Implementation of a Post-Quantum Anonymous Verifiable Credential Framework

Davide Margaria, Alessandro Pino, Andrea Vesco
Cybersecurity Research Group
LINKS Foundation
Torino, Italy
{name.surname}@linksfoundation.com

Giuseppe D'Alconzo, Antonio J. Di Scala,
Enrico Guglielmino, Carlo Sanna
Cryptography and Number Theory Group
Department of Mathematical Sciences – Politecnico di Torino
Torino, Italy
{name.surname}@polito.it

***Abstract*—Verifiable Credentials (VCs) can play a crucial role for the identity layer of the Internet. VCs allow Holders to share cryptographically verifiable claims issued by trusted Issuers for authentication purposes. However, plaintext VCs can compromise the privacy of the Holders, as they must disclose entire VCs even when only partial information is required. To address this growing concern, the concept of anonymous credentials has been introduced. In addition, with the advent of Cryptographically Relevant Quantum Computers, many cryptography fields, including that of anonymous credentials, face significant security challenges. This threat urge the design, development, and the implementation of Post-Quantum Anonymous Verifiable Credential frameworks. This paper contributes to this challenge by presenting the analysis and selection of a practical Post-Quantum Anonymous Credential framework, the adaptation of the framework to the VC concept introduced by the Self-Sovereign Identity Model, and the software implementation with 128 bit security with the initial performance evaluation.**

***Index Terms*—Post-Quantum Cryptography, Anonymous Credentials, Self-Sovereign Identity, Selective Disclosure.**

I. Introduction

The Self-Sovereign Identity (SSI) is an emerging decentralized identity model [3] that allow a subject to build and prove its identity in a decentralized manner. The SSI model leverages Verifiable Credentials (VCs) and Verifiable Presentations (VPs), currently under standardization in W3C, as the core technologies. A VC is an unforgeable digital credential about the identity of a subject [4]. It contains the metadata to describe properties of the credential (e.g. context, id, type, Issuer, issuance, and expiration dates), the claim(s) about the identity of the subject, and the information for a Verifier of the VC to check the status of revocation. The addition of a proof, with the digital signature made by the Issuer of the VC, makes it tamper-evident and trustworthy.

Credential systems leveraging the SSI model can be described by the Triangle of Trust (ToT) depicted in Figure 1, where three different roles coexist:

This work has been developed within the QUBIP project (https://www.qubip.eu), funded by the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

D. Margaria, A. Pino, A. Vesco, tailored the BLNS framework defined in [1], [2] to the SSI working principles and made an efficient implementation. G. D'Alconzo, A. J. Di Scala, E. Guglielmino, and C. Sanna studied the state of the art of PQ VCs and provided the support for understanding the BLNS framework [1], [2], translating it into ready-to-be-implemented pseudocode.

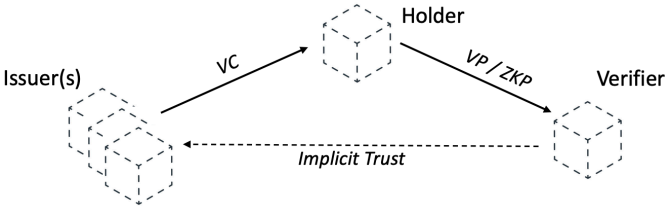

Fig. 1. The Triangle of Trust at the core of the SSI reference model.

- **Issuer(s)** asserts claims about a peer, creates a VC from these claims, and issues the VC to the Holder. In addition, the Issuer manages VC revocation [4] and provides evidences for Verifiers to check the revocation status [5] of all issued VC without compromising the privacy of the Holders.
- **Holder** owns one or more VC and generates a VP [4] to authenticate with a Verifier and request services/resources. A VP is constructed as an envelope of the VC issued by an Issuer, where the proof contains the Holder's signature.
- **Verifier** receives a VP from the Holder and verifies the authenticity by checking the signature made by the Issuer on the VC and by the Holder on the VP, checks the revocation status of the VC and the claim(s) and metadata before granting or rejecting access to the Holder. The Verifiers have an implicit trust on the Issuer(s).

However, credential systems based on *plaintext* VC do not protect the privacy of the Holder because the VC with the claim(s) and the signature of the Issuer are exchanged in plain text. In this sense, plaintext VCs result in linkability and traceability of Holders.

The *anonymous* VC [6], [7] represents a privacy-preserving alternative, that enables the Holder to manage its VC by choosing the level of information to disclose for authentication purposes. The roles in the ToT change as follows:

- **Issuer(s)** asserts the capability of a Holder to prove its identity, possibly based on the knowledge of some secret attributes. After proper verification, the Issuer issues the *anonymous* VC to the Holder, and manages the VC revocation without compromising the privacy of the Holder.
- **Holder** owns one or more *anonymous* VCs and can take advantage of Zero-Knowledge (ZK) primitives and proto-

cols to interact with the Issuer(s) and Verifier in a privacy-preserving way. This means that the Holder can obtain a VC and prove that its identity satisfies certain properties, still maintaining the desired level of privacy. In fact, these operations can happen in a completely anonymous way (i.e. without revealing any identity attributes) or by selectively disclosing only a subset of the claims in the VC (i.e. revealing only selected attributes). Moreover, the Holder can prove to the Verifier that its VC is legit, that means it contains a valid signature generated by an Issuer using a Proof of Knowledge of a Signature (PoKS) [6].

- **Verifier** receives a PoKS from the Holder and verifies that the Holder actually holds a valid VC, containing the undisclosed secrets and a valid signature from an Issuer.

At the core of *anonymous* credential mechanisms there is a PoKS, which consists of a *signature scheme with efficient protocols* [6], informally, a signature scheme with specific features such as the ability to sign committed hidden messages and to prove knowledge of a signature on such messages, and an associated Zero-Knowledge Proof (ZKP) system.

Anonymous credential schemes based on traditional cryptography, exists (e.g. CL [6] or BBS$^+$ [7]), are feasible from the implementation point of view, but are not suitable in the light of the development of a Cryptographically Relevant Quantum Computer (CRQC). While there are proposals for Post-Quantum (PQ) ZKP systems, the state of *PQ signature scheme with efficient protocols* is still at the frontier of research.

This paper presents an extensive work consisting on the analysis of existing PQ anonymous credential frameworks, the selection of a practical framework, and its adaptation to the SSI operating principle and data models. The paper also presents our new open source implementation [8] with a proper experimental assessment of the framework defined by Bootle, Lyubashevsky, Nguyen, and Sorniotti (BLNS) [1], [2]. Overall, this paper contributes to the pioneering work of bringing theoretical results on PQ anonymous credentials to the real world while pursuing the goal of enabling an Internet with quantum-secure ZK authentication with selective disclosure of identity attributes, in accordance with the SSI principles.

II. Review and Selection of Viable Schemes

A. PQ Anonymous Credential

Anonymous credential schemes based on Post-Quantum Cryptography (PQC) have only recently been proposed (not before 2022) [1], [2], [9], [10], with the exception of [11], which is impractical as the signature size is at least 670 MB [9]. Overall, the literature on PQ anonymous credentials is still in its infancy, essentially just the schemes above, and none of them address implementation issues. To the best of our knowledge, only a single implementation of the scheme in [1], [2] is publicly available, the LaZer library [12]. However, it lacks of flexibility on setting the security parameters of the scheme and also shows a limited portability, since it can basically run only on a CPU featuring the AVX-512 instruction set, that is no more supported on modern generations of consumer Intel CPUs.

In order to identify a suitable candidate for implementing PQ anonymous VC among the above options and to achieve a flexible software implementation, we focused on three main aspects: (*i*) the cryptographic assumptions on which the schemes base their security, (*ii*) the signature size, and (*iii*) the completeness and clarity of the pseudocode presented in the papers proposing the schemes.

In terms of cryptographic assumptions, all the candidates are based on lattice assumptions. This choice provides good confidence in current and future security, as the lattice assumptions are at the core of three of the four algorithms selected by NIST for standardisation. Jeudy et al. in [9] present a lattice-based framework for anonymous credentials, improving the previous state-of-the-art [11]. The size of a proof is around 640 KB. Lai et al. in [10] build on top of [9] with the aid of a new cryptographic primitive, called Commit-Transferable Signatures (CTS), to realize the scheme more efficiently. They obtain a proof with dimension around 500 KB. The BLNS framework defined in [1], [2] takes a different approach from the previous two schemes and it exhibits shorter credentials, with a size of around 125 KB. The security of the scheme is based on a new assumption, which is a variation of classic lattice problems.

These candidates have important ideas in common, for example they all use a Non-Interactive Zero-Knowledge proof (NIZK) derived from the NIZK proposed in [13] and they use the Ajtai commitment [14]. Both [9] and [10] exploit the *signature scheme with efficient protocol* paradigm introduced in the seminal work of Camenisch and Lysyanskaya [6] to build the anonymous credential framework, and both [10] and [1] use the sampling trapdoor algorithm described in [15]. In addition, the ISIS_f assumption on which the BLNS framework [1] is based is a very natural generalization of the underlying problem upon classic lattice-based signature schemes such as [15] are based, and it is also similar to other recently, and independently, proposed lattice assumptions [16]. Under specific choices of f, the new assumption is proven to be equivalent to SIS, a well-known assumption introduced by Ajtai in [14], on which many constructions base their security, like the Dilithium signature [17] relying on the variant $\mathsf{SelfTargetMSIS}$.

In light of these considerations, we have chosen the BLNS framework [1], [2] because it excels in (*a*) having the shortest proof dimension, resulting in the smallest size for a VC (i.e. around 125 KB), (*b*) having a higher degree of maturity in terms of implementability by providing a detailed pseudocode, and (*c*) presenting an application of anonymous credentials suitable for the PQ anonymous VC scenario.

From a cryptographic point of view, the BLNS framework has common functions with the PQ KEM NTRU [18], also implemented in the FALCON digital signature [19] selected by NIST for future standardization. The BLNS framework [1] uses two functions from NTRU, the trapdoor generator and the Gaussian sampler. More specifically, the NTRU trapdoor consists of a *short* basis $\mathbf{B}$ of a publicly available lattice $\mathcal{L}$. Finding such a special basis is infeasible. The signature

of a message m is constructed using the Gaussian Sampler, taking as input $\mathbf{B}$ and a vector v_m encoding the message, returning a vector s in the lattice $\mathcal{L}$ such that the difference $s - v_m$ is short enough, i.e. it has to satisfy some bounds. This is linked to the general lattice problem called Closest Vector Problem (CVP). Furthermore, the BLNS framework is based on many different cryptographic assumptions that, under some additional hypothesis, are considered equivalent to the Module-SIS (MSIS) or the Module-LWE (MLWE) assumptions [20], two standard cryptographic assumptions both used in the newly standardized ML-KEM and ML-DSA algorithms. These assumptions state that, given some public equations, finding a solution with a small norm is intractable.

Under the above assumptions, the BLNS framework is secure against any coalition including malicious Issuer(s), malicious Holder(s) and Verifier(s) who try to obtain some information about a target Holder. Moreover, any coalition of malicious Verifiers cannot link the same Holder across different verifications.

B. Revocation Mechanism

An efficient revocation mechanism represents a crucial requirement in any credential system, independently of any privacy-enhancing features it offers (e.g. anonymity or selective disclosure). The possible reasons why a credential needs to be revoked can be grouped in three categories: (*i*) natural expiration, i.e. where the credential reaches its expiration date becoming outdated and unusable, (*ii*) Issuer-initiated revocation, where the Issuer revokes the credential before its expiration date and (*iii*) Holder-initiated revocation, where the Holder asks for a revocation of its credential.

While the W3C standard approach to revocation [5] can also be adopted with PQ plaintext VC, because it does not rely on cryptography vulnerable to CRQC, the same approach is not suitable for privacy-sensitive contexts. Two main approaches suitable to implement an efficient revocation mechanism for *anonymous* VC exist:

1) *Accumulator approach*: Camenisch *et al.* in [21], [22] propose the use of a dynamic accumulator to solve the revocation problem. An accumulator is a box storing a set of values that can be sequentially added. Along with this box, a witness to prove that a value is inside the accumulator is provided, for instance a ZK-Proof of Knowledge that a committed value is inside the accumulator. A dynamic accumulator allows deleting values from the box using a trapdoor known only by the owner (or generator) of the accumulator. The revocation consists in the owner removing the credential unique value from the accumulator.
2) *Timestamp approach*: Camenisch *et al.* in [23] had yet another idea to implement a revocation mechanism, based on timestamps. Each credential is valid only for a time interval called *epoch*, and the credential must be refreshed for every epoch. The revocation consists in non-issuing the credential for the next epoch. At verification time, the Verifier checks that the VC is referred to the correct epoch. Credential revocation is implemented by encoding a validity time property (i.e. a timestamp) into one of the Issuer-controlled attributes. Thus, an Issuer can periodically update valid credentials off-line and publish a small per-credential update value, for instance on a public bulletin-board every minute, hour, or day, trading-off complexity and security.

Since there are no PQ versions of the accumulator-based solution, we built on top of the timestamp approach in [23] that is quantum-secure by construction (i.e. it does not rely on asymmetric cryptography vulnerable to a CRQC). This approach has the advantage that the Verifier only checks that the VC is referred to the correct epoch, without having to check a revocation list. The cost of updating the VC is minimal for the Holder and is comparable to other solutions for the Issuer, e.g. [5], [24]. Furthermore, this approach enables a *rich revocation semantic*, since the credential can be partially revoked and/or updated (i.e. only some VC attributes). Notably, this solution is suitable to be directly combined with the BLNS framework, since it just requires an anonymous credential scheme capable to handle multiple attributes, one of which is dedicated to the timestamp.

III. Tailoring the BLNS Framework to PQ VCs

A. General Principles and High-level Architecture

Figure 2 presents the architecture and the dependency graph of BLNS core function implementation. The resulting credential system consists of three actors as in previous Figure 1: a trusted Issuer who issues VCs, the Holder who owns VCs and presents a PQ ZKP to prove his identity, and the Verifier who checks the proof before granting or denying access.

During the *initialization phase*, the Issuer runs the KeyGen algorithm to generate the private-public key pair $(\mathsf{isk}, \mathsf{ipk})$ running the trapdoor generation inherited from NTRU called $\mathsf{NTRU.TrapGen}$.

During the *issuing protocol*, the Holder interacts with the Issuer that signs the attributes attrs assigned to the Holder with his private key, sampling a short vector having some properties with the Gaussian Sampler $\mathsf{GSampler}$. The Holder may hide some of its attributes, indexed by $\overline{\mathsf{idx}}$, using a hiding commitment and producing a NIZK of the well-formedness of this commitment. This NIZK is obtained by running $\mathsf{Prove}_{\mathsf{Com}}$ and $\mathsf{Verify}_{\mathsf{Com}}$, which in turn need a Linearly Homomorphic Commitment (LHC) implemented in the LHC functionalities to achieve the required security properties. Finally, the output of this phase is the VC for the Holder.

During the *verification protocol*, the Verifier checks that the PQ ZKP of the Holder is correct. To produce such a proof, the Holder chooses which attributes to selectively disclose and produces a NIZK of the possession of a signature of the Issuer on those attributes. This is done through the function $\mathsf{Prove}_{\mathsf{ISIS}}$, proving that they know a short vector satisfying a known relation, i.e. the signature. After verifying the hidden signature of the Issuer using $\mathsf{Verify}_{\mathsf{ISIS}}$, the Verifier accepts the PQ ZKP of the Holder.

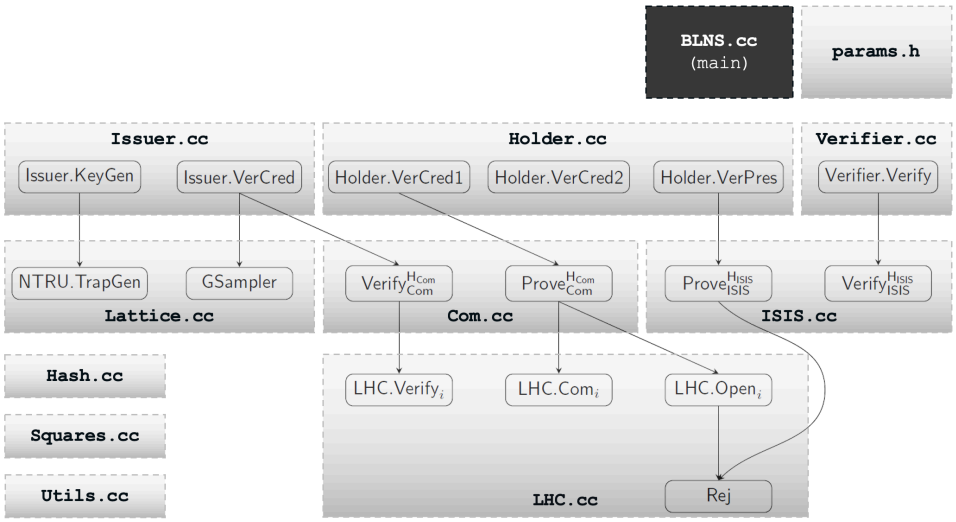

Fig. 2. Architecture and dependency graph of BLNS core functions implementation.

A rejection sampling algorithm Rej is run in multiple points of the interaction, in order to hide the sensitive committed data, and ensuring the security of the scheme.

B. Issuing Protocol

Figure 3 illustrates the protocol between the Issuer and the Holder for issuing a VC. The process involves one round of interaction, where the Holder commits to the set of hidden attributes and the Issuer signs both the hidden and disclosed attributes with his private key. The Holder initializes the VC and sends the material ρ_1 to the Issuer. In this material there is also a zero-knowledge proof that the commitment is well-formed. Upon receiving the signature of the Issuer ρ_2, the Holder finalizes the VC generation.

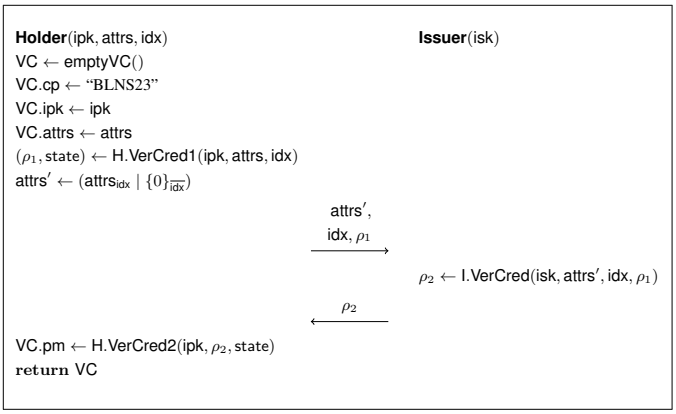

Fig. 3. Issuing protocol.

C. Verification Protocol

Figure 4 illustrates the generation of the PQ ZKP from the Holder's VC and the subsequent verification by a Verifier. The Holder selects the attributes to disclose, and then generates the PQ ZKP as a PoKS of the Issuer on the attributes. The Holder anonymizes the VC and uses Holder.VerPres to generate the PoKS and sends both to the Verifier for verification with the Verifier.Verify.

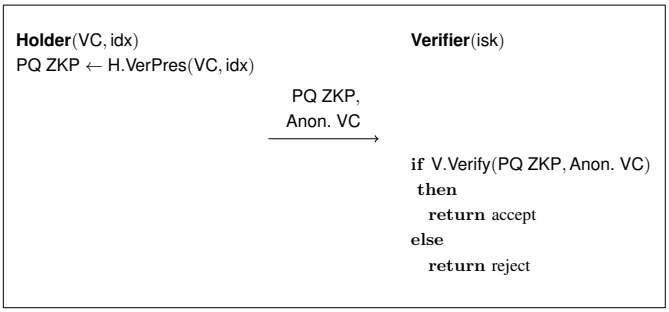

Fig. 4. Verification protocol.

D. Security Parameters

Table I shows the parameters of the framework defined in [2]. For the security level of 128 bit, it reports both the original parameters proposed in [2] (first row) and the ones inspired by the LaZer paper [12] (second row). Note that [2] does not report any parameters sets for the higher security levels (i.e. 192 and 256 bit). We estimated these values starting from the FALCON parameters [19]. FALCON defines the parameters for the key generation (i.e. q, d, $\mathfrak{s}$) for 128 and 256 bit security. Therefore, we maintained the values associated to 256 bit for both 192 and 256 bit security levels and we estimated the remaining parameters (N, h, ψ, ℓ_r, ℓ_m) accordingly.

IV. Software Implementation

A. Key Implementation Choices and External Dependencies

We implemented the BLNS framework in C++ language, following the original pseudocode in [1], [2] and maintaining as much as possible the same notation for functions and variables. In the design of the data structures and functions, we prioritized the code portability and the flexibility on

security parameters with respect to the pure performance. In this sense, we avoided relaying on specific CPU instruction sets or parallelization/multi-threading. These implementation choices have the objective to ease future maintenance and possible improvements on the algorithms and to ensure a good flexibility on the parameters for achieving higher security levels. Among the security parameters in Table I, we used the ones in the second row (inspired by the LaZer paper [12]) to demonstrate the feasibility and achieve an adequate performance, at least for the 128 bit security level.

We investigated some generic arithmetic backends supporting the basic mathematical operations needed for vectors, matrices, and polynomials over the integers and over finite fields, especially the portable libraries NTL [25] and FLINT [26]. Taking into account the comparative analysis reported in [27], we selected the NTL library mainly for its simpler usability and higher expected performance. Moreover, NTL [25] only requires the GMP [28] arithmetic library to be installed, thus limiting the number of external dependencies to two.

It is worth to highlight that the current Issuer.KeyGen and Issuer.VerCred functions use the optimised key generation and Gaussian sampling algorithms from FALCON [19]. Nonetheless, this external dependency can be avoided by setting the flag `USE_FALCON = 0` in the `Makefile`, to directly use our flexible, but less efficient, implementation of the NTRU.TrapGen and GSampler algorithms.

Finally, several functions in the BLNS framework require using a custom hash function. To provide a 128 bit security level, we selected a public domain implementation of the SHAKE128 extendable-output function from SHA-3 family.

B. Core Functions and Preliminary Performance

Figure 2 presents the high-level architecture of our implementation, showing the C++ source files related to core functions in the dashed boxes. Apart the main file (`BLNS.cc`) and the parameter file (`params.h`), the source code is organised in three hierarchical levels: (*i*) the high-level functionalities for the three roles in the BLNS protocols (i.e. `Issuer.cc`, `Holder.cc`, `Verifier.cc`), (*ii*) the mid-level functions related to the key generation, Gaussian sampling (`Lattice.cc`), and the ZK primitives to prove and verify the commitments on secret attributes (`Com.cc`, `ISIS.cc`), (*iii*) the low-level utility functions, custom hash functions, LHC, and algorithms related to the sum of squares (`Utils.cc`, `Hash.cc`, `LHC.cc`, `Squares.cc`).

We have already applied different optimizations to these core functions, achieving some remarkable performance gains, but there is still room for improvements. In this first implementation, we have prioritized the profiling and the optimisation of the most computationally intensive functions for the Credential Issuing and Verification Protocols (i.e. $\mathsf{Prove}_{\mathsf{Com}}$, $\mathsf{Verify}_{\mathsf{Com}}$, $\mathsf{Prove}_{\mathsf{ISIS}}$, $\mathsf{Verify}_{\mathsf{ISIS}}$). In detail, when possible, we have limited the numerical precision to 64 bit for several operations on integer and floating point numbers instead of directly using the arbitrary-precision numbers supported by NTL and GMP libraries. In fact, arbitrary-precision integers and floating points become inefficient with matrices or polynomials with a large size. In the case of integers with a modulus smaller than 64 bit or floating points with small expected absolute values, they can be safely replaced with `long int` or `double` types, without incurring in numerical errors, instabilities, or significant degradation on the precision of the results. We have also refactored some loops and expensive operations in the original pseudocode in [1], significantly reducing the execution time and the memory requirements. Finally, we have tried to minimise dynamic memory allocation and deallocation operations, that rapidly became expensive with matrices/polynomials with a large size, passing them by reference as inputs and outputs between the involved functions.

TABLE I
PARAMETERS FOR DIFFERENT SECURITY LEVELS (λ).

λ	q	d	$\mathfrak{s}$	N	h	ψ	ℓ_r	ℓ_m
128 BLNS	17179861781	4096	2^{27}	2^{640}	512	2	2	1
128 LaZer	12289	512	165.73	2^{512}	64	3	2	1
192	12289	1024	168.39	2^{960}	128	1	1	1
256	12289	1024	168.39	2^{1280}	128	2	2	1

TABLE II
BENCHMARK ON AN INTEL® CORE™ I7-1255U 1.70 GHZ, 24 GB RAM. STATISTICS OVER 1000 RUNS ON SINGLE CORE (TIME IN MILLISECONDS)

Protocol	Function	min	max	average	median	std
Init	Issuer.KeyGen	3.26	13.49	4.94	4.57	1.28
	Holder.Init	25.75	57.90	28.85	28.43	2.01
Credential Issuing	Holder.VerCred1	173.55	3064.23	629.56	467.15	475.98
	Issuer.VerCred	82.56	235.04	114.26	109.37	20.97
	Holder.VerCred2	0.54	3.51	0.74	0.69	0.25
Verification Protocol	Holder.VerPres	317.67	2316.57	543.91	422.67	260.84
	Verifier.Verify	120.39	265.11	155.29	149.13	21.57

Table II presents a preliminary performance assessment of our implementation on a laptop featuring an Intel® Core™ i7-1255U CPU running at 1.70 GHz, 24 GB RAM, Ubuntu 22.04 OS, and the scaling governor set to `performance`. The statistics for each function of the BLNS framework are reported in milliseconds and have been assessed with 1000 executions on a single core, after a warm-up of 100 executions, explicitly disabling the thread pools on the NTL library and independently generating random keys and messages at each iteration. Each execution involves the use of a random VC with 8 attributes, with the Holder selectively disclosing only 4 attributes to the Issuer and Verifier for authentication purpose.

All core functions show reasonable performance for the PQ anonymous VC use case, with average and median execution times in the order of hundreds of milliseconds. The average time for a single repetition of all functions (i.e. the complete BLNS scheme) is approximately 1.48 s.

Concerning the initialization of the protocols, the Issuer.KeyGen is faster (approx. 5 ms) than the Holder.Init (i.e. the generation of common random strings and matrices, requiring approx. 29 ms). The Credential Issuing protocol is significantly more expensive (approx. 745 ms) and consists in Holder.VerCred1, Issuer.VerCred, and Holder.VerCred2.

However, in a real deployment these operations are executed only once to generate a valid VC.

After that, the Holder can execute the Verification Protocol with a Verifier with an average execution time of 699 ms. It is worth to remark that the Holder needs 544 ms to prove the knowledge of the signature and attributes in the VC, while the Verifier can verify it in just 155 ms. In the current implementation, the size of this proof is approx. 100 KB, that is a feasible dimension for the PQ anonymous VC use case.

Finally, it can be noted that the last column of Table II reports some large standard deviation values (std) for two functions: Holder.VerCred1 and Holder.VerPres. These results can be explained and justified by the rejection sampling algorithm Rej [1] that is used in $\mathsf{Prove}_{\mathsf{Com}}$ and $\mathsf{Prove}_{\mathsf{ISIS}}$: it results in multiple iterations for their internal computations to find a valid solution (i.e. an average of 4.6 and 2.5 repetitions, respectively) and, thus, affects the current performance of Holder.VerCred1 and Holder.VerPres.

V. Conclusions and Future Works

This paper has presented the implementation of the BLNS framework to start the journey toward an Internet with quantum-secure ZK authentication with selective disclosure of identity attributes in accordance with the SSI principles. The implementation, published in open-source [8] for the whole community, provides promising performance.

While flexible and already compatible to different security parameters, our implementation still has significant room for optimization. We have planned to improve some mathematical aspects (e.g. investigate alternatives to reduce the impact of the rejection sampling) and the efficiency of the underlying algorithms under multiple configurations (e.g. different number of hidden and/or disclosed attributes). We will also optimize the implementation by working on an appropriate data serialization to diminish the proofs dimensions and to enhance the overall communication efficiency. Security will also be a key focus for future work, especially aiming to a constant-time implementation to mitigate timing attacks.

Acknowledgments

The authors would like to thank Gessica Alecci and Andrea Gangemi (Politecnico di Torino) for their help in writing and reviewing the pseudocode, and Alberto Carelli (LINKS Foundation) for the valuable suggestions to improve the efficiency of the implementation.

References

[1] J. Bootle, V. Lyubashevsky, N. K. Nguyen, and A. Sorniotti, "A Framework for Practical Anonymous Credentials from Lattices," Cryptology ePrint Archive, Paper 2023/560, 2023, https://eprint.iacr.org/2023/560.

[2] ——, "A Framework for Practical Anonymous Credentials from Lattices," in *CRYPTO 2023*, Santa Barbara (CA), Aug. 2023, pp. 384–417.

[3] A. Preukschat and D. Reed, *Self-Sovereign Identity: Decentralized digital identity and verifiable credentials*. NY: Manning, 2021.

[4] W3C, "Verifiable Credentials Data Model v2.0 W3C Recommendation," 2024, https://www.w3.org/TR/vc-data-model-2.0/.

[5] ——, "Bitstring Status List v1.0: Privacy-preserving status information for Verifiable Credentials. W3C Working Draft," 2024, https://www.w3.org/TR/vc-bitstring-status-list/.

[6] J. Camenisch and A. Lysyanskaya, "A Signature Scheme with Efficient Protocols," in *International Conference on Security in Communication Networks*, Amalfi (IT), Sep. 2002, pp. 268–289.

[7] J. Camenisch, M. Drijvers, and A. Lehmann, "Anonymous Attestation Using the Strong Diffie-Hellman Assumption Revisited," in *International Conference on Trust and Trustworthy Computing*, Vienna (AT), Aug. 2016, pp. 1–20.

[8] LINKS Foundation, "Implementation of BLNS Framework for PQ Anonymous Verifiable Credentials," 2025, https://github.com/Cybersecurity-LINKS/pqzk-blns.

[9] C. Jeudy, A. Roux-Langlois, and O. Sanders, "Lattice Signature with Efficient Protocols, Application to Anonymous Credentials," Cryptology ePrint Archive, Paper 2022/509, 2022, https://eprint.iacr.org/2022/509.

[10] Q. Lai, C. Chen, F.-H. Liu, A. Lysyanskaya, and Z. Wang, "Lattice-based Commit-Transferrable Signatures and Applications to Anonymous Credentials," Cryptology ePrint Archive, Paper 2023/766, 2023, https://eprint.iacr.org/2023/766.

[11] B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang, "Signature Schemes with Efficient Protocols and Dynamic Group Signatures from Lattice Assumptions," in *International Conference on the Theory and Applications of Cryptology and Information Security*, Hanoi (VN), Dec. 2016, pp. 373–403.

[12] V. Lyubashevsky, G. Seiler, and P. Steuer, "The LaZer Library: Lattice-Based Zero Knowledge and Succinct Proofs for Quantum-Safe Privacy," in *ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 3125–3137.

[13] V. Lyubashevsky, N. K. Nguyen, and M. Plançon, "Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General," in *CRYPTO 2022*, Y. Dodis and T. Shrimpton, Eds. Cham: Springer Nature Switzerland, 2022, pp. 71–101.

[14] M. Ajtai, "Generating Hard Instances of Lattice Problems," *Electronic Colloquium on Computational Complexity*, vol. 96, no. 7, Jan. 1996, https://eccc.weizmann.ac.il/report/1996/007/download.

[15] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for Hard Lattices and New Cryptographic Constructions," Cryptology ePrint Archive, Paper 2007/432, 2007, https://eprint.iacr.org/2007/432.

[16] M. R. Albrecht, V. Cini, R. W. F. Lai, G. Malavolta, and S. A. Thyagarajan, "Lattice-Based SNARKs: Publicly Verifiable, Preprocessing, and Recursively Composable," Cryptology ePrint Archive, Paper 2022/941, 2022, https://eprint.iacr.org/2022/941.

[17] L. Ducas *et al.*, "CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 1, p. 238–268, Feb. 2018, https://tches.iacr.org/index.php/TCHES/article/view/839.

[18] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *Algorithmic Number Theory*, J. P. Buhler, Ed. Berlin, Heidelberg: Springer, 1998, pp. 267–288.

[19] P.-A. Fouque *et al.*, "FALCON: Fast-Fourier Lattice-based Compact Signatures over NTRU v1.2," 2020, https://falcon-sign.info.

[20] A. Langlois and D. Stehlé, "Worst-case to Average-case Reductions for Module Lattices," *Designs, Codes and Cryptography*, vol. 75, no. 3, pp. 565–599, Jun. 2015.

[21] J. Camenisch and A. Lysyanskaya, "Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials," in *CRYPTO 2002*, M. Yung, Ed. Berlin, Heidelberg: Springer, 2002, pp. 61–76.

[22] J. Camenisch, M. Kohlweiss, and C. Soriente, "An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials," in *Public Key Cryptography*, S. Jarecki and G. Tsudik, Eds. Berlin, Heidelberg: Springer, 2009, pp. 481–500.

[23] ——, "Solving Revocation with Efficient Update of Anonymous Credentials," in *Security and Cryptography for Networks*, J. A. Garay and R. De Prisco, Eds. Berlin, Heidelberg: Springer, 2010, pp. 454–471.

[24] Y. Sheffer *et al.*, "Support for Short-Term, Automatically Renewed (STAR) Certificates in the Automated Certificate Management Environment," RFC 8739, Mar. 2020, https://www.rfc-editor.org/info/rfc8739.

[25] V. Shoup, "NTL: A Library for doing Number Theory," https://libntl.org.

[26] The FLINT team, "FLINT: Fast Library for Number Theory," https://flintlib.org.

[27] V. Shoup, "NTL vs FLINT," 2021, https://libntl.org/benchmarks.pdf.

[28] The GMP team, "GMP: The GNU Multiple Precision Arithmetic Library," https://gmplib.org/.