0) \tag{35}$$

8.2.3 Signature Verification. The general signature verification resumes to homomorphically test for zero eqs. (23), (24), (26) and (32) to (35). Generating and transmitting $\overrightarrow{\boldsymbol{otk}}$ for all of these tests is cumbersome and inefficient. Fortunately, the number of $\overrightarrow{\boldsymbol{otk}}$ can be reduced to only 2 by grouping the tests $\mathcal{E}_x(\mathtt{pt}_i)$ by decryption modulo, and them compute

$$\mathcal{E}_x(\boldsymbol{T}) = \sum \mathsf{Hash}(\mathcal{E}_x(\boldsymbol{t_i})).\mathcal{E}_x(\boldsymbol{t_i}) \stackrel{?}{=} \mathcal{E}_x(\mathbf{0})$$

and test for zero the result, which will be zero only if all $\mathcal{E}(\boldsymbol{t_i}) \stackrel{?}{=} \mathbf{0}$ are zero encryptions, with high probability[38]. Equations eqs. (23), (24), (26) and (33) can be grouped together and eventually with any tests for zero required to compute over attributes (e.g. is age over 18). All are computed over plaintext p and ciphertext q, and therefore can be decrypted with $\overrightarrow{\boldsymbol{otk}} \in R^{k+2}_{d',q}$. Equations eqs. (32), (34) and (35) are computed on plaintext $\beta.p$ and ciphertext $\beta.q$ and require a different $\overrightarrow{\boldsymbol{otk}}_\beta \in R^{k+2}_{d',\beta.q}$.

The different modulo used in some of the encryptions in equations eqs. (32), (34) and (35) also affect the presentation message. Instead of one encryption $\mathcal{E}(m)$ we need two

$$\mathcal{E}(m_0) = \mathcal{E}\left(\{\vec{0}, \vec{\sigma}(\mathbf{c}), \vec{\sigma}(\vec{\boldsymbol{\mu}}), \vec{\sigma}(\vec{\boldsymbol{o}}), \vec{\sigma}(\vec{\boldsymbol{s}})^*, \vec{v}, \vec{\sigma}(\vec{x})\}\right)$$
$$\mathcal{E}_\beta(m_1) = \mathcal{E}_\beta\left(\{\vec{0}, \vec{\sigma}(\vec{z})^*\}\right)$$

The holder must send both encryptions and both $\overrightarrow{\boldsymbol{otk}}$ to the verifer, which is about $904KB$ in size, assuming a ring size $d' = 8192$, a ciphertext modulus $\approx \log q = 88$ bits, and two $\overrightarrow{\boldsymbol{otk}} \in R^6_{d',q}$ vectors such that $\|\overrightarrow{\boldsymbol{otk}}\|_\infty < 2^{44}$.

This value is cut to $734KB$ using the ring-switching technique from [24] on $\mathcal{E}_\beta(m_1)$, given that the computations with it are shallow and only 256 slots are filled. The encryption $\mathcal{E}_\beta(m_1)$ and the corresponding $\overrightarrow{\boldsymbol{otk}}$ key can be eliminated if a different technique is used to test whether $\mathbf{v}_i \in \mathsf{PRGN}(seed)$ is used or if instead a signature such as the one proposed in [5] is used, which would reduce the overall size of the presentation to $\approx 452KB$. Further reducing the size of the presentation seems to imply a fundamental change in the proposed paradigm.

9 CONCLUSION

With this work, we have shown that competitive and practical solutions for quantum-resistant anonymous credential systems can be built from homomorphic encryption schemas. The solution is not only viable, but also effective in implementing common identity management attributes (e.g. > 18, residence at), and requirements (e.g. revocation). In the last couple of years other solutions for quantum-resistant anonymous credentials have emerged and others are expected to appear based on known strategies not yet explored. Evaluating and comparing all these strategies is a future-relevant work.

REFERENCES

[1] M. Ajtai. 1996. Generating Hard Instances of Lattice Problems (Extended Abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing (STOC '96)*. Association for Computing Machinery, New York, NY, USA, 99–108. https://doi.org/10.1145/237814.237838

[2] Olivier Blazy, Céline Chevalier, Guillaume Renaut, Thomas Ricosset, Eric Sageloli, and Hugo Senet. 2023. Efficient Implementation of a Post-Quantum Anonymous Credential Protocol. In *Proceedings of the 18th International Conference on Availability, Reliability and Security*. ACM, Benevento Italy, 1–11. https://doi.org/10.1145/3600160.3600188

[3] Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. 2013. Key Homomorphic PRFs and Their Applications. In *Advances in Cryptology – CRYPTO 2013 (Lecture Notes in Computer Science)*, Ran Canetti and Juan A. Garay

(Eds.). Springer, Berlin, Heidelberg, 410–428. https://doi.org/10.1007/978-3-642-40041-4_23
[4] Dan Boneh, Amit Sahai, and Brent Waters. 2011. Functional Encryption: Definitions and Challenges. In *Theory of Cryptography (Lecture Notes in Computer Science)*, Yuval Ishai (Ed.). Springer, Berlin, Heidelberg, 253–273. https://doi.org/10.1007/978-3-642-19571-6_16
[5] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Alessandro Sorniotti. 2023. A Framework for Practical Anonymous Credentials from Lattices. In *Advances in Cryptology – CRYPTO 2023 (Lecture Notes in Computer Science)*, Helena Handschuh and Anna Lysyanskaya (Eds.). Springer Nature Switzerland, Cham, 384–417. https://doi.org/10.1007/978-3-031-38545-2_13
[6] Cecilia Boschini, Jan Camenisch, and Gregory Neven. 2018. Relaxed Lattice-Based Signatures with Short Zero-Knowledge Proofs. In *Developments in Language Theory*, Mizuho Hoshi and Shinnosuke Seki (Eds.). Vol. 11088. Springer International Publishing, Cham, 3–22. https://doi.org/10.1007/978-3-319-99136-8_1
[7] Cecilia Boschini, Jan Camenisch, Max Ovsiankin, and Nicholas Spooner. 2020. Efficient Post-quantum SNARKs for RSIS and RLWE and Their Applications to Privacy. In *Post-Quantum Cryptography*, Jintai Ding and Jean-Pierre Tillich (Eds.). Vol. 12100. Springer International Publishing, Cham, 247–267. https://doi.org/10.1007/978-3-030-44223-1_14
[8] Preston Bukaty. 2019. *The California Consumer Privacy Act (CCPA): An Implementation Guide* (illustrated edition ed.). Itgp, Cambridgeshire Business Park , UK.
[9] Jan Camenisch, Manu Drijvers, and Anja Lehmann. 2016. Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited. In *Trust and Trustworthy Computing (Lecture Notes in Computer Science)*, Michael Franz and Panos Papadimitratos (Eds.). Springer International Publishing, Cham, 1–20. https://doi.org/10.1007/978-3-319-45572-3_1
[10] Jan Camenisch and Thomas Groß. 2012. Efficient Attributes for Anonymous Credentials. *ACM Transactions on Information and System Security (TISSEC)* 15, 1 (2012), 1–30.
[11] Jan Camenisch and Anna Lysyanskaya. 2001. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *Advances in Cryptology — EUROCRYPT 2001 (Lecture Notes in Computer Science)*, Birgit Pfitzmann (Ed.). Springer, Berlin, Heidelberg, 93–118. https://doi.org/10.1007/3-540-44987-6_7
[12] Jan Camenisch and Anna Lysyanskaya. 2004. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *Advances in Cryptology – CRYPTO 2004 (Lecture Notes in Computer Science)*, Matt Franklin (Ed.). Springer, Berlin, Heidelberg, 56–72. https://doi.org/10.1007/978-3-540-28628-8_4
[13] Jan Camenisch and Els Van Herreweghen. 2002. Design and Implementation of the Idemix Anonymous Credential System. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS '02)*. Association for Computing Machinery, New York, NY, USA, 21–30. https://doi.org/10.1145/586110.586114
[14] David Chaum. 1985. Security without Identification: Transaction Systems to Make Big Brother Obsolete. *Commun. ACM* 28, 10 (Oct. 1985), 1030–1044. https://doi.org/10.1145/4372.4373
[15] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2016. A Homomorphic LWE Based E-voting Scheme. In *Post-Quantum Cryptography (Lecture Notes in Computer Science)*, Tsuyoshi Takagi (Ed.). Springer International Publishing, Cham, 245–265. https://doi.org/10.1007/978-3-319-29360-8_16
[16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2017. Faster Packed Homomorphic Operations and Efficient Circuit Bootstrapping for TFHE. In *Advances in Cryptology – ASIACRYPT 2017 (Lecture Notes in Computer Science)*, Tsuyoshi Takagi and Thomas Peyrin (Eds.). Springer International Publishing, Cham, 377–408. https://doi.org/10.1007/978-3-319-70694-8_14
[17] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2020. TFHE: Fast Fully Homomorphic Encryption over the Torus. *Journal of Cryptology* 33, 1 (2020), 34–91.
[18] Aisling Connolly, Pascal Lafourcade, and Octavio Perez Kempner. 2022. Improved Constructions of Anonymous Credentials from Structure-Preserving Signatures on Equivalence Classes. In *Public-Key Cryptography – PKC 2022 (Lecture Notes in Computer Science)*, Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe (Eds.). Springer International Publishing, Cham, 409–438. https://doi.org/10.1007/978-3-030-97121-2_15
[19] David Bruce Cousins, Giovanni Di Crescenzo, Kamil Doruk Gür, Kevin King, Yuriy Polyakov, Kurt Rohloff, Gerard W. Ryan, and Erkay Savas. 2018. Implementing Conjunction Obfuscation Under Entropic Ring LWE. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, Oakland, 354–371. https://doi.org/10.1109/SP.2018.00007
[20] Stephen Curran. 2023. AnonCreds Specification. Hyperledger.
[21] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. 2012. Multiparty Computation from Somewhat Homomorphic Encryption. In *Advances in Cryptology – CRYPTO 2012*, Reihaneh Safavi-Naini and Ran Canetti (Eds.). Vol. 7417. Springer Berlin Heidelberg, Berlin, Heidelberg, 643–662. https://doi.org/10.1007/978-3-642-32009-5_38
[22] European Parliament and Council of the European Union. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council. of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation). https://data.europa.eu/eli/reg/2016/679/oj.
[23] Christina Garman, Matthew Green, and Ian Miers. 2013. Decentralized Anonymous Credentials.
[24] Craig Gentry, Shai Halevi, Chris Peikert, and Nigel P. Smart. 2012. Ring Switching in BGV-Style Homomorphic Encryption. In *Security and Cryptography for Networks (Lecture Notes in Computer Science)*, Ivan Visconti and Roberto De Prisco (Eds.). Springer, Berlin, Heidelberg, 19–37. https://doi.org/10.1007/978-3-642-32928-9_2
[25] Craig Gentry, Shai Halevi, and Nigel P. Smart. 2012. Better Bootstrapping in Fully Homomorphic Encryption. In *Public Key Cryptography – PKC 2012*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Marc Fischlin, Johannes Buchmann, and Mark Manulis (Eds.). Vol. 7293. Springer Berlin Heidelberg, Berlin, Heidelberg, 1–16. https://doi.org/10.1007/978-3-642-30057-8_1
[26] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. 2008. Trapdoors for Hard Lattices and New Cryptographic Constructions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*. ACM, Victoria British Columbia Canada, 197–206. https://doi.org/10.1145/1374376.1374407
[27] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. 2015. Leveled Fully Homomorphic Signatures from Standard Lattices. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*. ACM, Portland Oregon USA, 469–477. https://doi.org/10.1145/2746539.2746576
[28] Ilia Iliashenko and Vincent Zucca. 2021. Faster Homomorphic Comparison Operations for BGV and BFV. *Proceedings on Privacy Enhancing Technologies* 2021, 3 (2021), 246–264.
[29] Corentin Jeudy, Adeline Roux-Langlois, and Olivier Sanders. 2022. Lattice Signature with Efficient Protocols, Application to Anonymous Credentials.
[30] Myungsun Kim, Hyung Tae Lee, San Ling, and Huaxiong Wang. 2016. On the Efficiency of FHE-based Private Queries. *IEEE Transactions on Dependable and Secure Computing* 15, 2 (2016), 357–363.
[31] Qiqi Lai, Chongshen Chen, Feng-Hao Liu, Anna Lysyanskaya, and Zhedong Wang. 2023. Lattice-Based Commit-Transferrable Signatures and Applications to Anonymous Credentials.
[32] Adeline Langlois and Damien Stehlé. 2015. Worst-Case to Average-Case Reductions for Module Lattices. *Designs, Codes and Cryptography* 75, 3 (June 2015), 565–599. https://doi.org/10.1007/s10623-014-9938-4
[33] Jorn Lapon, Markulf Kohlweiss, Bart De Decker, and Vincent Naessens. 2011. Analysis of Revocation Strategies for Anonymous Idemix Credentials. In *Communications and Multimedia Security (Lecture Notes in Computer Science)*, Bart De Decker, Jorn Lapon, Vincent Naessens, and Andreas Uhl (Eds.). Springer, Berlin, Heidelberg, 3–17. https://doi.org/10.1007/978-3-642-24712-5_1
[34] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. 2017. Zero-Knowledge Arguments for Lattice-Based PRFs and Applications to E-Cash. In *Advances in Cryptology – ASIACRYPT 2017 (Lecture Notes in Computer Science)*, Tsuyoshi Takagi and Thomas Peyrin (Eds.). Springer International Publishing, Cham, 304–335. https://doi.org/10.1007/978-3-319-70700-6_11
[35] Michael Lodder and Dmitry Khovratovich. 2019. Anonymous Credentials 2.0.
[36] Anna Lysyanskaya. 2002. *Signature Schemes and Applications to Cryptographic Protocol Design.* Thesis. Massachusetts Institute of Technology.
[37] Vadim Lyubashevsky and Daniele Micciancio. 2006. Generalized Compact Knapsacks Are Collision Resistant. In *Automata, Languages and Programming (Lecture Notes in Computer Science)*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.). Springer, Berlin, Heidelberg, 144–155. https://doi.org/10.1007/11787006_13
[38] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. 2022. Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General. In *Advances in Cryptology – CRYPTO 2022 (Lecture Notes in Computer Science)*, Yevgeniy Dodis and Thomas Shrimpton (Eds.). Springer Nature Switzerland, Cham, 71–101. https://doi.org/10.1007/978-3-031-15979-4_3
[39] Vadim Lyubashevsky, Ngoc Khanh Nguyen, Maxime Plancon, and Gregor Seiler. 2021. Shorter Lattice-Based Group Signatures via "Almost Free" Encryption and Other Optimizations.
[40] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. 2021. SMILE: Set Membership from Ideal Lattices with Applications to Ring Signatures and Confidential Transactions.
[41] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. On Ideal Lattices and Learning with Errors over Rings. In *Advances in Cryptology – EUROCRYPT 2010 (Lecture Notes in Computer Science)*, Henri Gilbert (Ed.). Springer, Berlin, Heidelberg, 1–23. https://doi.org/10.1007/978-3-642-13190-5_1
[42] Daniele Micciancio and Chris Peikert. 2012. Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In *Advances in Cryptology – EUROCRYPT 2012 (Lecture Notes in Computer Science)*, David Pointcheval and Thomas Johansson (Eds.).

Springer, Berlin, Heidelberg, 700–718. https://doi.org/10.1007/978-3-642-29011-4_41

[43] Daniele Micciancio and Oded Regev. 2007. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM J. Comput.* 37, 1 (2007), 267–302.

[44] Daniele Micciancio and Oded Regev. 2009. Lattice-Based Cryptography. In *Post-Quantum Cryptography*, Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen (Eds.). Springer, Berlin, Heidelberg, 147–191. https://doi.org/10.1007/978-3-540-88702-7_5

[45] National Institute of Standards and Technology. 2023. *Module-Lattice-Based Digital Signature Standard.* Technical Report Federal Information Processing Standards Publications (FIPS PUBS) 204. U.S. Department of Commerce, Washington, D.C. https://doi.org/10.6028/NIST.FIPS.204.ipd

[46] National Institute of Standards and Technology. 2023. *Module-Lattice-Based Key-Encapsulation Mechanism Standard.* Technical Report Federal Information Processing Standards Publications (FIPS PUBS) 203. U.S. Department of Commerce, Washington, D.C. https://doi.org/10.6028/NIST.FIPS.203.ipd

[47] Chris Peikert and Alon Rosen. 2006. Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices. In *Theory of Cryptography (Lecture Notes in Computer Science)*, Shai Halevi and Tal Rabin (Eds.). Springer, Berlin, Heidelberg, 145–166. https://doi.org/10.1007/11681878_8

[48] Benjamin Hong Meng Tan, Hyung Tae Lee, Huaxiong Wang, Shuqin Ren, and Khin Mi Mi Aung. 2020. Efficient Private Comparison Queries over Encrypted Databases Using Fully Homomorphic Encryption with Finite Fields. *IEEE Transactions on Dependable and Secure Computing* 18, 6 (2020), 2861–2874.

A PSM DETAILED ALGORITHMS

We describe below the detailed PSM algorithms.

Algorithm 2 Spread Function

```
function SPREAD($\mathcal{E}(x)$, d)
    $t \leftarrow 1$
    $\mathcal{E}(\mathrm{y}) \leftarrow \mathcal{E}(1)$
    $\mathcal{E}(\mathrm{x}) \leftarrow \mathcal{E}(x)$
    while $d > 1$ do
        if d is odd then
            $\mathcal{E}(\mathrm{y}) \leftarrow \mathcal{E}(\mathrm{x}) + \mathcal{E}(\mathrm{y}) \lll t$
        end if
        $\mathcal{E}(\mathrm{x}) \leftarrow \mathcal{E}(\mathrm{x}) + \mathcal{E}(\mathrm{x}) \lll t$
        $t \leftarrow 2 * t$
        $d \leftarrow \lfloor d/2 \rfloor$
    end while
    return $\mathcal{E}(\mathrm{x}) + \mathcal{E}(\mathrm{y}) \lll t$
end function
```

Algorithm 3 Sum All Slots Function

```
function SUMALL($\mathcal{E}(\mathrm{x})$, d)
    $t \leftarrow 2^{\lfloor \log_2 d \rfloor - 1}$
    $t_1 \leftarrow 2^{\lfloor \log_2 d \rfloor}$
    $\mathcal{E}(\mathrm{y}) \leftarrow \mathcal{E}(\mathrm{x})$
    while $t > 0$ do
        $\mathcal{E}(\mathrm{y}) \leftarrow \mathcal{E}(\mathrm{x}) + \mathcal{E}(\mathrm{y}) \ggg t$
        if $t$ & $d$ then
            $\mathcal{E}(\mathrm{x}) \leftarrow \mathcal{E}(\mathrm{x}) + \mathcal{E}(\mathrm{y}) \ggg t_1$
            $t_1 \leftarrow t$
        end if
        $t \leftarrow \lfloor t/2 \rfloor$
    end while
    return $\mathcal{E}(\mathrm{x})$
end function
```

Algorithm 4 Subtract and Map

```
function SUBMAPADD($\mathcal{E}(\mathrm{x})$, $\overrightarrow{s}$, $e$)
    $\mathcal{E}(\mathbf{o}) \leftarrow \perp$
    for all $\mathrm{s}_i \leftarrow \vec{s}$ do
        $\mathcal{E}(\mathrm{pt}) \leftarrow \mathcal{E}(\mathrm{x}) - \mathrm{s}_i$
        $\mathcal{E}(\mathrm{pt}) \leftarrow$ ISZERO($\mathcal{E}(o)$)    ▷ Map every slot to $\mathcal{E}(0)$ or $\mathcal{E}(1)$
        $\mathcal{E}(\mathbf{o}) \leftarrow \mathcal{E}(\mathbf{o}) +$ PRODALL($\mathcal{E}(\mathrm{pt})$, $e$)
    end for
    return $\mathcal{E}(\mathbf{o})$
end function
```

B GPV-LIKE SIGNATURE SECURITY PROOF

To prove security, we show that if there is an adversary $\mathcal{A}$ that can generate a new valid signature $(\vec{\mu}, \vec{\varsigma})$ in an existential forgery game, we can build a simulator $\mathcal{B}$ that solves $\mathbf{RSIS}_{d,q,k+2,B_t}$, $\mathbf{RSIS}_{d,q,m,B_\mu}$ or $\mathbf{RSIS}_{d,q,m+k+2,\sqrt{B_\mu^2+B_t^2}}$ with nonnegligible probability.

The adversary $\mathcal{A}$ can request from $\mathcal{B}$ at most N signatures for chosen messages $\vec{\mu_i}$, for which the simulator $\mathcal{B}$ proceeds as follows. Samples a sort vector $\vec{s_i} \leftarrow \mathcal{D}_{R_q^{k+2},\sigma_s}$, computes $\mathrm{v}_i = \langle \vec{a}, \vec{s_i} \rangle - \langle \vec{b}, \vec{\mu_i} \rangle$, programs $\mathbf{PRGN}(seed)$ to include v_i, and returns a signature $(\vec{\mu_i}, \vec{s_i}, \mathrm{v}_i)$. In some cases, the adversary may choose to request only the one-time public key v_i. In such situations, the simulator $\mathcal{B}$ samples a message $\vec{\mu_i} \leftarrow R_q^m$ and proceeds as before.

The distribution of the signature tuples $(\vec{\mu_i}, \vec{s_i}, \mathrm{v}_i)$ generated by this procedure is indistinguishable from the one generated by the actual signature schema. Notice that $\vec{s_i}$ is drawn from the same distribution as when it is sampled by **SamplePre** and v_i is uniformly random, provided that σ_s is greater than the smoothing factor of the lattice $\Lambda_{\frac{\perp}{\vec{a}}}$ smoothing factor [43].

The existential forgery $(\vec{\mu}, \vec{s}, \mathrm{v})$ provided by the adversary $\mathcal{A}$ at the end of the game must satisfy equations eqs. (16) to (19) to win the game. To this end, the adversary $\mathcal{A}$ must choose v from the set of one-time public keys disclosed $[\mathrm{v}_1, \ldots, \mathrm{v}_l]$, $l < N$, or guess one $\mathrm{v} = [\mathrm{v}_{l+1}, \ldots, \mathrm{v}_N]$ not yet disclosed. Assuming a secure PRGN, the probability of guessing one not yet disclosed is $P = \frac{N}{|R_q|}$, which is easily negligible.

If $\mathcal{A}$ chooses a $\mathrm{v} \leftarrow [\mathrm{v}_1, \ldots, \mathrm{v}_l]$ then $\mathcal{B}$ has a signature $(\vec{\mu_i}, \vec{s_i}, \mathrm{v})$ for the same v, and may compute

$$\langle \vec{a}, \vec{s} - \vec{s_i} \rangle = \langle \vec{b}, \vec{\mu} - \vec{\mu_i} \rangle \tag{36}$$

If $\vec{\mu} = \vec{\mu_i}$ then $\mathcal{B}$ outputs $\vec{s}\,' = \vec{s} - \vec{s_i}$ as a solution for $\mathbf{RSIS}_{d,q,k+2,B_t}$. If $\vec{s} = \vec{s_i}$ then $\mathcal{B}$ outputs $\vec{m} = \vec{\mu} - \vec{\mu_i}$ as a solution for $\mathbf{RSIS}_{d,q,m,B_\mu}$, otherwise it outputs $\vec{c} = [\vec{s} - \vec{s_i} \parallel \vec{\mu} - \vec{\mu_i}]$ as a solution for

$$\mathbf{RSIS}_{d,q,m+k+2,\sqrt{B_\mu^2+B_t^2}}$$

C EXACT NORM BOUND

In Section 8.2.1 we have shown how to check an approximate bound 2^w of the norm of a specific vector $\vec{s}$ encoded in the encrypted presentation when that vector is sampled by **TrampSampl**. We rely on the sampled vector satisfying $\|\vec{s}\| < s$ w.h.p. to encode each element of $\vec{\sigma}(\vec{s})$ in $w+1$ bits; one for the sign and $w = \left\lceil \log s\sqrt{d} \right\rceil$ for the modulo. Let $\vec{\sigma}(\vec{s})^- \in \{1,-1\}^{d(k+2)}$ denote the vector with the sign of each element of $\vec{\sigma}(\vec{s})$, and $\vec{\sigma}(\vec{s})^+ \in \{0,1\}^{d.w(k+2)}$ the bit decomposition of the modulo of each element of $\vec{\boldsymbol{\sigma}}(\vec{\mathbf{s}})$, then it is possible to both recover

$$\vec{\boldsymbol{\sigma}}(\vec{\mathbf{s}}) = \vec{\boldsymbol{\sigma}}(\vec{\mathbf{s}})^{-} \odot (G.\,\vec{\boldsymbol{\sigma}}(\vec{\mathbf{s}})^{+})$$

and to verify that $\vec{\boldsymbol{\sigma}}(\vec{\mathbf{s}})^{-} \in \{1,-1\}^{d(k+2)}$, and that $\vec{\boldsymbol{\sigma}}(\vec{\mathbf{s}})^{+} \in \{0,1\}^{d.w(k+2)}$ by using lemmas 2 and 1, homomorphically.

Although, ensuring that $\| \vec{\mathbf{s}} \|_\infty < 2^w$ is usually enough to ensure $\mathbf{RSIS}_{d,q,m,B'_t}$ hardness, with

$$B'_t = 2^w \sqrt{d(k+2)} < \min(p, 2^{\sqrt{4d(k+2)\log p \log \delta}})$$

(cf. eq. (20)) it does not prove that $\| \vec{\mathbf{s}} \|_\infty < s$. However, if necessary, following the strategy from [38], the holder may send another binary encrypted decomposed value $\vec{b}^{\,+} \in \{0,1\}^{d.w(k+2)}$, such that

$$s.\,\vec{1} = G.\left(\vec{\sigma}(\vec{\mathbf{s}})^{+} + \vec{b}^{\,+}\right) \tag{37}$$

The verifier checks that $\mathcal{E}\left(\vec{b}^{\,+}\right)$ encrypt a binary vector and that eq. (37) holds homomorphically, which indirectly ensures that $\| \vec{\sigma}(\vec{\mathbf{s}})^{+} \|_\infty < s$

D OTK DECRYPTION ON BIGGER MODULO

The proof of the correct decryption of eq. (30) using the $\mathbf{OTKDecrypt}_\beta$ algorithm is just an algebraic unroll of the $\mathbf{OTKDecrypt}_\beta$ definition that we described here for completeness.

Specifically, we want to show that, if

$$\begin{aligned} &\mathcal{E}(\mathbf{c}) = \beta.\mathcal{E}(\mathbf{y}) + \mathcal{E}_\beta(\mathbf{z}) - p.\mathcal{E}(\mathbf{x}) \pmod{\beta p} \\ &\beta \ll p \end{aligned}$$

then

$$\mathsf{OTKDecrypt}_\beta(\boldsymbol{PK}, \overrightarrow{\boldsymbol{otk}}, \mathcal{E}(\mathbf{c})) = \beta.\mathbf{y} + \mathbf{z} - p.\mathbf{x}$$

Let $\mathcal{E}(\mathbf{c}) = (\mathbf{c}_0, \mathbf{c}_1), \mathcal{E}(\mathbf{y}) = (\mathbf{y}_0, \mathbf{y}_1), \mathcal{E}_\beta(\mathbf{z}) = (\mathbf{z}_0, \mathbf{z}_1)$ and $\mathcal{E}(\mathbf{x}) = (\mathbf{x}_0, \mathbf{x}_1)$, then

$$\begin{aligned}
&\mathsf{OTKDecrypt}_\beta(\boldsymbol{PK}, \overrightarrow{\boldsymbol{otk}}, \mathcal{E}(\mathbf{c}_i)) = \\
&\Gamma\left(\mathbf{c}_1 - \langle \overrightarrow{\boldsymbol{otk}}, \vec{\boldsymbol{b}} \rangle \bmod p.q\right) = \\
&\Gamma\left(\mathbf{c}_1 - \mathbf{s}.\mathbf{c}_0 + \langle \overrightarrow{\boldsymbol{otk}}, \vec{\boldsymbol{e}} \rangle \bmod p.q\right) = \\
&\Gamma\left(\beta(\mathbf{y}_1 - \mathbf{s}.\mathbf{y}_0) + (\mathbf{z}_1 - s.\mathbf{z}_0) - p(\mathbf{x}_0 - s.\mathbf{x}_1) + \right. \\
&\qquad \left. \langle \overrightarrow{\boldsymbol{otk}}, \vec{\boldsymbol{e}} \rangle \bmod p.q\right) = \\
&\Gamma\left(\beta(\Delta\mathbf{y} + \mathbf{e}' + k'q) + (\Delta\mathbf{z} + \mathbf{e}'' + k''\beta q) - p(\Delta\mathbf{x} + \mathbf{e}''' + k'''q) + \right. \\
&\qquad \left. \langle \overrightarrow{\boldsymbol{otk}}, \vec{\boldsymbol{e}} \rangle \bmod p.q\right) = \\
&\Gamma\left(\Delta(\beta\mathbf{y} + \mathbf{z} - p\mathbf{x}) + (\beta\mathbf{e}' + \mathbf{e}'' - p\mathbf{e}''') + (k'\beta q + k''\beta q - k'''p.q) + \right. \\
&\qquad \left. \langle \overrightarrow{\boldsymbol{otk}}, \vec{\boldsymbol{e}} \rangle \bmod p.q\right) = \\
&\Gamma\left(\Delta(\beta\mathbf{y} + \mathbf{z} - p\mathbf{x}) + (\beta\mathbf{e}' + \mathbf{e}'' - p\mathbf{e}''') + \right. \\
&\qquad \left. k\beta q + \langle \overrightarrow{\boldsymbol{otk}}, \vec{\boldsymbol{e}} \rangle \bmod p.q\right) = \qquad\qquad (\text{if } \beta \ll p) \\
&\left\lfloor (\beta\mathbf{y} + \mathbf{z} - p\mathbf{x}) + \frac{1}{\Delta}(\beta\mathbf{e}' + \mathbf{e}'' - p\mathbf{e}''') + k\beta p + \langle \overrightarrow{\boldsymbol{otk}}, \vec{\boldsymbol{e}} \rangle \bmod \beta p \right\rceil = \\
&\beta\mathbf{y} + \mathbf{z} - p\mathbf{x}
\end{aligned}$$