Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Pure and Applied Mathematics (38^{th}cycle)

Cryptographic Protocols for Blockchain Interoperability and Self-Sovereign Identity

By

Enrico Guglielmino

******

Supervisor(s):
Prof. Danilo Bazzanella, Supervisor
Prof. Carlo Sanna, Co-Supervisor
Dr. Andrea Gangemi, Co-Supervisor

Doctoral Examination Committee:
Prof. Alessandro Zaccagnini, Referee, University of Parma
Dr. Giuliano Romeo, Referee, University of Trento

DISMA "G.L. Lagrange" - Politecnico di Torino
2022-2026

Abstract

The increasing adoption of blockchain systems and Self-Sovereign Identity frameworks is reshaping the way digital assets and personal data are managed, enabling new forms of decentralization and user control. However, these systems also introduce important challenges, particularly in enabling interoperability across heterogeneous blockchains and in designing secure privacy-preserving identity protocols. In this context, this thesis develops cryptographic protocols that address these challenges: a new protocol for atomic swaps is introduced, a post-quantum implementation of an anonymous credential scheme is presented, and a scheme for delegatable presentations of verifiable and anonymous credentials is proposed.

The thesis is organized into two main parts. The first part provides the theoretical background required to understand the contributions developed in the second part. In particular, Chapter 1 introduces the main cryptographic preliminaries, Chapter 2 describes the core principles of blockchain technology, and Chapter 3 focuses on verifiable and anonymous credential schemes. The second part of the thesis presents the original contributions. In Chapter 4, a new two-phase protocol (BTLE) for the decentralized exchange of digital assets between heterogeneous blockchains is presented. BTLE replaces the hash-time-locked contract (HTLC) mechanism commonly used in atomic swaps with a construction based on time-lock puzzles, thereby enabling exchanges even between blockchains that lack a compatible scripting language or shared hash functions. A second contribution of this work is the introduction of a new time-lock puzzle construction based on Pell conics, offering better performance compared to the classical approach of Rivest, Shamir, and Wagner. Chapter 5 focuses on the design and implementation of a post-quantum anonymous credential framework for Self-Sovereign Identity systems, and presents an initial performance evaluation at the 128-bit security level. Finally, Chapter 6 introduces the formal notion of a verifiable presentation delegation scheme. The work defines the core syntax and security requirements of delegation schemes, including correctness, unforgeability,

and unlinkability, and presents two concrete instantiations: one compatible with verifiable credentials as deployed in the EUDI Architecture Reference Framework, and one supporting BBS anonymous credentials. Both schemes are formally proven secure, and their integration into real-world infrastructures, such as the European Blockchain Services Infrastructure (EBSI) and the EUDI framework, is discussed.

Contents

List of Figures

List of Tables

Notation

Notation	Description
$[l]$	The set $\{1,\ldots,l\}$
$x_i, x[i]$	i-th component of the vector x
λ	Security parameter
pp	Public parameters
PPT	Probabilistic polynomial time
$H(-)$	Cryptographic hash function
$\mathsf{Invert}_{\mathscr{A},f}$	Invert Experiment for hash functions
$\mathsf{Hash}-\mathsf{coll}_{\mathscr{A},\mathscr{H}}$	Collision-Resistance Experiment for hash functions
$\mathsf{Second}-\mathsf{preimage}_{\mathscr{A},\mathscr{H}}$	Second preimage Experiment for hash functions
qSDH	q-Strong Diffie-Hellman assumption
$\mathscr{DS}$	Digital signature scheme
M	Message
$\mathsf{Setup}(-)$	Public parameter generation algorithm
$\mathsf{KeyGen}(-)$	Digital signature key generation algorithm
$\mathsf{Sign}(-)$	Signature generation algorithm
$\mathsf{Vf}(-)$	Signature verification algorithm
$\mathbf{e}:\mathbb{G}_1\times\mathbb{G}_2\rightarrow\mathbb{G}_T$	Bilinear pairing
$\mathsf{Com}(-)$	Commitment generation algorithm
$\mathsf{Open}(-)$	Commitment opening algorithm
P	Prover
V	Verifier
$\mathscr{R}$	Relation
$\mathsf{P}_{\mathsf{com}}(-)$	Commitment-generation algorithm in a Sigma protocol
$\mathsf{P}_{\mathsf{resp}}(-)$	Response-generation algorithm in a Sigma protocol
ch	Challenge
NIZKP	Non-interactive zero-knowledge proof
Sim	Simulator
SSI	Self-Sovereign Identity
$\mathsf{pk}_{\mathsf{Iss}}$	Issuer public key

Table 1 Notation used throughout the thesis.

Notation	Description
$\mathsf{sk}_{\mathsf{Iss}}$	Issuer private key
VC	Verifiable credential
AC	Anonymous credential
cred	Credential data structure
$\mathbf{a} = \{a_i\}_{i \in [l]}$	Set of attributes
$\mathsf{IssuerSetup}(-)$	Issuer setup algorithm
$\mathsf{CredIssuance}(-)$	Credential issuing protocol
$\mathsf{CredPresentation}(-)$	Presentation protocol
$\mathsf{PresVer}(-)$	Presentation verification protocol
$TLP(-)$	Time-lock puzzle
$\mathrm{CondTx}(-)$	Conditional transaction
CT	Credential table
PT	Presentation table
DT	Delegation table
D	Delegator
Δ	Delegatee
Δ_{ID}	Delegatee identifier
DP	Delegator payload
scope	Delegation scope
Hid	Set of indices of hidden attributes
$\mathsf{Rev} = [l] \setminus \mathsf{Hid}$	Set of indices of disclosed attributes
stmt	Set of revealed attributes
$\mathscr{S}$	Set of possible statements
$\mathsf{DelegIssuance}(-)$	Delegation issuance algorithm
DelegVer	Delegation verification algorithm
$\mathsf{DelegPres}(-)$	Delegated presentation algorithm
$\mathsf{DelegPresVer}(-)$	Delegated presentation verification algorithm
eIDAS	Electronic IDentification, Authentication, and trust Services
EUDI wallet	European digital identity wallet
EUDI ARF	European digital identity architecture and reference framework
$\mathsf{Exp}^{\mathsf{DelPresUnforgeability}}_{\mathscr{A}}$	Unforgeability Experiment
$\mathsf{Exp}^{\mathsf{DelegIssuanceUnlinkability}}_{\mathscr{A}}$	Unlinkability Experiment for Delegation Issuance
$\mathsf{Exp}^{\mathsf{DelegPresUnlinkability}}_{\mathscr{A}}$	Unlinkability Experiment for Delegated Presentation
$\mathsf{Exp}^{\mathsf{Indistinguishability}}_{\mathscr{R}}$	Indistinguishability Experiment

Table 2 Notation used throughout the thesis.

Part I
Basic Preliminaries

This first part of the thesis introduces the theoretical foundations upon which the original contributions of this work are based. After presenting the fundamental concepts of cryptography in the first chapter, the following chapters focus on two key applications: blockchain technology and verifiable and anonymous credentials. These topics are of particular importance, as they provide the foundation for the developments presented in the second part of the thesis.

Chapter 1

Cryptographic Preliminaries

This chapter presents the cryptographic preliminaries that will be used throughout this thesis. In particular, Section 1.1 introduces some basic cryptographic notions, while Section 1.2 presents the fundamental definitions and properties of Elliptic Curve Cryptography. Section 1.3 introduces cryptographic hash functions, which constitute a fundamental building block in the construction of blockchain systems, and discusses their fundamental cryptographic properties. Section 1.4 introduces the cryptographic assumptions that are used throughout the original contributions of the thesis. Section 1.5 introduces the Random Oracle Model, an idealized framework in which hash functions are modeled as truly random functions accessible to all parties. Section 1.6 presents digital signature schemes, illustrating their main properties and introducing the BBS signature scheme as a representative example. Section 1.7 introduces commitment schemes, with a particular focus on the binding and hiding properties. Section 1.8 is devoted to Merkle trees, explaining their structure and function within blockchain systems, as well as their importance for data integrity and efficient verification. Section 1.9 provides an overview of sigma protocols, describing the security properties they are required to satisfy. Section 1.10 introduces the Fiat-Shamir transform, a fundamental technique for converting interactive proof systems into non-interactive ones. Section 1.11 presents non-interactive zero-knowledge proofs, which enable the verification of statements without revealing any additional information. Finally in Section 1.12, timed-release cryptography was introduced as a mechanism for delaying access to encrypted information and the time-lock puzzle construction proposed by Rivest, Shamir, and Wagner is presented.

1.1 Basic notions

In this section, some basic cryptographic notions are introduced.

Definition 1.1.1 (Security parameter) *A scheme has security parameter λ (bit level security) if in order to break the scheme an adversary requires (on average) 2^λ operations.*

Usually $\lambda = 128, 192, 256$. The complexity of an algorithm is expressed in terms of the size (in bits) of the input of the algorithm. In particular, if an algorithm $\mathscr{A}$ takes as input an integer N, then the complexity of $\mathscr{A}$ is given in terms of the number of bits of N, which is approximately equal to $n = \log N$, where log is the logarithm in base 2. Therefore, $\mathscr{A}$ is polynomial-time if its complexity is $\mathscr{O}((\log N)^c)$ for some costant $c > 0$, and not $\mathscr{O}(N^c)$. In such a case $\mathscr{A}$ would be exponential-time, since $N = 2^n = 2^{\log N}$. However, oftentimes one wants that a polynomial-time algorithm $\mathscr{A}$ is polynomial-time in terms the security parameter λ (non in terms of the size of λ. A simple trick[1] is to give 1^λ as input to $\mathscr{A}$, where 1^λ is defined as the binary string $\underbrace{11\cdots1}_{\lambda \text{ times } 1}$. In the remainder of this thesis we will use this last notation.

Definition 1.1.2 (Asymptotic notation) *Let $f, g : \mathbb{N} \to \mathbb{R}$ be two functions. We write $f(n) = \mathscr{O}(g(n))$ if there exist a costant $C > 0$ and $n_0 \in \mathbb{N}$ such that $|f(n)| \leq C|g(n)|$ for all $n > n_0$.*

Definition 1.1.3 (Negligible function) *A function $f : \mathbb{N} \to \mathbb{R}$ is negligible if for every $c > 0$ it holds $f(n) = \mathscr{O}(\frac{1}{n^c})$.*

Definition 1.1.4 (PPT Algorithm) *A Probabilistic Polynomial-time (PPT) algorithm is a probabilistic algorithm that is polynomial-time, that is, it terminates in at most $\mathscr{O}(n^c)$ steps, where n is the number of bits of the input and $c > 0$ is a costant.*

[1] We use the unary encoding 1^λ to make the security parameter part of the input in a way that its length is exactly λ, ensuring that the running time is measured as a function of λ.

1.2 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) was independently introduced in 1985 by Neal Koblitz and Victor S. Miller [2, 3]. Its widespread adoption came several years later, mainly due to the fact that it provides the same level of security as classical public-key cryptosystems over finite fields, while requiring significantly shorter key sizes. The most commonly used elliptic curves in cryptographic applications are those given in short Weierstrass form over a finite field.

Definition 1.2.1 (Short Weierstrass Form) *An elliptic curve E in short Weierstrass form over a finite field $\mathbb{F}_q$, with* $\text{char}(\mathbb{F}_q) \notin \{2,3\}$*, is defined as the set of points $(x,y) \in \mathbb{F}_q^2$ satisfying the equation*

$$y^2 = x^3 + ax + b,$$

where $a,b \in \mathbb{F}_q$ are such that $4a^3 + 27b^2 \neq 0$.

Denote with $E(\mathbb{F}_q)$ the set of points of the curve E on the finite field $\mathbb{F}_q$, that is

$$E(\mathbb{F}_q) = \{(x,y) \in \mathbb{F}_q^2 \mid y^2 = x^3 + ax + b\} \cup \{\mathscr{O}\}.$$

where $\mathscr{O}$ is the so-called *point at infinity* of the curve[2].

It is well known that $E(\mathbb{F}_q)$ forms an abelian group with respect to a specific addition law. Given two points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$, the group operation is defined as follows:

- If $P \neq \pm Q$, then $R = P + Q = (x_R, y_R)$, where

$$x_R = \lambda^2 - x_P - x_Q$$
$$y_R = \lambda(x_P - x_R) - y_P$$

  with $\lambda = \frac{y_Q - y_P}{x_Q - x_P}$.

[2]Throughout this section we assume $P, Q \neq \mathscr{O}$. The group law is extended by defining $P + \mathscr{O} = \mathscr{O} + P = P$ for all points $P \in E(\mathbb{F}_q)$.

- If $P = Q$, then $R = 2P = (x_R, y_R)$, where

$$x_R = \lambda^2 - 2x_P$$
$$y_R = \lambda(x_P - x_R) - y_P$$

  and $\lambda = \frac{3x_P^2 + a}{2y_P}$.

- If $x_P = x_Q$ and $y_Q = -y_P$, then $P + Q = \mathcal{O}$.

Scalar multiplication on elliptic curves can be efficiently computed using the Double-and-Add algorithm [4].

Key generation over elliptic curves. In cryptographic applications, elliptic curves are commonly used to instantiate public-key cryptosystems. Each user independently generates a key pair as follows. Let $G \in E(\mathbb{F}_q)$ be a publicly known generator of a large cyclic subgroup of prime order r. The private key is an integer $sk \in \mathbb{Z}_r$, chosen uniformly at random. The corresponding public key is computed as $\mathsf{pk} = \mathsf{sk}\,G$. The security of the system relies on the hardness of the Elliptic Curve Discrete Logarithm Problem (ECDLP), namely, given G and $\mathsf{pk} = \mathsf{sk}G$, it is computationally infeasible to recover the secret key sk.

1.3 Hash functions

This section introduces **hash functions** and their fundamental properties.

Definition 1.3.1 (Hash function) *Let Σ be an alphabet and let Σ^* be the set of all words (of arbitrary length) over Σ. A* hash function *$h : \Sigma^* \to \Sigma^n$, with input lenght $\ell_{in}(\lambda)$ and output lenght $\ell_{out}(\lambda) = n$, for some fixed integer n, is a pair of PPT algorithms (Gen, H) satisfying the following properties:*

- Gen *takes as input 1^λ and returns a key k*[3]*. We assume that λ is implicit in k.*
- *H is a deterministic algorithm that takes as input a key k and a string $x \in \{0,1\}^{\ell_{in}(\lambda)}$ and output a string $H(k,x) \in \{0,1\}^{\ell_{out}(\lambda)}$.*

[3]We remark that the key k has nothing to do with the secret key and the public key of a cryptographic scheme.

Most applications use the binary alphabet $\Sigma = \{0,1\}$ and output sizes such as $n = 160$, 256, or 512. The value $h(x)$ is called the *hash* or *digest* of x. We are only interested in the case in which $\ell_{\text{in}} > \ell_{\text{out}}$. In such a case, hash functions cannot be injective and are called *compression functions*.

For cryptographic use, a hash function should satisfy the following properties:

- *Avalanche effect*: changing a single bit in the input should change most bits of the output;
- *One-wayness*: given a digest $h(x)$, it should be computationally infeasible to recover the input x;
- *Collision resistance*: it should be computationally infeasible to find two different inputs a and b such that $h(a) = h(b)$;
- *Second preimage resistance*: for a given $a \in \Sigma^*$, it should be computationally infeasible to find a different $b \in \Sigma^*$ with $h(a) = h(b)$.

The *invert experiment* and *one-way functions* are defined as follows.

Experiment 1 ($\mathsf{Invert}_{\mathscr{A},f}(1^\lambda)$)

1. *Generate a random* $x \xleftarrow{\$} \{0,1\}^\lambda$.
2. *Compute* $y \leftarrow f(x)$.
3. *Give* 1^λ *and* y *to the adversary* $\mathscr{A}$.
4. *Let* $\mathscr{A}$ *output* x'.
5. *Return 1 if* $f(x') = y$*, and return 0 otherwise.*

Definition 1.3.2 (One-Way Function) *A function* $f : \{0,1\}^* \to \{0,1\}^*$ *is* one-way *if the following properties holds:*

- Easy to compute. *There exists a polynomial-time algorithm to compute* f.

- Hard to invert. *For every PPT algorithm $\mathscr{A}$ there exists a negligible*[4] *function* negl *such that*
$$\mathbb{P}[\mathsf{Invert}_{\mathscr{A},f}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda).$$

Definition 1.3.3 (Collision) *Let* (Gen, H) *be a hash function. A* collision *for a given key* $k \leftarrow \mathsf{Gen}(1^\lambda$ *of* H *is a pair* $x, x' \in \{0,1\}^{\ell_{in}}$ *such that* $x \neq x'$ *and* $H(k,x) = H(k,x')$.

Note that for compression functions we have $\ell_{\text{in}} > \ell_{\text{out}}$ and so collisions certainly exist.

Experiment 2 ($\mathsf{Hash} - \mathsf{coll}_{\mathscr{A},\mathscr{H}}(1^\lambda)$**)**

1. *A random key is generated* $k \leftarrow \mathsf{Gen}(1^\lambda)$.
2. *The adversary* $\mathscr{A}$ *is given* k.
3. *The adversary* $\mathscr{A}$ *output* $x, x' \in \{0,1\}^{\ell_{in}}$.
4. *Return 1 (success) if* $x \neq x'$ *and* $H(k,x) = H(k,x')$. *Otherwise, return 0 (failure).*

Definition 1.3.4 (Collision-Resistance Hash Function) *A hash function* $\mathscr{H} = (\mathsf{Gen}, H)$ *is* collision-resistant *if for all PPT adversaries* $\mathscr{A}$ *there exists a negligible function* negl *such that*
$$\mathbb{P}[\mathsf{Hash} - \mathsf{coll}_{\mathscr{A},\mathscr{H}}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

Cryptographically secure hash function (as SHA-3, BLAKE3, Whirlpool) are beliviied to be collision-resistant, but no proof of this claim is currently known.

Experiment 3 ($\mathsf{Second} - \mathsf{preimage}_{\mathscr{A},\mathscr{H}}(1^\lambda)$**)**

1. $k \leftarrow \mathsf{Gen}(1^\lambda), \quad x \xleftarrow{\$} \{0,1\}^{2\lambda}$
2. *The adversary* $\mathscr{A}$ *is given the couple* (k,x) *and outputs* x'.
3. *The adversary* $\mathscr{A}$ *win if* $H(k,x) = H(k,x'), \quad x \neq x'$

[4] A function $f : \mathbb{N} \to \mathbb{R}$ is *negligible* if, for every $c > 0$, there exists an integer $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, $f(n) = O(1/n^c)$.

Definition 1.3.5 (Second preimage resistant) *A hash function* $\mathscr{H} = (\mathsf{Gen}, H)$ *is* second preimage resistant *if for all PPT adversaries* $\mathscr{A}$*, there exists a negligible function* negl *such that*

$$\mathbb{P}[\mathsf{Second-preimage}_{\mathscr{A},\mathscr{H}}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

Numerous cryptographic hash functions have been proposed and widely adopted in practice. Notable examples include *RIPEMD160* [5], *SHA256* [6], and *Keccak256* [7].

1.4 Cryptographic assumptions

Cryptographic protocols are typically based on **cryptographic assumptions**, which formalize the idea that certain mathematical problems are infeasible to solve in polynomial time. This section presents three cryptographic assumptions on which the three contributions of this thesis are based.

Factoring assumption. The factoring problem consists in recovering the prime factors of a composite integer $N = pq$. This problem is believed to be computationally hard when N is the product of two large primes of comparable size, and no polynomial-time algorithm is currently known for general instances. The most efficient known classical algorithms include the General Number Field Sieve and its variants [8].

This assumption is used in Chapter 4 and underlies the security of the proposed time-lock puzzle construction. In particular, the security relies on the hardness of factoring the modulus n: if an adversary were able to factor n, they could efficiently compute $\varphi(n)$ and thus solve the puzzle significantly faster.

The ISIS_f assumption. In [9, 10] the authors propose a new family of lattice problems that admit a practically efficient zero-knowledge proof for proving knowledge of a solution. They show how to apply this zero-knowledge proof to create simple constructions of various efficient privacy-preserving lattice-based primitives, such as

anonymous credentials, based on the presumed hardness of the new problem. The definition is given below.

Definition 1.4.1 (The ISIS_f Problem (informal)) *We are given a matrix $A \in \mathbb{Z}_p^{n \times m}$ (chosen from some distribution), a function $f : [N] \to \mathbb{Z}_p^n$, and access to an oracle that samples a random input $x \in [N]$ and outputs it together with a vector $\vec{s} \in \mathbb{Z}^m$ such that $A\vec{s} = f(x)$. The game is won by producing a fresh pair $(x', \vec{s}') \in [N] \times \mathbb{Z}^m$ such that $\vec{s}'$ satisfies the required norm bound and $A\vec{s}' = f(x')$*[5].

The hardness of the above problem depends on the choice of f, the distribution of A and how the oracle chooses the vector $\vec{s}$. A simple way to instantiate the function $f(x)$ is to choose it to be linear. For example, if we use the natural correspondence between binary vectors of lenght $\log N$ with the set $[N]$, then if $f(\vec{v}) = B\vec{v}$, where $\vec{v} \in \{0,1\}^{\log N}$, one can use the efficient zero-knowledge proofs from [11] to commit to $\vec{s}$ and $\vec{v}$ and show that they have small norms[6] and satisfy $A\vec{s} = B\vec{v}$. This may appear problematic, since the linearity of f could interact with the linear structure of lattices. For example, if $\vec{v}_1 + \vec{v}_2 = \vec{v}_3$ and $A\vec{s}_i = f(\vec{v}_i)$ for $i = 1, 2$, then by linearity we have $A(\vec{s}_1 + \vec{s}_2) = f(\vec{v}_3)$ suggesting that one might combine valid pre-images to obtain new ones. However, in our setting the vectors $\vec{v}_i$ are sampled uniformly at random from a binary domain. Thus, the sum of two binary vectors is unlikely to remain binary (i.e. the probability is $(3/4)^{\log N}$). So, the hardness of ISIS_f relies in large part on the fact that the domain of f is not closed under addition. Moreover, when summing multiple binary vectors, the resulting pre-image $\vec{s}$, obtained as the sum of valid pre-images, quickly grows in norm, and thus the probability that it remains sufficiently small becomes negligible. Since the hardness of lattice-based problems relies on finding *short* pre-images, this prevents an adversary from exploiting linearity in a meaningful way. For these reasons, instantiating f as a linear map over some ring, while restricting its domain to binary vectors, yields a hard instance of ISIS_f.

This assumption is at the core of the implementation presented in Chapter 5, which implements the anonymous credential scheme proposed by the authors in [9, 10].

[5] Note that x' can be equal to one of the x, as long as $\vec{s} \neq \vec{s}'$.

[6] Since $\vec{v} \in \{0,1\}^{\log N}$, all its entries are either 0 or 1. Therefore, its Euclidean norm is bounded by $\|\mathbf{v}\| \leq \sqrt{\log N}$, which is small compared to the typical parameter sizes considered in lattice-based constructions.

q-Strong Diffie–Hellman assumption. The q-Strong Diffie–Hellman (q-SDH) assumption can be seen as a generalization of the discrete logarithm[7] assumption in bilinear pairing settings. We report the definition below.

Definition 1.4.2 (q-Strong Diffie–Hellman (q-SDH) assumption) *Let $(\mathbb{G}_1, \mathbb{G}_2)$ be cyclic groups of prime order p with generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, and let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be an efficiently computable bilinear pairing. The* q-Strong Diffie–Hellman (q-SDH) problem *in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as follows: Given the $(q+2)$-tuple*

$$(g_1, g_2, g_2^x, g_2^{x^2}, \ldots, g_2^{x^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$$

for some randomly chosen secret $x \in \mathbb{Z}_p^*$, compute a pair (A, c) such that

$$A^{x+c} = g_1 \quad \textit{where } A \in \mathbb{G}_1,\ c \in \mathbb{Z}_p^*.$$

We say that the (q, t, ε)-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no algorithm running in time t has advantage at least ε in solving the q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$.

.

This assumption is used in Chapter 6 to prove the unforgeability of the proposed delegation scheme instantiated with BBS anonymous credentials. In particular, the security relies on the strong unforgeability of the underlying BBS-based anonymous credential scheme, which in turn holds under the q-Strong Diffie–Hellman assumption.

1.5 The Random Oracle Model

We now introduce a widely used heuristic for modeling the behavior of hash functions in cryptographic constructions. The **Random Oracle Model (ROM)** is an idealized framework in which all parties, including potential adversaries, are granted access to a public oracle which behaves as a black box implementing a truly random function. Whenever the oracle is queried on an input that has never appeared before,

[7] The discrete logarithm problem, in its classical form, asks to compute x given g and $y = g^x$ in a cyclic group of prime order p. In elliptic curve settings, this corresponds to finding the scalar x given a point G and $P = xG$.

it returns an independently and uniformly chosen output; for repeated queries, it returns the same value consistently. Since such ideal oracles cannot exist in practice, cryptographic hash functions are typically used as their concrete instantiation.

The central idea[8] is to model a hash function $H : \mathscr{M} \to \mathscr{T}$ as a truly random function $\mathscr{O}$, chosen uniformly at random from the set of all functions mapping $\mathscr{M}$ to $\mathscr{T}$. The function $\mathscr{O}$ is referred to as a *random oracle*. In this setting, both the challenger and the adversary are granted oracle access to $\mathscr{O}$, meaning that they may query it on arbitrary inputs and obtain the corresponding outputs. The ROM is particularly useful for proving the security of cryptographic schemes. Security in the random oracle model is defined by modifying the standard security game associated with a given cryptographic scheme. Specifically, every evaluation of the hash function H within the scheme is replaced by a query to the random oracle $\mathscr{O}$. The adversary is also allowed to query $\mathscr{O}$ adaptively on inputs of its choice. A scheme is considered secure in the ROM if any efficient adversary has only negligible advantage in the resulting game.

It is important to emphasize that the random oracle $\mathscr{O}$ is an idealized object: it cannot be explicitly constructed or efficiently implemented in practice, as it corresponds to a randomly chosen function from an exponentially large function space. Instead, the ROM serves as a tool for heuristic security analysis of schemes that, in practice, instantiate $\mathscr{O}$ with a concrete hash function H. Although security proofs in the random oracle model are mathematically rigorous, they do not guarantee security when a specific hash function is used. Nevertheless, such proofs are widely regarded as strong evidence of security in practice, under the assumption that well-designed cryptographic hash functions approximate the behavior of random oracles. The underlying idea is that, if an adversary $\mathscr{A}$ can break a scheme proven secure in the ROM (where hash functions are treated as perfect random oracles) then no real hash function used in practice would be sufficient for that scheme. For this reason, a security proof done under this model gives some trust, since it is assumed that cryptographic hash functions are secure if implemented correctly. The strength of the ROM lies in its ability to simplify proofs that would otherwise be extremely complex or even intractable in the standard model. Many primitives, including practical signature schemes, encryption mechanisms, and commitment protocols, derive their security guarantees from proofs carried out in this idealized setting. Of course, real

[8] See D. Boneh and V. Shoup, *A Graduate Course in Cryptography* [12], p. 327 for details.

hash functions can never perfectly reproduce a truly random function, and there are known examples of constructions that are secure in the ROM but become insecure when instantiated with concrete hash functions. Nonetheless, proofs in the ROM generally provide meaningful assurance and are widely regarded as strong evidence of the practical security of a scheme.

1.6 Digital signatures

Digital signatures are cryptographic primitives that ensure the authenticity and integrity of messages. They allow a sender to produce a signature on a message such that any verifier, given the sender's public key, can check its validity (see [12], p. 528 for more details). In a digital signature scheme, the signing algorithm uses a *secret signing key* sk, while the verification algorithm uses the associated *public verification key* pk.

Definition 1.6.1 (Digital Signature Scheme) *A* digital signature scheme $\mathscr{DS}$ *consists of four probabilistic polynomial time (PPT) algorithms* $= (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vf})$ *such that:*

- Setup $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*: it takes as input* 1^λ *and outputs the public parameters* pp *used by the scheme.*
- Key Generation $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*: it takes as input the public parameters* pp *and outputs a pair* $(\mathsf{pk}, \mathsf{sk})$*, where* sk *is the secret key, and* pk *is the public key.*
- Signing $\sigma \leftarrow \mathsf{Sign}(M, \mathsf{sk})$*: a probabilistic algorithm that, given the secret key of the signer* sk *and a message M, outputs a* signature σ.
- Verification $(\mathsf{accept}/\mathsf{reject}) \leftarrow \mathsf{Vf}(\sigma, M, \mathsf{pk})$*: a deterministic algorithm that, given the public key* pk*,the message M, and the signature* σ*, outputs 1* (accept) *if the signature is valid and 0* (reject) *otherwise.*

We now introduce the main security properties of digital signature schemes, namely *correctness* and *unforgeability*.

Definition 1.6.2 (Correctness) *A digital signature is correct if for all key pairs* $(\mathsf{sk},\mathsf{pk})$ *output by* KeyGen*, and for all messages m, any signature* σ *generated by* Sign *must be accepted by* Vf*. Formally,*

$$\mathbf{P}\left[\mathsf{Vf}(\mathsf{pk},M,\sigma)=1 \;\middle|\; \begin{array}{c}\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda),\\ (\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{KGen}(\mathsf{pp}),\\ \sigma\leftarrow\mathsf{Sign}(\mathsf{sk},M).\end{array}\right]=1.$$

This means that the verification algorithm Vf *of a signature* σ *generated correctly will always output 1.*

In the definition of a secure signature scheme we give the adversary the power to request the signature of any message of his choice. Even with such power, the adversary should not be able to create a forgery, namely the adversary cannot output a valid message-signature pair (M,σ) for some new message M.

Definition 1.6.3 (Unforgeability) *The notion of* unforgeability *for digital signature schemes is defined via the following experiment between a challenger and an adversary* $\mathscr{A}$*:*

Setup: *The challenger runs the key generation algorithm* KeyGen *to obtain a public key* pk *and a private key* sk*. The adversary is given* pk*.*

Queries: *The adversary requests signatures on at most* q_S *messages of its choice* $M_1,\ldots,M_{q_S}\in\{0,1\}^*$ *under* pk*. The challenger responds to each query with a signature* $\sigma_i\leftarrow\mathsf{Sign}(\mathsf{sk},M_i)$*.*

Output: *The adversary outputs a pair* (M,σ) *and wins the game if:*

1. M *is not equal to any of* $M_1,\ldots,M_{q_S}$*;*
2. $\mathsf{Vf}(\sigma,M,\mathsf{pk})=1$*.*

We denote by $\mathsf{SigAdv}_{\mathscr{A}}$ *the probability that adversary* $\mathscr{A}$ *wins the above experiment, over the randomness of both* $\mathscr{A}$ *and the challenger.*

Definition 1.6.4 (Unforgeable signature scheme) *A signature scheme* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vf})$ *is said to be* unforgeable *if, for all probabilistic polynomial-time adversaries* $\mathscr{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\mathsf{SigAdv}_{\mathscr{A}} \leq \mathsf{negl}(\lambda),$$

where λ *is the security parameter.*

Digital signatures that satisfy this property are said to be unforgeable under a chosen message attack *(UF-CMA security).*

In Chapter 6, we adopt a more stringent notion of unforgeability, namely *strong unforgeability*, to prove the unforgeability of our delegation scheme, under the assumption that the BBS anonymous credentials scheme is strongly unforgeable. The definition and the relative experiment are defined below.

Definition 1.6.5 (Strong Unforgeability) *The notion of* strong unforgeability *for digital signature schemes [13] is defined via the following experiment between a challenger and an adversary* $\mathscr{A}$*:*

Setup: *The challenger runs the key generation algorithm* KeyGen *to obtain a public key* pk *and a private key* sk*. The adversary is given* pk*.*

Queries: *The adversary requests signatures on at most* q_S *messages of its choice* $M_1, \ldots, M_{q_S} \in \{0,1\}^*$ *under* pk*. The challenger responds to each query with a signature* $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}, M_i)$*.*

Output: *The adversary outputs a pair* (M, σ) *and wins the experiment if:*

1. (M, σ) *is not equal to any of* $(M_1, \sigma_1), \ldots, (M_{q_S}, \sigma_{q_S})$*;*
2. $\mathsf{Vf}(\sigma, M, \mathsf{pk}) = 1$.

We denote by $\mathsf{StrongSigAdv}_{\mathscr{A}}$ *the probability that adversary* $\mathscr{A}$ *wins the above experiment, over the randomness of both* $\mathscr{A}$ *and the challenger.*

Definition 1.6.6 (Strongly unforgeable signature scheme) *A signature scheme* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{V}$ *is said to be* strongly unforgeable *if, for all probabilistic polynomial-time adversaries*

$\mathscr{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ *such that*

$$\mathsf{StrongSigAdv}_{\mathscr{A}} \leq \mathsf{negl}(\lambda),$$

where λ *is the security parameter.*

Digital signatures that satisfy this property are said to be strongly unforgeable under a chosen message attack *(SUF-CMA security).*

After introducing digital signatures, we now consider a concrete instantiation, namely the BBS signature scheme, which is used in Chapter 6 as one of the two credentials framework on which our delegation scheme is built.

1.6.1 The BBS signature scheme

The **BBS signature scheme**[9] were implicitly introduced by Boneh, Boyen, and Shacham [15] as part of their group signature construction, and are of theoretical interest due to their simplicity and efficiency. In [17], Tessaro and Zhu proved that BBS signatures are secure under the q-Strong Diffie–Hellman assumption. They also devised simplified and shorter zero-knowledge proofs of knowledge of a BBS message-signature pair that support partial disclosure of the message. We now present the signature scheme below.

Let $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$, and $\mathbb{G}_T$ be groups of prime order p. A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a map that satisfies the following properties:

- Bilinearity: $\mathbf{e}(g_1^x, g_2^y) = \mathbf{e}(g_1, g_2)^{xy} \quad \forall x, y \in \mathbb{F}_q^*$.
- Non-degenerate: $\mathbf{e}(g_1, g_2) \neq 1$.
- Efficient computability: the pairing $\mathbf{e}$ can be computed in polynomial time with respect to the bit length of the security parameter λ.

The BBS signature scheme is defined by the following algorithms:

[9] Most modern applications consider the provably secure variant known as *BBS+ signatures*. The BBS+ scheme was later formalized in [14], building on ideas from [15, 16], and is proven secure under the q-Strong Diffie–Hellman assumption.

- Setup: Given $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, and the pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ as defined above, sample uniformly at random $(g_1, g_2, \mathbf{h} = (h_i)_{i\in[l]}) \xleftarrow{\$} \mathbb{G}_1^{l+2}$.

- KeyGen: Sample uniformly at random $x \xleftarrow{\$} \mathbb{Z}_p$, compute $X = g_2^x \in \mathbb{G}_2$, and set $\mathsf{sk} = x$ and $\mathsf{pk} = X$.

- Sign: On input messages $\mathbf{a} = (a_i)_{i\in[l]}$, sample uniformly at random $e \xleftarrow{\$} \mathbb{Z}_p$ and compute

$$A = \left(g_1 \prod_{i=1}^{l} \mathbf{h}[i]^{\mathbf{a}[i]} \right)^{\frac{1}{x+e}}.$$

  Output the signature $\sigma = (A, e)$.

- Vf: on input a public key $X \in \mathsf{G}_2$, the messages $\mathbf{a} = (a_i)_{i\in[l]}$, and a signature $\sigma = (A, e)$, compute $C(\mathbf{a}) = g_1 \prod_{i=1}^{l} \mathbf{h}[i]^{\mathbf{a}[i]}$ and check if

$$\mathbf{e}(A, X g_2^e) = \mathbf{e}(C(\mathbf{a}), g_2).$$

  Return 1 if the check is correct, and 0 otherwise.

The most efficient implementations of the BBS signature are based on the curve BLS 12-381 [18] which defines a pairing such that there is not an efficient isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$, and where computations in $\mathbb{G}_1$ are more efficient than the computations in $\mathbb{G}_2$.

1.7 Commitment schemes

A **commitment scheme** is a cryptographic primitive that has several applications. A commitment scheme $\mathscr{C}$ enables a party, Alice, to commit to a message $M \in \mathscr{M}$ by producing a commitment string c. At a later stage, Alice can open the commitment and convince another party, Bob, that the committed message is indeed M.

Definition 1.7.1 (Commitment Scheme) *A* commitment scheme Π_{Com} *consists of three PPT algorithms* $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *and operates as follows:*

- $\mathsf{Setup}(1^\lambda)$ *takes as input* 1^λ *and outputs the public parameters* pp.

- $\mathsf{Com}(\mathsf{pp}, M; r)$ *takes as input the public parameters* pp*, a message* $M \in \{0,1\}^{\lambda}$*, a randomness* r *and returns as output the commitment* $\mathsf{com} = \mathsf{Com}(\mathsf{pp}, M; r)$.
- $\mathsf{Open}(\mathsf{pp}, M, \mathsf{com}; r)$ *takes as input the public parameters* pp*, the message* M*, the commitment* com *and the randomness* r*; it returns* accept *if* com *is the commitment of* M *or* reject *otherwise.*

A commitment scheme is considered secure if it satisfies the properties of *hiding* and *binding*[10].

Definition 1.7.2 (Hiding) *Let* $\Pi_{\mathsf{Com}} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *be a commitment scheme and let hiding*(Π_{Com}) *be the hiding game represented below.*

Adversary $\mathscr{A}$		***Challenger*** $\mathscr{C}$
	$\xleftarrow{pp}$	$pp \leftarrow \mathsf{Gen}(1^{\lambda})$
Choose $M_0, M_1 \in \{0,1\}^{\lambda}$	$\xrightarrow{M_0, M_1}$	$b \xleftarrow{\$} \{0,1\}$
	$\xleftarrow{\mathsf{com}}$	$\mathsf{com} \leftarrow \mathsf{Com}(pp, M_b; r)$
Guess b'	$\xrightarrow{b'}$	$\mathscr{A}$ *wins if* $b' = b$

A commitment scheme Π_{Com} *is* computationally hiding *if, for all* PPT *adversaries* $\mathscr{A}$*, there is a negligible function* $\mathsf{negl}(\lambda)$*, with* λ *being the security parameter, such that*

$$\mathbf{P}[\mathscr{A} \text{ wins } hiding(\Pi_{\mathsf{Com}})] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

If, for every pair M_0, M_1*, the commitments* com_0 *and* com_1 *have the same distribution, then the commitment is said to be* perfectly hiding.

Definition 1.7.3 (Binding) *A commitment scheme* $\Pi_{\mathsf{Com}} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is* computationally binding *if, for all* PPT *adversaries* $\mathscr{A}$*, there is a negligible function* $\mathsf{negl}(\lambda)$*, with* λ *being the security parameter, such that*

[10] For further detalis on the hiding and binding security experiments, see [19].

$$\mathbf{P}\left[\begin{array}{l|c} & \mathsf{pp} \leftarrow \mathsf{Pgen}(1^\lambda), \\ (\mathsf{com}, M_0, r_0, M_1, r_1) \leftarrow \mathscr{A}(\mathsf{pp}) & M_0 \neq M_1, \\ & \mathsf{Open}(M_0, \mathsf{com}, r_0) = \textbf{accept}, \\ & \mathsf{Open}(M_1, \mathsf{com}, r_1) = \textbf{accept} \end{array}\right]$$

is negligible in the security parameter λ*. This means that a commitment scheme is binding if it is computationally infeasible for the committer to generate a commitment* com *that can be opened to two distinct messages* M_0 *and* M_1*.*

If $\mathsf{negl}(\lambda) = 0$*, then the commitment scheme is said to be* perfectly binding.

A secure commitment scheme can be easily constructed using a cryptographic hash function H: to commit to a message M, sample a random $\mathsf{salt} \xleftarrow{\$} \{0,1\}^\lambda$ and compute $\mathsf{com} = H(M||\mathsf{salt})$. To open a commitment com, it is necessary to reveal the message-salt pair (M, salt), and the verifier can check if $\mathsf{com} = H(M||\mathsf{salt})$ (equivalently, $\mathsf{Open}(M, \mathsf{com}; \mathsf{salt}) = \mathsf{accept}$). This commitment scheme can be proved to be computationally hiding and binding[11].

One of the most well-known commitment schemes is the Pedersen commitment [20], which is perfectly hiding and computationally binding under the discrete logarithm assumption.

1.8 Merkle trees

A **Merkle tree** [21] is a fundamental cryptographic data structure used to efficiently commit to a large dataset and to support succinct membership proofs. It is defined as a binary tree where each leaf node contains the cryptographic hash of a data element. Each internal node is computed as the hash of the concatenation of the values of its two children, using a fixed and publicly known hash function. The value associated

[11] To ensure that this commitment is computationally binding, we require the hash function H to be collision resistant. It is easy to see that this assumption is sufficient to guarantee the binding property of the commitment scheme. Indeed, any efficient adversary $\mathscr{A}$ that breaks binding game can be used to construct a collision for H. Suppose $\mathscr{A}$ outputs two distinct messages $M_1 \neq M_2$ together with salts salt_1 and salt_2 such that, $\mathsf{Open}(M_1, c, \mathsf{salt}_1) = \mathsf{Open}(M_2, c, \mathsf{salt}_2) = \text{accept}$. This implies $H(M_1 \,\|\, \mathsf{salt}_1) = c = H(M_2 \,\|\, \mathsf{salt}_2)$, which yields a collision for H.

with the root of the tree, called the *Merkle root*, serves as a compact commitment C to the entire dataset.

Let's consider a complete balanced[12] Merkle tree with depth d, so that the number of leaves is $L = 2^d$. To prove that a data element D_i belongs to the committed set, the prover provides an *authentication path*. This path consists of the hash values corresponding to the nodes needed to reconstruct the path from the leaf associated with D_i up to the root. The verifier recomputes the hash values starting from the leaf and using the values in the authentication path, and accepts the proof if and only if the resulting root matches the commitment C. Both the size of the proof and the verification time are $\mathcal{O}(\log L)$. The structure of a Merkle tree is illustrated in Figure 2.2.

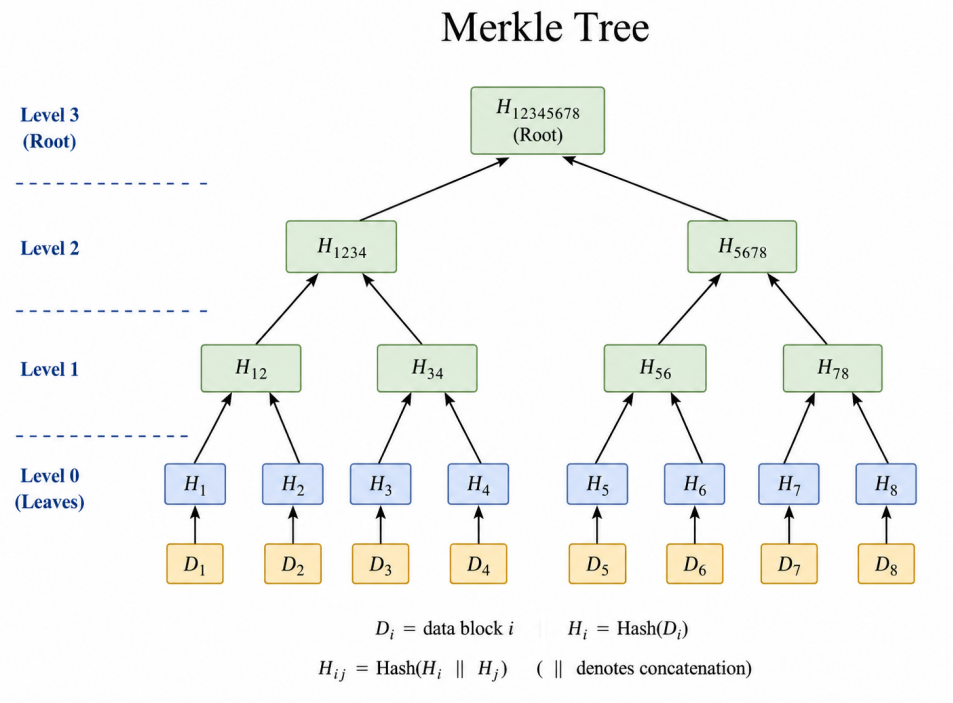

Fig. 1.1 Merkle tree structure. This is an example of complete balanced Merkle tree with depth $d = 3$.

It can be easily shown that, if the underlying hash function is collision-resistant, then the resulting commitment is computationally binding. Furthermore, when the hash is computed by incorporating fresh randomness (e.g., as a random nonce

[12] A *complete balanced Merkle tree* is a Merkle tree in which every internal node has exactly two children.

concatenated with the message), and such randomness is kept secret, the scheme achieves both computational binding and computational hiding. Merkle trees are used by blockchains to store in the header of a block the root of the hash of all transactions (see Chapter 2, Section 2.3).

1.9 Sigma protocols

Sigma protocols are a class of interactive protocols that allow a prover to convince a verifier of the validity of a public statement while satisfying specific security properties.

Let NP be the class of decision problems that can be efficiently verified.

Definition 1.9.1 (NP relation) *For each $L \in \mathsf{NP}$, we define the relation*

$$\mathscr{R}(L) = \{(x,y) \mid y \in L \text{ and } x \text{ is a witness for } y\}.$$

A relation $\mathscr{R}$ is an NP-relation *if there exists a language $L \in \mathsf{NP}$ such that $\mathscr{R} = \mathscr{R}(L)$.*

Definition 1.9.2 (Sigma protocol) *Let $\mathscr{R} \subseteq \mathscr{X} \times \mathscr{Y}$ be a* NP *relation. A* Sigma protocol *for $\mathscr{R}$ is a three-move interactive protocol between two PPT machines, a prover $\mathsf{P} = (\mathsf{P}_{\mathsf{com}}, \mathsf{P}_{\mathsf{resp}})$ and a verifier* V*. The interaction proceeds as follows:*

1. *The prover* P *takes as input a pair $(x,y) \in \mathscr{R}$ (where x is the witness and y is the associated statement) and generates a* commitment $\mathsf{com} \in \mathscr{T}$ *and sends it to the verifier.*
2. *The verifier*[13] V *chooses a random* challenge ch *from a finite challenge space $\mathscr{C}$ and sends it to the prover.*
3. *The prover* P *computes a* response $\mathsf{resp} \in \mathscr{Z}$*, and sends it back to the verifier.*
4. *The verifier uses* transcript $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ *and the statement y to calculate a deterministic decision*[14]*:* accept *or* reject.

[13] Note that the verifier's behavior is deterministic, except for the random choice of the challenge.
[14] The output of V is assumed to be in $\{0,1\}$.

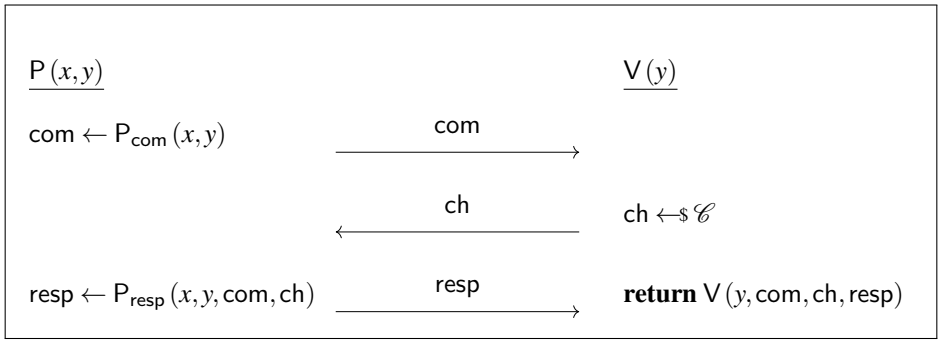

Fig. 1.2 Generic Sigma Protocol.

A sigma protocol must satisfy three well-known properties: *correctness*, *special soundness* and *special honest-verifier zero-knowledge* (see [22] for further details).

Definition 1.9.3 (Completeness) *A sigma protocol satisfy* completeness *if, for all* $(x,w) \in \mathscr{R}$*, then*

$$\mathbf{P}\left[\mathsf{V}(y,\mathsf{com},\mathsf{ch},\mathsf{resp}) = 1 \;\middle|\; \begin{array}{c} \mathsf{com} \leftarrow \mathsf{P}_{\mathsf{com}}(x,y), \\ \mathsf{ch} \leftarrow\!\!\$\, \mathscr{C}, \\ \mathsf{resp} \leftarrow \mathsf{P}_{\mathsf{resp}}(x,y,\mathsf{com},\mathsf{ch}) \end{array}\right] = 1.$$

Definition 1.9.4 (Special 2-soundness) *A Sigma protocol satisfies* special 2-soundness *if there exists a polynomial-time algorithm* $\mathscr{E}$*, called an extractor, such that from any* y *and any pair of accepting transcripts* $(\mathsf{com},\mathsf{ch}_1,\mathsf{resp}_1)$ *and* $(\mathsf{com},\mathsf{ch}_2,\mathsf{resp}_2)$ *with* $\mathsf{ch}_1 \neq \mathsf{ch}_2$*, it always outputs* x *such that* $(x,y) \in \mathscr{R}$*.*

Definition 1.9.5 (Special honest-verifier zero-knowledge) *A sigma protocol satisfy* special honest-verifier zero-knowledge *if there exists an efficient probabilistic algorithm* $\mathscr{S}$ *(called a* simulator*) which, on input* $(y,\mathsf{ch}) \in \mathscr{Y} \times \mathscr{C}$ *outputs an accepting transcript of the form* $(\mathsf{com},\mathsf{ch},\mathsf{resp})$ *whose distribution is identical to that of transcripts produced between* $\mathsf{P}(x,y)$ *and* $\mathsf{V}(y)$*.*

A classical example of a sigma protocol is the Schnorr Identification scheme [23]. Sigma protocols are an essential building block for many applications such as blind signatures and anonymous credentials. For example, the Camenisch-Lysyanskaya (CL) anonymous credential scheme [24] employs zero-knowledge proofs based on Sigma protocols.

1.10 The Fiat-Shamir transform

The **Fiat-Shamir transform** is a technique used to convert a sigma protocol into a digital signature scheme. It was introduced by Amos Fiat and Adi Shamir in 1986 [25]. The key idea behind the Fiat-Shamir transform is to replace the verifier's generation of the challenge with the output of a hash function. As a result, the signer can generate a signature that can be verified by anyone who knows the public parameters of the protocol. In this way, an interactive protocol is transformed into a non-interactive one using the Fiat-Shamir transform. It is important to note that this transform can be applied only to *public-coin* protocols, namely protocols in which the verifier's randomness is publicly known by the prover in the interactive version of the protocol.

The general technique makes use of the following building blocks:

- a sigma protocol[15] (P,V) for a relation $\mathscr{R} \subseteq \mathscr{X} \times \mathscr{Y}$
- a key generation algorithm for $\mathscr{R}$
- a hash function $H : \mathscr{M} \times \mathscr{T} \to \mathscr{C}$ which will be modeled as a random oracle. The set $\mathscr{M}$ will be the message space of the signature scheme.

The Fiat-Shamir signature scheme derived from (P,V) works as follows:

- The key generation algorithm generates a public key $\mathsf{pk} = y \in \mathscr{Y}$ and a secret key $\mathsf{sk} = x \in \mathscr{X}$.
- To sign a message $M \in \mathscr{M}$ using a secret key sk, the prover P:
  1. Given the input (x, y), computes a commitment $\mathsf{com} \in \mathscr{T}$.
  2. Computes a challenge $\mathsf{ch} \leftarrow H(M, \mathsf{com})$.
  3. Computes a response resp, and outputs the signature $\sigma = (\mathsf{com}, \mathsf{resp}) \in \mathscr{T} \times \mathscr{Z}$.
- To verify a signature $\sigma = (\mathsf{com}, \mathsf{resp}) \in \mathscr{T} \times \mathscr{Z}$ on a message $M \in \mathscr{M}$ using a public key $\mathsf{pk} = y$, the verifier V computes $\mathsf{ch} \leftarrow H(M, \mathsf{com})$ and checks that $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ is an accepting transcript for y.

[15] We assume that transcripts are of the form $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$.

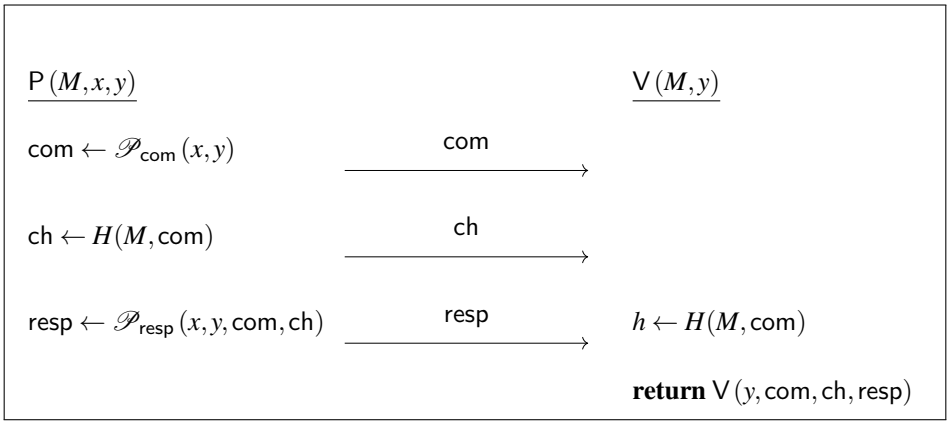

Fig. 1.3 The Fiat-Shamir transform used to convert a sigma protocol into a digital signature scheme.

1.11 Non-Interactive Zero-Knowledge Proofs

In this section, we first provide the definition of **Non-Interactive Zero-Knowledge Proofs (NIZKPs)** and then show how to transform a sigma protocol into a NIZKP using the Fiat-Shamir transform.

Definition 1.11.1 (Non Interactive Zero-Knowledge Proof) *Let $\mathscr{R} \subseteq \mathscr{X} \times \mathscr{Y}$ be an* NP *relation. A* Non Interactive Zero-Knowledge Proof *(NIZKP) for $\mathscr{R}$ consists of two algorithms* (P, V)*, where* P *has in input a witness-statement pair $(x,y) \in \mathscr{R}$, while* V *only has in input the statement y. The prover algorithm outputs a proof $\pi \leftarrow \mathsf{P}(x,y)$ which is sent to the verifier who can verify it running* $\mathsf{reject}/\mathsf{accept} \leftarrow \mathsf{V}(y,\pi)$.

The algorithm must satisfy the following properties:

1. Completeness*: For $(x,y) \in \mathscr{R}$,* $\mathsf{V}(\mathsf{P}(x,y),y) = 1$*;*
2. Zero-Knowledge*: For any PPT distinguisher $\mathscr{D}$, there exists a PPT simulator* Sim *such that, for any statement y chosen by $\mathscr{D}$, the following two distributions are computationally indistinguishable:*

$$\{\pi \leftarrow \mathsf{P}(x,y) : (x,y) \in \mathscr{R}\} \approx_c \{\pi' \leftarrow \mathsf{Sim}(y)\}.$$

In other words, even without knowing the witness x, the simulator can generate a proof π' for y that is indistinguishable from a proof π honestly generated by a prover who knows x;

3. Knowledge Soundness*: There exists a probabilistic polynomial-time (PPT) algorithm* Ext*, called the* extractor*, such that for any PPT adversary $\mathscr{A}$, if $\mathscr{A}$ can produce a valid proof π for some statement y with non-negligible probability, then given y and rewindable oracle access to $\mathscr{A}$, the extractor* Ext *can extract a witness x such that $(x,y) \in \mathscr{R}$ with non-negligible probability.*

The Fiat-Shamir transform (see Section 1.10) can also be used to convert a sigma protocol into a non-interactive zero-knowledge proof. The basic idea is very simple: instead of relying on a verifier to generate a random challenge ch, we use a hash function H to derive the challenge from the statement y and the commitment com.

Definition 1.11.2 (Fiat-Shamir NIZKP) *Let (P,V) be a sigma protocol for a* NP *relation $\mathscr{R} \subseteq \mathscr{X} \times \mathscr{Y}$. Assume that the transcripts $(\mathsf{com},\mathsf{ch},\mathsf{resp})$ belong to $\mathscr{T} \times \mathscr{C} \times \mathscr{Z}$. Let $H : \mathscr{Y} \times \mathscr{T} \to \mathscr{C}$ be a cryptographic hash function. The* Fiat-Shamir transform *converts (P,V) into a non-interactive zero-knowledge proof as follows:*

1. *Given an input $(x,y) \in \mathscr{R}$, the prover generates a commitment $\mathsf{com} \in \mathscr{T}$;*
2. *The prover computes the challenge as $\mathsf{ch} = H(y,\mathsf{com})$, then they compute a response $\mathsf{resp} \in \mathscr{Z}$; finally, the prover sends $(\mathsf{com},\mathsf{resp}) \in \mathscr{T} \times \mathscr{Z}$ to the verifier;*
3. *Given $(y,(\mathsf{com},\mathsf{resp})) \in \mathscr{Y} \times (\mathscr{T} \times \mathscr{Z})$, the verifier computes $\mathsf{ch} = H(y,\mathsf{com})$ and checks whether $(\mathsf{com},\mathsf{ch},\mathsf{resp})$ is an accepting transcript for y.*

In certain settings, the challenge ch can be sent to V instead of the commitment com (or part of it), provided that the protocol admits a deterministic reconstruction of the commitment.

1.12 Time-Release Cryptography and Time-Lock Puzzles

Time-release cryptography addresses the problem of encrypting information such that it can only be accessed after a predetermined time. The central goal is, in a sense, to "send information into the future," ensuring that no party (including the sender[16]) can decrypt the message before the intended time. Several practical applications motivate this concept. For instance, in sealed-bid auctions, participants may wish to submit bids that remain confidential until the bidding period has closed. Other examples include the long-term protection of sensitive personal data (such as diaries intended for release decades later) and key-escrow systems, where authorities are granted access to decryption keys only after a legally mandated delay.

Two main approaches have been proposed to implement timed-release cryptography:

- The first relies on **trusted agents** who promise not to reveal certain information until a specified date. While conceptually simple, this approach introduces the challenge of ensuring the trustworthiness and reliability of these agents, often requiring additional mechanisms such as secret sharing to mitigate risks.
- The second approach is based on computational hardness assumptions and uses **Time-Lock Puzzles**. In this paradigm, a message is encrypted using a key that is itself embedded in a computational puzzle designed to require a specific amount of time to solve. A limitation of time-lock puzzles is that the time required to solve them depends on the computational resources available, including the type of hardware and the degree to which the underlying problem can be parallelized. Usually, a time-lock puzzle is used to encrypt a secret sk (for instance a key of a symmetric cryptosystem) so that it could be recovered only after a fixed amount of time T.

The primary difficulty of using time-lock puzzles, as highlighted above, is that adversaries with substantial computational power may gain an advantage in solving them, for instance by leveraging large-scale parallel computing systems. For this

[16]The reason why even the sender cannot recover the secret before the intended time is that, by design, the construction of the time-lock puzzle requires the generator to discard the trapdoor information used to create it.

reason, the main goal of Rivest et al. [26] was to design time-lock puzzles that are *intrinsically sequential* in nature and cannot be solved substantially faster through large investments in hardware[17].

Despite this elegant construction, there are two main limitations:

- First, the correspondence between computational time and real-world time is only approximate, as different hardware platforms may execute computations at different speeds. If precise timing of the information release is essential, an approach based on the use of trusted agents is preferable.
- Second, the puzzle does not automatically become solvable at a predetermined point in time. Instead, it requires continuous computation until the solution to the puzzle is obtained. For example, a puzzle designed to take ten years requires a dedicated machine working for the entire duration. If computation begins only five years after the puzzle is generated, the solution will be obtained approximately ten years after that starting point (potentially slightly earlier if computational capabilities have improved in the meantime).

For these reasons, time-lock puzzles are often better suited to scenarios requiring moderate delays (on the order of months or weeks) rather than precise long-term scheduling. Nevertheless, they provide a powerful cryptographic primitive that eliminates the need for trusted intermediaries, offering a purely computational approach to timed-release cryptography. In the following section, we review the time-lock puzzle construction proposed by Rivest, Shamir, and Wagner [26], focusing on its design and the mechanism that enforces sequential computation.

1.12.1 The time-lock puzzle construction by Rivest, Shamir, and Wagner

The time-lock puzzle proposed by Rivest, Shamir, and Wagner in [26] is simple yet effective and exploits repeated squarings. The authors propose the following construction.

[17] Meaning that no shortcut or parallelization can reduce the time needed to compute the solution.

Creating the time-lock puzzle. Suppose Alice wants to encrypt a message M using a time-lock puzzle with a delay of T seconds. She constructs the time-lock puzzle as follows:

- She generates a composite modulus $n = pq$, where p and q are large randomly chosen secret primes.
- She computes the Euler totient function $\varphi(n) = (p-1)(q-1)$.
- She sets the number of required squarings as

$$t = TS,$$

  where S denotes the number of modular squarings per second that can be performed by the solver[18], and T is the desired delay in seconds.
- She generates a random symmetric key[19] sk for a conventional cryptosystem such as RC5.
- She encrypts the message M using sk, obtaining the ciphertext

$$C_M = \mathrm{RC5}(\mathsf{sk}, M).$$

- She selects a random value $a \in \mathbb{Z}_n^*$ and encrypts the key sk as

$$C_{\mathsf{sk}} = \mathsf{sk} + a^{2^t} \bmod n.$$

- To compute this efficiently, she first evaluates

$$e = 2^t \bmod \varphi(n),$$

  and then computes

$$b = a^e \bmod n.$$

[18] Clearly S strongly depends on the processor used.

[19] This key is sufficiently long (e.g., 160 bits or more, as suggested in the original construction [27]) so that exhaustive search remains computationally infeasible, even under significant advances in computing power expected over the lifetime of the puzzle. While modern cryptographic practice typically adopts 128-bit or 256-bit keys, the underlying assumption of brute-force infeasibility remains valid.

- Finally, she outputs the time-lock puzzle $(n, a, t, C_{\mathsf{sk}}, C_M)$ and erases all auxiliary values, including the secret primes p and q. The solver can then begin the sequential computation required to recover sk and decrypt C_M.

Solving the time-lock puzzle. By design, directly recovering the symmetric key sk through exhaustive search is computationally infeasible. Therefore, the most efficient known approach to solving the puzzle consists in computing

$$b = a^{2^t} \bmod n.$$

If the Euler totient $\varphi(n)$ were known, the exponent 2^t could be efficiently reduced modulo $\varphi(n)$, allowing b to be computed much more efficiently. However, computing $\varphi(n)$ from n is computationally equivalent to factoring n. Consequently, once Alice publishes the puzzle and discards the secret primes p and q, there is no known shortcut to compute b other than performing t sequential squarings, starting from a. Although factoring n provides an alternative method to solve the puzzle, this approach is significantly less efficient when p and q are chosen sufficiently large. The number t of squarings required to solve the puzzle can be exactly controlled. This allows to calibrate the difficulty of the puzzle according to the desired time delay. Thus, repeated squaring appears to be an inherently sequential process, for which no substantial parallelization is known. While limited parallelism may be achievable within individual operations, the overall computation must proceed sequentially. As a result, having access to multiple processors does not significantly reduce the solving time; instead, performance primarily depends on the speed of a single processor. Therefore, the time required to solve the puzzle depends mainly on the computational power of individual machines rather than on large-scale parallel resources. Since the performance gap between consumer hardware and specialized high-end systems is typically bounded, the difference in time to solution is reasonably controllable.

Chapter 2

Introduction to Blockchain

This chapter provides an overview of the fundamental concepts underlying blockchain systems. Section 2.1 highlights the main differences between distributed ledgers and traditional centralized databases. Section 2.2 introduces the structure and the main typologies of blockchain systems, outlining their key design characteristics and classification criteria. In Section 2.3, the internal structure of blockchain blocks is analyzed. Section 2.4 discusses consensus mechanisms, describing their role in ensuring agreement among distributed nodes and comparing the most widely adopted approaches: PoW and PoS. Section 2.5 examines blockchain forks, distinguishing between the main types of forks and discussing their causes, implications, and impact on network evolution. Section 2.6 introduces Cross Chain Communication and swap protocols to enable transactions between different blockchains. Finally, Section 2.7 outlines the key limitations of current blockchain systems particularly in terms of performance and scalability.

2.1 DLT and centralized database

In the following section, we first compare the most relevant properties of **Distributed Ledger Technology (DLT)** with those of a generic centralized system. Secondly, we focus on the basic concepts of blockchain, the actors involved, and its main types and properties.

DLT refers to a decentralized digital system that records and verifies transactions across multiple nodes within a network. Unlike traditional centralized systems,

where a single authority or entity controls the ledger, DLT operates on a distributed network, making it highly secure, transparent, and resistant to tampering. We denote a *writer* as any entity with write access to the database. In a DLT setting, a writer is a node of the network, also called a validator, who participates in the consensus protocol and accumulates transactions within a block to append the block to the chain. We denote a *reader* as any entity that cannot insert a block in the chain but can read the information written in the blocks and participate in the transaction creation process.

The main properties of DLTs and centralized systems are:

- **Public Verifiability** enables the verification of the correctness of the system's state. In a DLT, each state transition is confirmed by verifiers (e.g., miners in Bitcoin); nonetheless, anyone can verify that the state of the DLT was changed according to the consensus protocol. In contrast, in a centralized system, users may have different opinions regarding the state of the system itself. For example, they cannot verify that all state transitions have been executed correctly. In this case, users need to trust the central authority that provides them with the correct state.

- **Transparency** of information and the process of updating the system's state are fundamental for public verifiability. However, the amount of information that is transparent to a participant may change, and not every user may have access to all data.

- **Privacy** is an essential property of any system. It is easier to achieve in a centralized system because the first two properties are not fundamental for the functioning of the system; however, it depends on how loyal the central authority is. Through cryptography, privacy can be maintained while making everything public and transparent. For example, in Bitcoin, it is guaranteed through the use of the SHA-256 hash function and public key cryptography on elliptic curves.

- **Integrity** of information guarantees that data are protected from unauthorized modifications. Blockchains are designed to securely store data, making it highly resistant to modification once stored. However, in extremely rare cases, information may be altered through the use of hard forks, leading to a divergence from the previous version of the blockchain. Integrity is also closely

connected to public verifiability: if a system provides public verifiability of data, any user can verify its integrity. In contrast, in a centralized system, integrity can only be ensured if the latter is not compromised.

- **Redundancy** is achieved by the structure of the distributed ledger itself, as each node of the network has a copy of the ledger on its own device. In centralized systems, redundancy is generally achieved through backups and replication of data on physical servers.
- **Trust Anchor** represents the authority of a given system that has capability to grant and revoke read-and-write access. It is important to note that in a public permissionless blockchain, there is no central authority with such control.
- **Scalability** is the system's ability to handle a growing amount of work. A significant scalability challenge in public blockchain consensus protocols is the requirement for all transactions to pass through and be processed by all participating nodes. For instance, Bitcoin's blockchain can only process 7-10 transactions per second, whereas centralized systems can achieve higher scalability performances.

2.2 Typologies of blockchain

A **blockchain**, a specific type of DLT with unique properties, derives its name from its technical structure: a distributed database maintaining a growing list of blocks. Each block is linked to the previous one through a cryptographic hash function. Consequently, the blockchain expands continuously by adding new blocks at regular intervals, serving as a ledger for recording all transactions. Once a block is added, it becomes immutable, as its hash is included in the next block of the chain. The method of inserting a block into the chain depends on the consensus mechanism employed. Transactions, generated and exchanged by network peers, modify the blockchain's state. They enable nodes to exchange cryptocurrencies, which can also be used to execute smart contracts. A smart contract is a decentralized piece of code stored on the blockchain, programmed to execute autonomously when predefined conditions are met. Despite residing on the blockchain, smart contracts interact with external services known as oracles, which gather information from various sources. In other words, oracles play a crucial role in ensuring the accurate execution of a smart

contract. Additionally, one of the advantages of blockchain is its ability to facilitate transactions between individuals or entities without the need for intermediaries such as a Certificate Authority (CA) or a central entity in general.

It is crucial to clarify that there are three main types of blockchains:

1. **Public permissionless**: in a public permissionless blockchain, any peer can potentially join and leave the network as reader and writer at any times.

   Bitcoin [28] and Ethereum [29] exemplify public permissionless blockchains that are open, decentralized, and lack a central authority. Conversely, to restrict readers and writers to a specific set, permissioned blockchains have been introduced.

2. **Public permissioned**: in a public permissioned blockchain, anyone can read the data stored on the blockchain. However, the term "permissioned" implies that only a selected group of nodes have permission to control the transaction validation process and ledger maintenance. An example is Ripple [30], a blockchain-based digital payment network with its own cryptocurrency, XRP.

3. **Private permissioned**: private permissioned blockchains are centralized systems where access authorization is given only to a small number of nodes of the network. In a private permissioned blockchain, a central authority decides the right of individual peers of the network to participate in the write-and-read operations[1]. In other words, this is a centralized blockchain where only authorized and predefined nodes can read and write transactions. In general, a private blockchain is faster and more scalable than a public one [31].

   Hyperledger Fabric [32] and R3 Corda [33] are notable instances of private permissioned blockchains.

In Table 2.1, we present a comparison of key characteristics among public permissionless blockchains, public permissioned blockchains, private permissioned blockchains, and a central database. As observed, central databases generally exhibit higher performance in terms of latency and throughput when compared to distributed systems. This discrepancy arises from the additional complexity introduced by

[1]Public permissioned and private permissioned blockchains have a substantial difference: a public blockchain enables anyone to read the blocks of the chain and, thus, verify the validity of the data, while a private blockchain allows to read the chain only to a restricted number of nodes.

	Public Permissionless	Public Permissioned	Private Permissioned	Central Database
Throughput	Low	Medium	High	Very High
Latency	Slow	Medium	Fast	Very Fast
Number of users	High	Medium	Low	High
Numbers of readers	High	Medium	Low	High
Number of writers	High	Medium	Low	High
Consensus mechanism	PoW, PoS	BFT	BFT	None
Centrally managed	No	Partly	Yes	Yes

Table 2.1 Analysis of public permissionless, public permissioned, private permissioned, and central database characteristics.

consensus mechanisms in DLTs and blockchains. Given that the permissioned setting is less intricate than the permissionless one, permissioned blockchains can employ more efficient consensus protocols, such as Practical Byzantine Fault Tolerance (PBFT) [34], which have been established for many years.

2.2.1 Feasibility evaluation

Blockchain technology has garnered significant attention in recent years for its potential to revolutionize various industries, spanning from finance and supply chain management to healthcare and beyond. However, it is easy to get carried away by the hype around blockchain technology. For this reason, before implementing blockchain solutions, it is crucial to conduct a **feasibility evaluation**, considering multiple key factors. This assessment involves a comprehensive analysis to determine whether blockchain technology is the most suitable solution for a specific use case. Decision flowcharts, also known as decision trees, were recognized as the primary method used in most of the studies examined to aid decision-makers to integrate blockchains into business settings. According to Wust and Gervais [35], permissionless and permissioned blockchains only make sense when multiple mutually mistrusting entities seek to interact, changing the system's state without relying on an online Trusted Third Party (TTP). Building on their assumptions, the authors proposed a flow chart to ascertain whether a blockchain is the appropriate technical solution for a given problem. Additionally, also Fraga-Lamas and Fernàndez-Caramés [36] create a flow diagram, offering general guidance on determining the appropriateness of using blockchain and selecting its specific type. Drawing inspiration from [35]

and [36], we present a decision flow chart for evaluating the feasibility of blockchain as a technical solution for a specific use case (Figure 2.1).

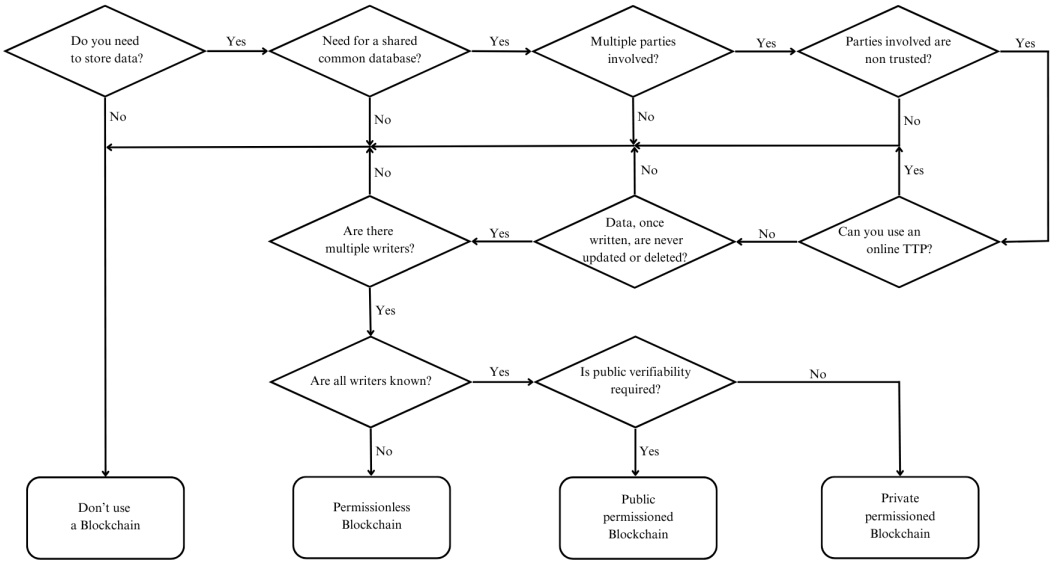

Fig. 2.1 A flow diagram to determine whether a blockchain is the appropriate technical solution for a specific application.

We are now presenting three main factors widely used in the feasibility evaluation process:

1. **Use case identification**: the initial step in evaluating the feasibility of blockchain technology is to clearly define the use case. Blockchain may not always be the sole suitable solution, and its advantages become evident when there is a need for transparency, security, and decentralization.

2. **Technical requirements**: once the use case is identified, organizations must assess the technical requirements. Blockchain technology comes in various forms, including public and private, permissioned and permissionless, with different consensus mechanisms. This step involves considerations related to scalability, data storage, etc.

3. **Cost-benefit analysis**: organizations should consider the initial investment required to set up the blockchain infrastructure, ongoing maintenance costs, and the potential returns on investment.

2.2.2 Data storage

The decision to store data on-chain or off-chain is critical in the design of blockchain-based applications, as it significantly impacts performance, security, and scalability. On-chain storage guarantees immutable and transparent transaction records, which are essential for applications requiring high levels of trust and verifiability, such as financial transactions and supply chain management. However, on-chain storage can present scalability challenges as data volumes increase, which can result in slower transaction processing and higher storage costs. The necessity to validate and store each transaction can exacerbate these issues, particularly during periods of high network congestion, leading to delays and increased transaction fees. Conversely, off-chain storage improves scalability by offloading large data volumes from the blockchain, thereby reducing the system's load and enabling faster transaction processing at lower costs. This method is particularly advantageous for applications that require high transaction frequency or manage large datasets, and it is generally more cost-effective due to the avoidance of high on-chain storage fees. However, off-chain data may lack the transparency and immutability inherent in on-chain data, which can be a significant drawback for applications where trust and verifiability are critical. Additionally, off-chain storage solutions require careful design to ensure data security and necessitate additional infrastructure to link on-chain transactions with off-chain data, thereby increasing development and maintenance complexity. Comprehensive diagrams [37], [38] explicitly incorporate the on-chain versus off-chain storage decision, providing a detailed framework that guides designers through the trade-offs between transaction speed, scalability, cost, and security.

2.3 Blocks

A **block** represents the basic unit of data within a blockchain. Although its exact structure may vary depending on the underlying protocol, several common elements appear across most designs. Typically, a block is composed of two main sections: the *header* and the *body*. Each block is uniquely identified by a digest, referred to as the *block hash*[2]. Moreover, aach block contains the hash of the previous block header, thereby linking blocks together and forming a chain of cryptographic hashes,

[2]The block hash is typically defined as the hash of the block header.

from which the term "blockchain" originates. The header also includes essential metadata, among which the following are usually present:

- the *timestamp*, specifying the date and time at which the block was produced;
- the *block height*, indicating the block's position within the chain, with the genesis block conventionally assigned height 0;
- the *Merkle root*, corresponding to the root of the Merkle tree [21] built using the hashes of all transactions in the block;

The body, on the other hand, contains the ordered list of transactions included in the block. The Merkle root acts as a cryptographic commitment to these transactions, enabling efficient proofs of inclusion via Merkle proofs.

2.4 Consensus protocols

In the absence of a central authority responsible for data management, distributed ledger systems require a mechanism that enables participants to reach a common agreement on the current and future state of the ledger. This objective is achieved through a **consensus protocol**, which allows a set of mutually distrustful nodes to collectively validate transactions, ensure the consistency of the transaction history, and regulate the process of block creation and addition to the blockchain. One of the primary security requirements of consensus mechanisms is resistance to *Sybil attacks*, in which an adversary attempts to gain disproportionate influence by creating multiple pseudonymous identities. Among the various consensus protocols proposed in the literature, the most widely adopted are *Proof-of-Work* (PoW) and *Proof-of-Stake* (PoS).

2.4.1 Proof of work

Proof of Work (PoW) is a cryptographic proof that allows a party to demonstrate to others that a significant amount of computational work has been performed, while ensuring that the verification of such proof requires only minimal computational effort. The basic idea underlying Proof of Work was introduced by Cynthia Dwork

and Moni Naor in 1992 [39] as a mechanism to mitigate denial-of-service attacks and email spam. The core principle is to require a user to perform a certain amount of computational effort before accessing a service. The term *Proof of Work* was later formalized by Markus Jakobsson and Ari Juels in 1999 [40]. However, the most influential and widely adopted instantiation is due to Adam Back, who proposed the Hashcash protocol in 2002 [41]. Hashcash is conceptually simple and effective. Given a text, the prover appends a random text, called a *nonce*, and computes the hash of the resulting string. The goal is to find a nonce such that the hash output is below a predefined target. If the condition is not satisfied, the prover must try again with a different nonce until a valid hash is found. Verification, on the other hand, is efficient: it suffices to recompute a single hash using the provided nonce and check whether the result meets the required target. In PoW-based systems, such as *Bitcoin*, consensus is achieved by requiring participants (called *miners*) to solve a computational puzzle based on cryptographic hash functions. The probability of a miner being selected to append a new block is proportional to its computational power, which makes large-scale attacks economically costly. PoW has been widely studied, offering strong security guarantees under standard cryptographic assumptions. However, it also presents several limitations. In particular, the consensus process requires a substantial amount of energy, as miners must continuously perform intensive hash computations. Moreover, PoW systems are subject to centralization pressures: miners often join *mining pools* to increase their chances of obtaining rewards. A well-known security concern is the possibility of a 51% *attack*. If an adversary controls the majority of the network's computational power, it can generate blocks faster than the honest network and thus build an alternative chain (see Section 2.5). By extending this chain, the adversary may cause the network to adopt a different transaction history, enabling attacks such as *double spending*. Many blockchain systems, such as *Bitcoin*, use PoW as their consensus mechanism. A comprehensive overview of PoW-based consensus protocols can be found in [42].

2.4.2 Proof of stake

Proof-of-Stake (PoS) is the second most widely adopted class of consensus protocols in blockchain systems. Originally proposed to address the high energy consumption of PoW, in PoS systems, participants known as *validators* are selected to propose and validate blocks according to the amount of cryptocurrency they lock as stake.

Validators perform two main roles: proposing new blocks and attesting to the validity of blocks proposed by other validators. Several mechanisms have been proposed in the literature to select the validator responsible for proposing the next block. Common approaches include:

- *randomness*, where the probability of selection is proportional to the amount of cryptocurrency staked;
- *seniority*, which takes into account both the quantity of staked currency and the duration for which it has been held, and is typically reset after a successful block proposal;
- *velocity*, which considers the rate at which the currency is used within the system.

Once selected, a validator proposes a block that is subsequently evaluated by a committee of validators chosen according to a randomized procedure. If a sufficient fraction of this committee attests to the correctness of the proposed block, the block is appended to the blockchain. Misbehavior can be penalized through mechanisms such as *slashing*, which provides strong economic disincentives against protocol violations. These penalties are designed to provide strong economic incentives for honest participation and to ensure that a large fraction of validators remains continuously available, which is essential for the scalability and reliability of the system. In PoS-based blockchains, any user with a computer and an internet connection can potentially act as a validator. However, participation requires staking a certain amount of cryptocurrency, which may be too high for individual users. To address this limitation, users can join *staking pools*, which are analogous to mining pools in PoW systems. In this setup, multiple participants combine their stakes, increasing the chances that one of the pool's nodes is selected as a validator. If one the nodes successfully proposes a block that is accepted by the network, all participants in the pool receive rewards proportional to their contributed stake. One of the main issues in PoS systems is the so-called *nothing-at-stake* problem. When a fork occurs (see Section 2.5), a validator may attempt to extend both branches of the chain, since it has a stake in each of them. In this way, for example, a validator could attempt to spend the same funds in two different transactions simultaneously.

2.5 Forks

The term **fork** is used in blockchain literature to denote different phenomena, depending on the underlying cause and its impact on protocol rules. In particular, it is common to distinguish among *regular forks*, *soft forks*, and *hard forks*. A regular fork arises when two or more valid blocks are generated nearly simultaneously and propagated through the peer-to-peer network with comparable delays. Due to the decentralized and asynchronous nature of the network, different nodes may temporarily observe different views of the blockchain and extend distinct branches. Each miner or validator follows the protocol by appending new blocks to the branch it considers canonical at that time.

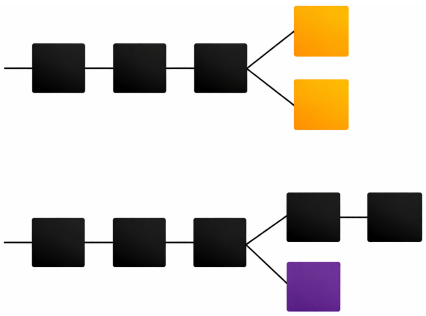

Fig. 2.2 Top: Example of a regular fork. Bottom: The fork is resolved by selecting the upper branch as the canonical chain, while the orphan block is highlighted in purple.

As the protocol progresses, one of the competing branches eventually becomes longer or accumulates more consensus, causing the network to converge toward a single canonical chain. Blocks belonging to the losing branches are excluded from the final ledger state and are commonly referred to as *orphaned blocks*. Transactions in these blocks do not take part in the final state of the system. Consequently, it is standard practice to wait for a sufficient number of blocks to be appended after a transaction before considering it confirmed. A soft fork is a software update to the blockchain protocol that is backward compatible and usually introduces improvements or optimizations. A node may choose not to install the update and continue participating in the blockchain without disruption. In contrast, a hard fork is a protocol update that is not backward compatible, and therefore all nodes must upgrade in order to remain part of the network. If this does not happen, two different situations may arise. In the first case, all nodes adopt the new rules and the blockchain continues to evolve as a single chain. In the second case, the network

splits into two groups: some nodes follow the new protocol, while others continue using the old one, resulting in the creation of two separate blockchains. These chains share the same history up to the point of the fork, and then evolve independently.

2.6 Cross-Chain Communication and Swap protocols

With the emergence of numerous projects based on blockchain, another problem emerged: the problem of communication between these different distributed systems. Formally, given two blockchains, the problem consists in enabling transactions between different blockchains. This problem is generally referred to as **Cross Chain Communication (CCC)** [43]. In this context, such cross-chain transactions are commonly referred to as **atomic swaps**. One of the first works that systematizes the knowledge of this problem was that of Vitalik Buterin in 2016 [44]. This report divided the methods of coin exchange into two categories: centralized and decentralized. Centralized methods were those in which participants sent funds to a third party entity that was considered trusted, as it was the keeper of the private keys of said funds. These methods are sometimes also called custodial. Examples of these centralized entities are online exchanges. The advantages of the centralized methods are their ease of implementation, their ease of use and their speed. However, this comes at the expense of user security. In fact, the custodian can carry out attacks such as censorship or literally run away with the funds. The second method of coin transfer, the decentralized one, allows participants to exchange coins without the use of a third party entity. In these cases the parties who want to exchange funds generate and solve cryptographic problems to lock or unlock the funds on either blockchain. A decentralized method, while safer for users, currently occurs at the expense of ease of use [45]. The difference between centralized and decentralized methods is, however, a distinction of convenience: some projects are a middle ground between the two methods. For example, some bridges use a group of people, usually called notaries, as a trusted third party. In this case the notaries must reach a consensus before moving funds. This method is safer for the user as more parties need to be non-honest for him to lose money. On the other hand, in an anonymous system it is not possible to know if these parties are colluding so they act as a single entity. So we can see that choosing a centralized or decentralized method comes with different costs and different benefits [45]. The underlying problem with decentralized methods

is demonstrated in the impossibility result of a fair exchange. This result discovered by Asokan [46] states that in an asynchronous context, such as blockchain, it is impossible to have a secure and fair exchange without a trusted third party. The trusted third party operates a synchronization work between the two parties. To bypass this outcome there are two methods. The first one is to make the trusted party more decentralized: this is the approach of the notaries, which cannot achieve complete decentralization because of the problems explained above. The second is to introduce synchronization methods that do not require a trusted party, leaving the method completely decentralized[3], or peer-to-peer. The most used method today to achieve synchronicity between two blockchains is based on Hashed TimeLock Contract (HTLC) [47, 48]. We briefly explain its working here (see Section 4.2.3 for further details). A HTLC system uses four transactions between two parties (Alice and Bob) to exchange funds from a blockchain B_A to a blockchain B_B. Whether the participants are honest or not, only two of these transactions are published, one for each blockchain. In the case that both Alice and Bob are honest, the two transactions actually exchange funds to Bob and Alice respectively. In the case where Alice or Bob (or both) are dishonest, however, the two transactions are structured in such a way that the parties can take back the transferred funds and lose nothing but time in the process. In Chapter 4, we present an alternative approach based on time-lock puzzles for implementing atomic swaps.

2.7 Limitations of blockhain technology

Blockchain is ushering in a paradigm shift across global industries by revolutionizing transactional and interactive processes. Its unique capacity to foster trust and facilitate value exchange among distrustful parties, while eliminating the need for intermediaries, is reshaping established norms. However, we must take into account that blockchain technology has its limitations. For instance, encrypting data and the consensus mechanism can enhance confidentiality but may decrease transparency and performance. Storing only a hash of data on-chain, while keeping the raw data off-chain, can improve both confidentiality and performance. In terms of performance the research is continuing to improve the efficiency of the consensus mechanism. For example, consensus algorithm as Ethash [49] and X13 [50] can come to a con-

[3] In Chapter 4, we will only deal with fully decentralized methods for exchanging funds.

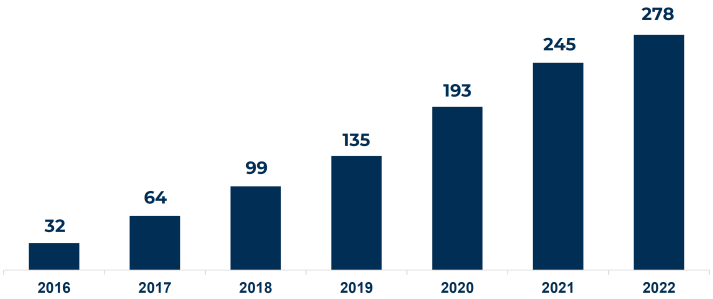

Fig. 2.3 Analysis of 1046 international projects Blockchain-based developed by companies and public administrations from 2016 to 2022. This research was carried out on February 2023 by the Blockchain & Web3 Observatory of the Politecnico di Milano [1].

sensus within 10 to 20 seconds, while Bitcoin takes 10 minutes on average. The linking of blocks through cryptographic hash establishes a form of immutability for all transactions recorded on the blockchain. While this typically ensures data integrity, it can also lead to complications. Real-world blockchain implementations face challenges such as disputed transactions, incorrect addresses, exposure or loss of private keys, data-entry errors or unexpected changes to assets tokenized on the blockchain. Moreover, using blockchain for data integrity may incur higher costs

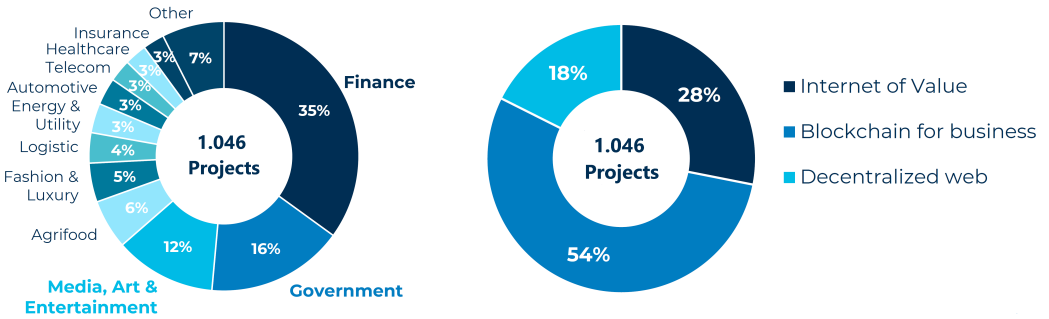

Fig. 2.4 Main fields of application of Blockchain technology. This census was executed in February 2023 by the Blockchain & Web3 Observatory of the Politecnico di Milano [1].

compared to classical methods. Although blockchains currently face scalability issues, this limitation may not be fundamental and could be resolved in the near future. Consortium and private blockchains, when meticulously designed and optimized for performance, exhibit significantly better scalability compared to public blockchains. Finally, in terms of throughput, conventional public blockchains are currently limited to process an average of 3-20 transactions per second, whereas established payment services such as Visa and Mastercard can handle 50,000 transactions per second.

Chapter 3

Verifiable and Anonymous Credentials

This chapter provides an overview of verifiable and anonymous credentials. After a brief introduction in Section 3.1, Section 3.2 introduces the Self-Sovereign Identity (SSI) model and discusses potential vulnerabilities within this model. Section 3.3 introduces the concept of verifiable credential schemes, outlining the algorithms and security properties that they must satisfy. Section 3.4 presents the mdoc VC scheme. Section 3.5 introduces anonymous credentials, describing the most important challenges associated with them. Section 3.6 presents the concept of an anonymous credential scheme, outlining the algorithms and security properties it must satisfy, and introduces the Linked Blinded Secret. Section 3.7 describes the BBS anonymous credential scheme. Finally, Section 3.8 provides an analysis of revocation mechanisms for verifiable and anonymous credentials.

3.1 Introduction

The eIDAS 2.0 regulation [51] aims to regulate electronic identification and trust services in the EU's internal market and to improve cross-border interoperability across Europe by introducing a new digital identity system that will be adopted by each member state. The regulation explicitly requires that this digital identity system provide citizens with the ability to selectively disclose the attributes certified in their credentials and to perform unlinkable authentications, with the goal of

protecting their privacy by reducing the amount of information they disclose[1], and to prevent tracking of their activities by colluding relying parties. The solution proposed to achieve these goals is the adoption of verifiable credentials (VCs) stored in a digital wallet called the **EUDI Wallet** [53]. VCs are the digital analogue of physical credentials, and their security relies on the use of cryptographic techniques. A VC is an unforgeable digital credential about the identity of a subject [54]. It contains the metadata to describe properties of the credential (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.

3.1.1 Main credentials format

One pivotal decision regarding the design of the EUDI Wallet concerns the choice of types of VCs it should support, enabling the selective disclosure of attributes and the unlinkability of presentations. So far, the community has highlighted two main approaches for the VC format [53, 55]. The first approach describes VCs designed exploiting the properties of hiding commitments and standard signatures [56], which we refer to as *mdoc VC*[2], while the second approach describes VCs whose design relies on signature schemes that support Non-Interactive Zero-Knowledge Proofs (NIZKPs), referred to as *anonymous credentials (AC)*[3]. Although both ACs and mdoc VC support selective disclosure of attributes[4] [59], ACs offer stronger privacy guaranties and ensure that multiple presentations of the same credential result to be unlinkable (a property often referred to as multi-show unlinkability) even against an adversary who corrupts both the issuer and the verifiers. Instead, multiple presentations generated from the same mdoc VC are linkable since every presentation always reveals information that identifies the credential used to generate it. To achieve a similar level of privacy for users, mdoc VCs rely on the batch

[1] in accordance with the privacy principles required by the GDPR [52]

[2] This name refers to the mobile driving license described in the standard ISO/IEC 18013-5[57].

[3] In particular the focus is on ACs designed following the framework of Camenisch and Lysyanskaya [58].

[4] In practice, attributes may include personally identifiable information (PII), such as a person's age or address, as well as derived statements like "age > 18", and non-PII information, such as membership in a group or organization. Every credential can thus be viewed as encoding at least one attribute.

issuance of single-use credentials. This countermeasure protects the holder against corrupted verifiers, but it does not protect the user's privacy if the verifiers collude with the issuer (or the verifier is also the issuer). Additionally, the batch issuance of credentials naturally presents relevant complications for secure deployment.

3.2 The Self-Sovereign Identity model

The **Self-Sovereign Identity (SSI)** is an emerging decentralized identity model [60] 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. Credential systems leveraging the SSI model can be described by the **Triangle of Trust (ToT)** depicted in Figure 3.1, where three different roles coexist:

- **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 [54] and provides evidences for Verifiers to check the revocation status [61] of all issued VC without compromising the privacy of the Holders.
- **Holder** owns one or more VC and generates a VP [54] 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).

In general, roles are dynamic and context-dependent, allowing a user to function as a holder, verifier, or issuer based on the specific operations they wish (or are allowed) to perform. For example, a verifier might also issue its own credentials for later access control, or a peer-to-peer environment may have holders mutually verifying each other. A given system may have one or many issuers, and an issuer may also be responsible for initialising the system. Issuers interact with holders

to issue credentials and they are generally not expected to interact directly with verifiers.

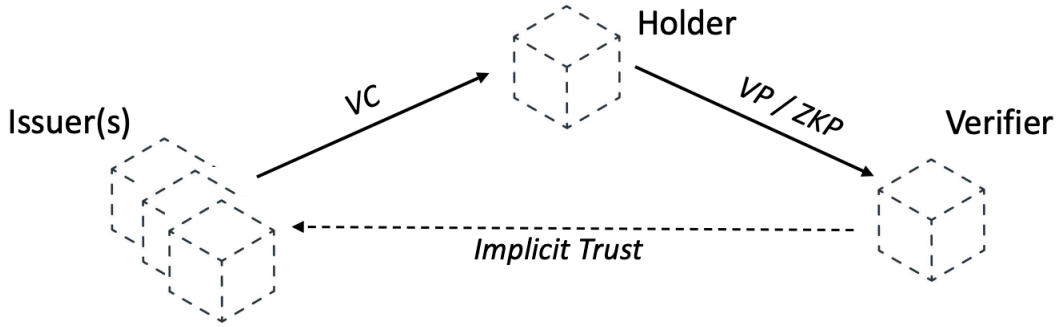

Fig. 3.1 The Triangle of Trust at the core of the SSI reference model.

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 credentials* [62, 63] 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)** are entities that sign and issue credentials for holders. They 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 protocols 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 (without revealing any identity attributes) or by selectively disclosing only a subset of the claims in the VC (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) [62].

- **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* [62], 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.

3.2.1 Potential Vulnerabilities in SSI systems

A credential-based framework, including both verifiable and anonymous credential systems, may present several weaknesses that depend on the underlying trust model, system design, and implementation choices. Some of these limitations are inherent to systems providing anonymity and unlinkability, while others depend on deployment choices and trust assumptions.

- **Malicious Issuer**: An issuer may behave dishonestly by issuing invalid credentials or revoking valid ones without justification. This risk is typically mitigated by relying on a set of trusted issuers and appropriate governance mechanisms.
- **Malicious Verifier**: A verifier may request more attributes than necessary, potentially undermining user privacy. To address this, holders should retain control over which attributes are disclosed and only reveal the minimum required information.
- **Credentials sharing**: Credentials may be shared between users, allowing unauthorized parties to use them. This risk can be mitigated by binding credentials to a holder's secret (see Section 3.6.2) and protecting such secret within secure hardware.
- **Issuer Leakage**: Sensitive information about credentials and holders may be exposed if an issuer's database is compromised. This can be mitigated by minimizing stored data and applying strong encryption techniques.

- **Holder Leakage**: Information may also be leaked from the holder's device. Secure storage, encryption, and the use of tamper-resistant hardware are important to protect credentials and cryptographic secrets.
- **Weak Randomness**: The security of many cryptographic protocols relies on high-quality randomness. Poor or predictable randomness can lead to vulnerabilities, making signatures and proofs easier to forge.
- **Issuer–Verifier collusion**: may allow identification of a holder if shared information can be correlated. In particular, issuer–verifier collusion may compromise holder anonymity.
- **Verifier–Verifier collusion**: Verifier–verifier collusion primarily affects unlinkability property, as multiple verifiers can correlate separate interactions and track a holder across different service providers, especially when static attributes are disclosed.

Overall, these vulnerabilities highlight that the security and privacy guarantees of SSI systems depend not only on the underlying cryptographic primitives, but also on system design, trust assumptions, and secure implementation practices.

3.3 Verifiable Credential scheme

A **verifiable credential scheme** is defined by a set of algorithms and protocols that establish secure processes to issue, present and verify digital credentials [59].

Definition 3.3.1 (Verifiable Credential scheme) *A* verifiable credential scheme *is defined by the following algorithms:*

- Issuer Setup *(*$\{\mathsf{pp}, (\mathsf{sk}_{\mathsf{Iss}}, \mathsf{pk}_{\mathsf{Iss}})\} \xleftarrow{\$} \mathsf{IssuerSetup}(1^{\lambda})$*): this algorithm is executed by the issuer, and it generates the public parameters* pp *for the verifiable credential scheme together with the issuer key pair* $(\mathsf{sk}_{\mathsf{Iss}}, \mathsf{pk}_{\mathsf{Iss}})$.
- Credential Issuance *(*$\mathsf{cred} \xleftarrow{\$} \mathsf{CredIssuance}(\{a_i\}_{i\in[l]}, \mathsf{sk}_{\mathsf{Iss}})$*): The issuer executes this algorithm to generate a verifiable credential* cred *for the attributes* $\{a_i\}_{i\in[l]}$.

- Credential Presentation *(*$\mathsf{pres} \xleftarrow{\$} \mathsf{CredPresentation}(\mathsf{cred}, \mathsf{stmt})$*): This algorithm is executed by the holder to generate a proof (the presentation* pres*) that it possesses a credential* cred *that satisfies a statement* stmt[5].
- Presentation Verification *(*$\{0,1\} \xleftarrow{\$} \mathsf{PresVer}(\mathsf{pres}, \mathsf{stmt}, \mathsf{pk}_{\mathsf{Iss}})$*): The verifier executes this algorithm to verify the validity of the presentation provided by the holder, ensuring that it satisfies* stmt.

3.3.1 Security Properties of a Verifiable Credential Scheme

A Verifiable Credential scheme must satisfy two fundamental security properties:

- **Correctness:** A CredPresentation/PresVer protocol will succeed if the input is a valid credential resulting from a CredIssuance protocol previously run between the holder and an issuer known to the verifier.
- **Unforgeability:** A Holder cannot feasibly produce a presentation pres that is accepted as valid by the Verifier without possessing one or more valid credentials issued by the claimed Issuer, such that each asserted attribute is attested by at least one of these credentials.

In both verifiable and anonymous credential systems, each credential is associated with a digital signature over a set of attributes. However, a presentation may involve multiple credentials. The key difference is that in verifiable credential systems these signatures are directly verified, while in anonymous credential systems the prover must provide a zero-knowledge proof of knowledge of the underlying signatures. Verifiable credential systems do not provide *collusion resistance*[6]: multiple parties can jointly contribute valid credentials to a single presentation without any guarantee that they are controlled by the same entity. In contrast, anonymous credential systems that provide collusion resistance require that all credentials in a presentation are

[5]The presentation can allow the holder to selectively disclose the attributes of the credential, revealing only the attributes in stmt.

[6]Collusion resistance is the property that prevents two or more holders from combining their credentials in order to generate a proof that none of them could have produced individually. In verifiable credential systems, the verifier only checks the validity of individual credentials and their signatures, without enforcing any binding between them. As a result, two different holders can combine their credentials in a single presentation to satisfy a joint statement, even if neither of them could satisfy the statement individually, thus enabling successful collusion.

bound to a common user secret (*master secret*), and that the prover demonstrates knowledge of this secret within the zero-knowledge proof. Under this assumption, any valid presentation of one or more credentials must be produced by a single entity, thereby preventing collusion.

3.4 mdoc VC scheme

The cryptographic primitives used to design **mdoc VCs** are (1) hiding commitments based on hash functions and (2) digital signature schemes $= (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vf})$. The digital signature scheme $\mathscr{DS}$ can be any SUF-CMA secure digital signature scheme.

Definition 3.4.1 (mdoc VC scheme) *Let* $\mathscr{DS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vf})$ *be a digital signature scheme and H a cryptographic hash function. The* mdoc VC scheme *can be described as follows:*

- Issuer setup: $\{\mathsf{pp}, (\mathsf{sk}_{\mathsf{Iss}}, \mathsf{pk}_{\mathsf{Iss}})\} \xleftarrow{\$} \mathsf{IssuerSetup}(1^{\lambda})$.

  The issuer executes the IssuerSetup *algorithm which consists in executing the* Setup *and* KeyGen *algorithms of the underlying digital signature scheme* $\mathscr{DS}$. *This algorithm generates:*

  - *the public parameters:* $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^{\lambda})$;
  - *the issuer key pair:* $(\mathsf{sk}_{\mathsf{Iss}}, \mathsf{pk}_{\mathsf{Iss}}) \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{pp})$.

- Credential issuance: $\mathsf{cred} \xleftarrow{\$} \mathsf{CredIssuance}(\{a_i\}_{i\in[l]}, \mathsf{sk}_{\mathsf{Iss}})$.

  The issuer executes the following operations:

  - *sample uniformly at random salts* $\mathsf{salt}_1, \ldots, \mathsf{salt}_l \xleftarrow{\$} \{0,1\}^{\lambda}$;
  - *compute* $\mathsf{com}_i \leftarrow H(a_i || \mathsf{salt}_i) \, \forall i \in [l]$;
  - *generate* $(\mathsf{sk}_{\mathsf{cred}}, \mathsf{pk}_{\mathsf{cred}}) \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{pp})$;
  - *sign the commitments and the public key* $(\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}})$ *computing* $\sigma \xleftarrow{\$} \mathsf{Sign}((\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}}), \mathsf{sk}_{\mathsf{Iss}})$;

  - *set* $\mathsf{cred} \leftarrow ((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}}), \{a_i\}_{i\in[l]}, \{\mathsf{salt}_i\}_{i\in[l]}, \mathsf{sk}_{\mathsf{cred}})$[7].

- Credential presentation*:* $\mathsf{pres} \xleftarrow{\$} \mathsf{CredPresentation}(\mathsf{cred}, \mathsf{stmt}, \mathsf{nonce})$.

  The statement $\mathsf{stmt} = \{a_i\}_{i\in\mathsf{Rev}}$ *is given by the set of attributes that the holder wants to reveal* $\mathsf{Rev} \subseteq [l]$*, and the nonce* nonce *is sent by the verifier to the holder to guarantee the freshness of the presentation. The holder performs the following operations:*

  - *compute* $\mathsf{pres}' \leftarrow ((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}}), \{\mathsf{salt}_i\}_{i\in\mathsf{Rev}}, \{a_i\}_{i\in\mathsf{Rev}}, \mathsf{nonce})$*;*
  - *sign* pres' *computing* $\sigma' \xleftarrow{\$} \mathsf{Sign}(\mathsf{pres}', \mathsf{sk}_{\mathsf{cred}})$*;*
  - *set* $\mathsf{pres} \leftarrow (\mathsf{pres}', \sigma')$.

- Presentation Verification*:* $\{0,1\} \xleftarrow{\$} \mathsf{PresVer}(\mathsf{pres}, \mathsf{stmt}, \mathsf{pk}_{\mathsf{Iss}})$.

  The verifier performs the following operations:

  - *parse* $\mathsf{pres} \rightarrow (\mathsf{pres}', \sigma')$
    and then $\mathsf{pres}' \rightarrow ((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}}), \{\mathsf{salt}_i\}_{i\in\mathsf{Rev}}, \{a_i\}_{i\in\mathsf{Rev}}, \mathsf{nonce})$*;*
  - *verify the signature of the issuer:* $1 \leftarrow \mathsf{Vf}(\sigma, (\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}}), \mathsf{pk}_{\mathsf{Iss}})$*;*
  - *check that* $\mathsf{com}_i = H(a_i||\mathsf{salt}_i), \forall i \in \mathsf{Rev}$*;*
  - *verify the signature of the holder:* $1 \leftarrow \mathsf{Vf}(\sigma', \mathsf{pres}', \mathsf{pk}_{\mathsf{cred}})$.

  If the previous checks are satisfied, the verifier accepts and outputs 1, otherwise it outputs 0.

Privacy Concerns for mdoc VCs. This kind of VC offer limited privacy protection because each presentation of the same credential can be linked to the original credential (e.g. the signature of the issuer, or the public key of the credential must be always revealed). This linkability implies that repeated use of the same credential may allow verifiers to track and correlate different interactions, thereby undermining user privacy. For this reason the EUDI ARF currently instructs the member states to implement batch issuance of these VCs that must be considered as single-use credentials. However, even if these credentials are used only once, if the verifier

[7]To ensure a secure binding between a credential and the device storing it, the secret key of the credential ($\mathsf{sk}_{\mathsf{cred}}$) is stored within a hardware security module (HSM) or a trusted execution environment (TEE) embedded in the device.

colludes with the issuer, or it is the same issuer who generated the VC, then the VC presentation can be easily linked to its issuance.

3.5 Anonymous Credentials

Anonymous credentials (ACs) are a class of cryptographic tools, first introduced by Chaum in 1985 [64], that enable users to prove aspects of their identity, such as group membership or other attributes, without disclosing their personal information. Interest in these systems has grown significantly in recent years, largely driven by the increasing integration of the Internet and web-based services into everyday life, which has in turn raised substantial privacy concerns at scale. In this context, users are increasingly motivated to minimize the disclosure of personal information when accessing digital services. Users are often required to provide *Personally Identifiable Information (PII)* to demonstrate that they satisfy certain criteria, such as having reached the age of majority. Ideally, we would like to be able to demonstrate compliance with such requirements without revealing any additional personal information. Moreover, even in the absence of explicit PII, the ability to link a user's activities across multiple sessions can enable the construction of detailed profiles over time. A primary approach to mitigating both of these issues is the use of anonymous credentials. These systems are designed to protect the privacy of individuals while still allowing them to authen ticate themselves and access certain resources or services. Since the publication of the first fully anonymous credential scheme by Camenish and Lysyanskaya [65] (EUROCRYPT 2001) there have been numerous applications of anonymous credentials in various settings including: enabling users to access online services without revealing their identity [66], supporting privacy-preserving electronic identity cards [67], managing sensitive health information [68], enhancing privacy in smart energy grids [69], enabling secure e-voting systems [70], and providing privacy-preserving access control in cloud computing environments [71]. Anonymous credentials have predominantly been a matter of academic study for many decades, however they have also started to find their way into practice. In particular there are commercial implementations of anonymous credentials, for instance in Microsoft's uProve [72] and in IBM's idemix [66, 73].

Blind Attributes. In some anonymous credential systems, certain attributes are encoded so that the issuer can sign the credential without learning their values. This

is particularly important when the credential includes the prover's secret key as an attribute, allowing the prover to demonstrate ownership of the credential [24].

3.5.1 Anonymous Credentials in practice

Despite more than two decades of research on fully developed anonymous credential systems, their adoption in real-world applications remains limited. In this section, we discuss the main challenges that practical anonymous credential systems must address, beyond their core security properties, and review the solutions proposed in the literature

Efficiency. Efficiency is one of the most widely recognized challenges in the literature. Many credential schemes incur significant computational costs in order to achieve their core unlinkability properties. Over time, however, the efficiency of anonymous credential systems has improved, driven by advances in zero-knowledge proofs, the design of more streamlined protocols, and the introduction of self-blindable[8] credentials.

Non-transferability. One of the most challenging and practical issues is preventing users from sharing their credentials with others. Although anonymous credential systems typically provide collusion resistance, this does not prevent a user from directly transferring or sharing their credential with another party. This issue was already identified by Camenisch and Lysyanskaya in [24], although no concrete technical solution was proposed. Instead, they suggested discouraging such behaviour by binding all credentials of the same user to a shared secret (see Section 3.6.2), or by linking credential usage to an external and valuable resource, such as a bank account.

3.6 Anonymous Credential scheme

An **anonymous credential scheme** is defined by a set of algorithms and protocols that establish secure processes to issue, present and verify the AC.

[8] Self-blindable credentials are credentials that can be randomized by the holder without interacting with the issuer. This allows the same credential to be presented multiple times in a way that produces unlinkable transcripts and reducying reliance on complex proof mechanisms.

Definition 3.6.1 *Let $\mathscr{DS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vf})$ be a digital signature scheme and H a cryptographic hash function. An* anonymous credential scheme *is defined by the following algorithms:*

- Issuer setup: $\{\mathsf{pp}, (\mathsf{sk}_{\mathsf{Iss}}, \mathsf{pk}_{\mathsf{Iss}})\} \xleftarrow{\$} \mathsf{IssuerSetup}(1^{\lambda})$.

  The issuer executes the IssuerSetup *algorithm which consists in executing the* Setup *and* KeyGen *algorithms of the underlying digital signature scheme $\mathscr{DS}$. This algorithm generates:*

  - *the public parameters:* $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^{\lambda})$*;*
  - *the issuer key pair:* $(\mathsf{sk}_{\mathsf{Iss}}, \mathsf{pk}_{\mathsf{Iss}}) \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{pp})$.

- Credential issuance: $\mathsf{cred} \xleftarrow{\$} \mathsf{CredIssuance}(\{a_i\}_{i\in[l]}, \mathsf{sk}_{\mathsf{Iss}})$.

  The issuer executes the following operations:

  - *sign the attributes $\{a_i\}_{i\in[l]}$ using their private key $\mathsf{sk}_{\mathsf{Iss}}$, computing $\sigma \xleftarrow{\$} \mathsf{Sign}(\{a_i\}_{i\in[l]}, \mathsf{sk}_{\mathsf{Iss}})$;*
  - *set* $\mathsf{cred} \leftarrow (\sigma, \{a_i\}_{i\in[l]})$.

- Credential presentation: $\pi \xleftarrow{\$} \mathsf{CredPresentation}(\mathsf{cred}, \mathsf{stmt}, \mathsf{nonce})$.

  The statement $\mathsf{stmt} = \{a_i\}_{i\in\mathsf{Rev}}$ is given by the set of attributes that the holder wants to reveal $\mathsf{Rev} \subseteq [l]$, and the nonce *is sent by the verifier to the holder to guarantee the freshness of the presentation. The holder constructs a non-interactive zero-knowledge proof π that demonstrates that the attributes $\{a_i\}_{i\in\mathsf{Rev}}$ are included in* anoncred. *To do that, the holder:*

  - *given $(\sigma, \{a_i\}_{i\in[l]})$, generates a commitment t;*
  - *computes the challenge $c \leftarrow H(t, \mathsf{nonce})$;*
  - *computes a response z, and set $\pi = (t, z)$.*

- Presentation Verification: $\{0,1\} \xleftarrow{\$} \mathsf{PresVer}(\pi, \mathsf{stmt})$.

  The verifier performs the following operations:

  - *computes $c = H(t, \mathsf{nonce})$;*
  - *checks whether (t, c, z) is an accepting transcript for $\{a_i\}_{i\in\mathsf{Rev}}$.*

  If the previous checks are satisfied, the verifier accepts and outputs 1, otherwise it outputs 0.

3.6.1 Security Properties of an Anonymous Credential Scheme

In this system, no entity (Issuer, Holder, Verifier) is assumed to be fully trusted, and two primary threats must be considered. First, a malicious holder may attempt to have a presentation accepted as valid by the verifier, even if it is derived from a credential that is invalid, was never issued, or contains data inconsistent with the claims being presented. Second, a malicious issuer, verifier, eavesdropper, or any colluding subset of these parties may attempt to identify a holder from a protocol transcript, for instance by linking the transcript to a previous interaction by the same holder. For these reasons, the basic properties that an anonymous credential system must satisfy are as follows:

- **Correctness:** A CredPresentation/PresVer protocol will succeed if the input is a valid credential resulting from a CredIssuance protocol previously run between the holder and an Issuer known to the Verifier.
- **Unforgeability AC:** A Holder cannot feasibly produce a presentation pres that is accepted as valid by the verifier without possessing one or more valid credentials issued by the claimed issuer, such that each asserted attribute is attested by at least one of these credentials.
- **Issuer Unlinkability:** A probabilistic polynomial-time adversary observing a valid transcript of the CredPresentation/PresVer protocol should have only negligible advantage in determining which execution of the CredIssuance protocol generated the credential used in that transcript.
- **Multi-show Unlinkability:** A probabilistic polynomial-time adversary observing multiple executions of the CredPresentation/PresVer protocols under the same issuer should have only negligible advantage in determining whether any two or more of these executions correspond to the same underlying credential.

Among these properties, multi-show unlinkability is the most significant, as it characterizes anonymous credential systems and distinguishes them from more general verifiable credential frameworks, as well as from related primitives such as blind signatures.

3.6.2 Linked Blinded Secret

In Section 3.3.1, we have shown that anonymous credential systems achieving collusion resistance require all credentials included in a presentation to be bound to a common user secret. A standard mechanism to enforce this requirement is the **Linked Blinded Secret (LBS)** [58], which enables more trusted interactions because it allows verifiers to ensure that one or more credentials presented in a proof are bound to the same holder. This is achieved through a zero-knowledge proof of knowledge of valid credential signatures together with a common underlying secret, ensuring both the authenticity of the credentials and their binding to a single entity. The Linked Blinded Secret is a large random number, a *master secret*, which is then committed using a cryptographic commitment algorithm. The commitment is sent by the holder to the issuer that signs it together with the credential attributes and allows the holder to prove it knows the secret without revealing it. It is referred to as a Linked Blinded Secret due to the following properties:

- **Linked**. The same master secret is incorporated into multiple credentials issued to the same holder. This enables a cryptographic binding between the holder and their credentials.
- **Blinded**. The master secret is concealed and randomized through a blinding commitment mechanism during each credential issuance.
- **Secret**. The master secret remains private and is never disclosed by the holder. Instead, only commitments to the secret and zero-knowledge proofs of knowledge of its value are shared.

As mentioned before, the Linked Blinded Secret has two purposes:

- **Proof of holder binding**. Since the holder is the only one who knows the master secret, no one else is able to generate a proof to present the credential(s).
- **Proof of credential linking**. The master secret is the same across all credentials owned by the same holder, which may combine more credentials in a single presentation. Each credential contains a different[9] Linked Blinded

[9]While the master secret is the same across all credentials of a holder, each credential contains a distinct commitment to this secret, obtained using fresh randomness. This ensures that all the credentials of a holder are cryptographically bound to the same underlying secret, while preserving unlinkability across different issuances.

Secret but the holder can prove that the secret behind those values is the same, without revealing it.

In anonymous credential systems without a master secret, zero-knowledge proofs ensure the correctness and privacy of individual credentials but do not enforce a binding between them. As a result, multiple holders can combine their credentials in a single presentation to satisfy a joint statement, leading to collusion.

3.7 BBS Anonymous Credential scheme

In this section, the **BBS Anonymous Credential scheme**, based on BBS signatures, is presented.

Definition 3.7.1 (BBS AC scheme) *Let* $\mathscr{DS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vf})$ *be a digital signature scheme and H a cryptographic hash function. The* BBS AC scheme *is defined as follows:*

- Issuer setup: $\{\mathsf{pp}, (\mathsf{sk}_{\mathsf{Iss}}, \mathsf{pk}_{\mathsf{Iss}})\} \xleftarrow{\$} \mathsf{IssuerSetup}(1^{\lambda})$.

  The issuer executes the IssuerSetup *algorithm which consists in generating the public parameters* pp *for the BBS credential scheme along with the key pair of the issuer* $(\mathsf{sk}_{\mathsf{Iss}}, \mathsf{pk}_{\mathsf{Iss}})$.

  $$\{\mathsf{pp}, (\mathsf{sk}_{\mathsf{Iss}}, \mathsf{pk}_{\mathsf{Iss}})\} \xleftarrow{\$} \mathsf{IssuerSetup}(1^{\lambda}),$$

  where $\mathsf{pp} = (g_1, (h_i)_{i=1}^{l}, g_2, \mathsf{pk}_{\mathsf{Iss}}, \mathbf{e})$, $\mathsf{sk}_{\mathsf{Iss}} = x \xleftarrow{\$} \mathbb{Z}_p$, *and* $\mathsf{pk}_{\mathsf{Iss}} = X_2 = g_2^x$.

- Credential issuance: $\mathsf{cred} \xleftarrow{\$} \mathsf{CredIssuance}(\{a_i\}_{i\in[l]}, \mathsf{sk}_{\mathsf{Iss}})$.

  The issuer executes this algorithm to generate a BBS anonymous credential. It takes as input the attributes $(a_1, \ldots, a_l)$ *and the issuer's private key* $\mathsf{sk}_{\mathsf{Iss}}$, *samples* $e \xleftarrow{\$} \mathbb{Z}_p$, *computes*

  $$A = \left(g_1 \prod_{i\in[l]} h_i^{a_i} \right)^{\frac{1}{x+e}},$$

and outputs a credential cred*:*

$$\mathsf{cred} = (\sigma, \{a_i\}_{i\in[l]}) = ((A,e), \{a_i\}_{i\in[l]}) \xleftarrow{\$} \mathsf{CredIssuance}(\{a_i\}_{i\in[l]}, \mathsf{sk}_{\mathsf{Iss}}).$$

This credential cred *includes the attributes* $(a_1,\ldots,a_l)$ *and its signature* (A,e) *obtained using the BBS signature scheme with secret key* $\mathsf{sk}_{\mathsf{Iss}} = x$*.*

- Credential presentation: $\mathsf{pres} \xleftarrow{\$} \mathsf{CredPresentation}(\mathsf{cred}, \mathsf{stmt}, \mathsf{nonce})$.

  The statement $\mathsf{stmt} = \{a_i\}_{i\in\mathsf{Rev}}$ *is given by the set of attributes that the holder wants to reveal* $\mathsf{Rev} \subseteq [l]$*, and the* nonce *is sent by the verifier to the holder to guarantee the freshness of the presentation. To produce a presentation, the holder executes the* CredPresentation *algorithm described in Figure 3.2:*

$$\mathsf{pres} = (\bar{A}, \bar{B}, c, s, t, (u_i)_{i\in\mathsf{Hid}}) \xleftarrow{\$} \mathsf{CredPresentation}(\mathsf{cred}, \mathsf{stmt}, \mathsf{nonce}),$$

- Presentation Verification: $\{0,1\} \xleftarrow{\$} \mathsf{PresVer}(\mathsf{pres}, \mathsf{stmt}, \mathsf{pk}_{\mathsf{Iss}})$.

  The verifier executes the the PresVer *algorithm described in Figure 3.2: if the checks are satisfied, the verifier accepts and outputs 1, otherwise it outputs 0.*

The CredPresentation *and* PresVer *algorithms are illustrated in Figure 3.2, which provides an overview of the non-interactive zero-knowledge proof (NIZKP) construction for BBS credentials.*

In [74], it was proved that this NIZKP is secure since it is derived from a sigma protocol to which is applied the Fiat-Shamir transform. In [74] the authors show that the sigma protocol satisfies *correctness*, *special soundness*, *honest-verifier zero-knowledge*. Moreover, it is easy to see that the sigma protocol has a large challenge space and high first message min-entropy; therefore, the derived NIZKP is secure [75].

Unlike anonymous credential systems that rely on a linked user secret to enforce binding between credentials, the BBS construction considered here does not explicitly incorporate a master secret shared across credentials. As a consequence, multiple credentials may be combined within a single presentation without any cryptographic guarantee that they originate from the same holder. This implies that the scheme, in this form, does not provide collusion resistance.

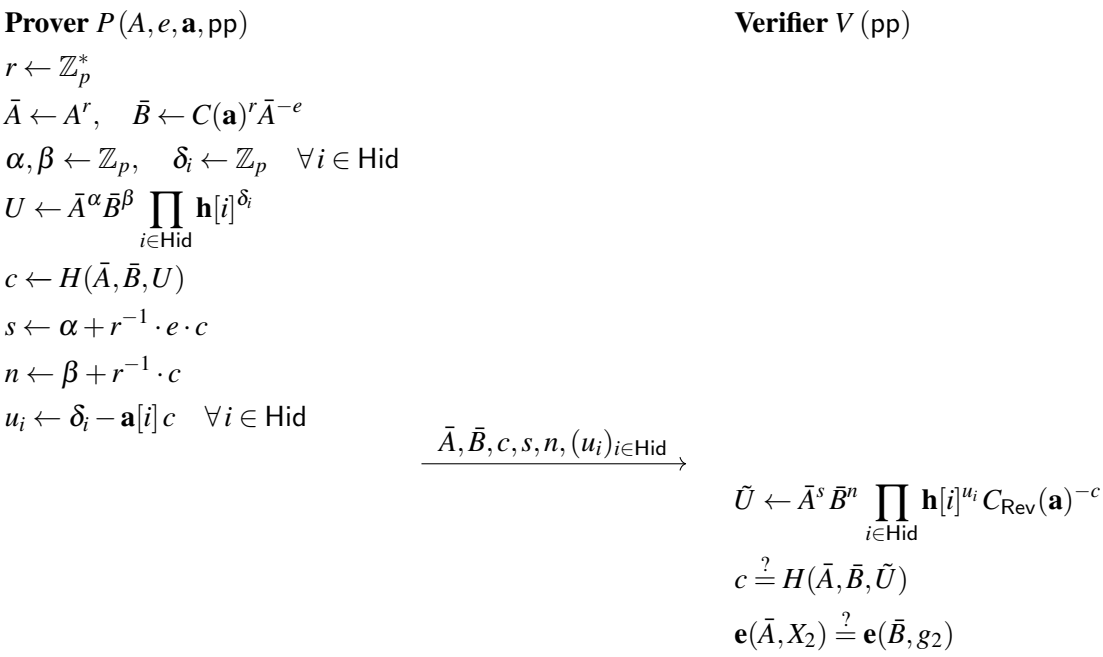

Fig. 3.2 NIZKP for BBS credentials. The value pp represents a set of public parameters known both to P and V.

3.8 Revocation Mechanisms

A fundamental and practical requirement of traditional certificate infrastructures is the ability to revoke certificates, thereby rendering them invalid when they are no longer trustworthy, for instance due to key compromise or misuse of privileges. This requirement is equally important in anonymous credential systems; however, it becomes considerably more challenging[10] to achieve in settings where credentials are designed to remain unlinkable across multiple uses. The possible reasons why a credential needs to be revoked can be grouped in three categories: (*i*) natural expiration, 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

[10]While the W3C standard revocation mechanism [61] can also be adopted in post-quantum plaintext verifiable credential systems, as it does not rely on cryptographic assumptions vulnerable to cryptographically relevant quantum computers (CRQCs), it is generally unsuitable for privacy-sensitive scenarios. In particular, the use of indexed status entries and deterministic status checking can introduce implicit linkability between credential uses, which is incompatible with strong privacy guarantees such as multi-show unlinkability in anonymous credential systems.

asks for a revocation of its credential. A naïve approach is to generate a new signing key and re-issue all non-revoked credentials whenever a revocation event occurs. While this method achieves zero latency at verification time, it introduces a substantial overhead, as the cost grows linearly with the number of valid credentials in the system. Lapon et al. [76] study different strategies for supporting credential revocation, identifying six main categories of revocation approaches, each with its own trade-offs in terms of security and efficiency. In the following, we briefly review some of the most important approachs to revoke credentials.

Pseudonyms. The first approach to revocation consists of attaching a consistent alias to each credential, which can then be used to register which credentials have been revoked. While this enables straightforward revocation checking, it fundamentally compromises multi-show unlinkability, since all presentations of the same credential can be linked through the pseudonym. Consequently, relying exclusively on this approach eliminates the main privacy advantage of anonymous credentials over linkable credential systems.

Limited Lifespan. An alternative approach is to include an expiration date within each credential and enforce a short validity period, requiring users to periodically renew their credentials. When used as the sole revocation mechanism, this approach resembles the naïve solution, but allows for a trade-off between revocation latency and the overhead associated with frequent re-issuance. Nevertheless, it remains one of the least scalable solutions. Expiration dates are, however, an essential component of practical credential systems and have commonly been implemented as one of the attributes since the introduction of Camenisch–Lysyanskaya credentials [24]. In zero-knowledge credential systems, proving that a credential is still valid typically requires demonstrating that the expiration date lies in the future without revealing it. This is often achieved through range proofs, which can be relatively inefficient, since the expiry date can otherwise become linkable information.

Signature Lists. Another approach to revocation, introduced in works such as Tsang et al. [77, 78] and Brands et al. [79], avoids the need for a TTP by maintaining a public list of revoked credential identifiers. Provers can then demonstrate in zero-knowledge that their credential is not included in this list, while keeping their identifier hidden through a one-way encoding. The main drawback of this approach is its computational cost, since proving non-membership in a set is generally expensive. Some optimisations have been proposed, but they introduce new drawbacks: either

they reduce privacy by requiring more explicit disclosure of identifiers [80], or they increase administrative overhead due to the need for frequent list updates [81].

Accumulator. Camenisch *et al.* in [82, 83] propose the use of a dynamic accumulator to solve the revocation problem. This primitive allows a large set of integer values to be compressed into a single value, with each member of the set having an associated witness that can be used in conjunction with the accumulator value to prove its membership in the set without access to the full set. 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. The Issuer knowing the trapdoor is the only able to add or remove values from the accumulator that acts as a membership list. A prover, using his witness, can produce a ZK-Proof of Knowledge proving its value is included in the accumulator. Accumulators have more efficient proving and verifying protocols than signature lists, and reduce the communication cost of publishing the list, but their primary drawback is the need for provers to update their witness values regularly with update information sent by the entity (usually an Issuer) managing the accumulator.

Timestamp. Camenisch *et al.* in [84] 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 (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.

Part II
Original contributions

After introducing the necessary background, the second part of the thesis presents the original contributions of this work. In particular, Chapter 4 introduces a secure protocol for atomic swaps based on time-lock puzzles. We present BTLE (Broadcast Time-Lock Exchange Protocol), a two-step protocol designed to decentralize the exchange of funds between two blockchains in scenarios similar to online exchanges. Chapter 5 describes the implementation of a post-quantum anonymous verifiable credential framework. Finally, Chapter 6 provides a formal definition of a VP delegation scheme, together with the security properties it must satisfy. We present two instantiations of the scheme, one compatible with verifiable credentials in the EUDI ARF, and another supporting anonymous credentials based on BBS signatures.

Chapter 4

BTLE: Atomic swaps with time-lock puzzles

The result presented in this chapter is a joint work with Nadir Murru, Claudio Schifanella and Fadi Barbara and it can be found in [85].

We present BTLE (Broadcast Time-Lock Exchange Protocol), a two-step protocol that aims to decentralize exchange of funds between two blockchains in scenarios similar to online exchanges. BTLE leverages time-lock puzzles to achieve that. In the first phase, the BTLE-MA protocol allows for a matching between a market maker and one of the competing market takers. In the second phase, the BTLE-AS algorithm allows the exchange between the market maker and the winning market taker. It is not necessary to use both the BTLE-MA and BTLE-AS algorithms in a decentralized-exchange scenario: existing atomic swaps based on HTLC can benefit from BTLE-MA and can be adapted to an exchange where there are multiple possible participants. Moreover, BTLE computations are off-chain, so BTLE can be used in those blockchain pairs where at least one of the two does not have a scripting language or where the pair do not have the same hash function in common. This solves a limitation of HTLC-based atomic swaps. We also propose a new time-lock puzzle based on Pell conic calculations as an alternative to the classical time-lock puzzle of Rivest et. al. BTLE has been implemented and tested. Experiments demonstrate that this new time-lock puzzle based on the Pell conic is superior for the intended goal. With an N-bit modulus of 2000 bits, the RSW-TL approach resolves

the puzzle in approximately 100 seconds, whereas our BM-TL method requires over 4000 seconds, significantly reducing the number of squaring operations needed.

4.1 Introduction

With the birth of the Internet and the beginning of message exchanges, the next problem was that of value exchange. The first projects that aimed to solve ”remote exchange of value" were centralized, since it was not possible to solve the problem of double spending in a decentralized way. Formally, it has been shown that it is possible to create a decentralized randomized Byzantine agreement based on the Bitcoin protocol [86]. Bitcoin was the first system capable of solving this problem in a decentralized manner. Its history can be traced through numerous academic articles, such as the ones on Proof of Work [87, 88] and the one on backlinking to maintain data integrity [89], which all together give rise to the system we nowadays call blockchain. Starting with Bitcoin, various projects have sprung up to solve the new problems that the Bitcoin project presented [90]. One of the first problems was that of scalability. In fact, a decentralized system such as Bitcoin had the ability to operate only 7 transactions per second (tx/s). For this reason, in 2015 a scaling proposal was based on the idea of operating exchanges on parallel blockchains, in order to balance the number of transactions on each blockchain [91]. This idea was further explored in the following years, e. g., in the Liquid [92] and Plasma projects [93].

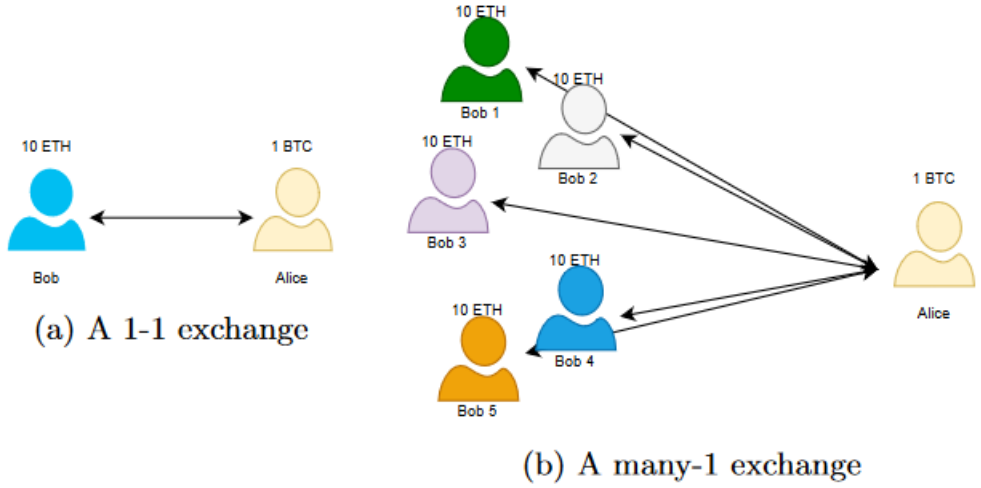

Fig. 4.1 In a 1-1 exchange there are only two parties involved. In a many-1 exchange, multiple parties want to exchange the same amount of funds. In this case a matching algorithm is used to choose Alice's partner. An example of a many-1 exchange are the online (crypto-)currency markets.

As blockchain technology has evolved and diversified, another important problem that has emerged is the need for interaction between distinct distributed systems, commonly referred to as Cross-Chain Communication (CCC) (see Section 2.6). While HTLCs are widely used in the context of CCC, they are not the only approach to achieving synchronization between blockchains. Another alternative is based on *time-lock puzzles*. In particular, the method proposed by Rivest et al. [26] aims to achieve time-bounded encryption: the goal is to obtain a method to make a message remain encrypted only for a predictable amount of time. A time-lock puzzle is also described as a "time capsule" for a message. In our case, the time-lock puzzle is used by Alice and Bob to reveal part of a private key to obtain the counterpart funds while maintaining security. For example, if Alice behaves dishonestly, Bob is able to redeem her funds, knowing that Alice cannot steal them before a certain time defined by Bob himself in the time-lock puzzle. Unfortunately, however, there are few systems to implement a time-lock puzzle other than Rivest's method. In this paper, starting from the preliminary work [94], we propose a new method to obtain time-lock puzzles starting from Rivest's method, laying the groundwork for further study to create new puzzles in different environments. In addition to Asokan's impossibility result, as pointed out in Zamyatin et al [43], most decentralized coin-transfer methods proposed so far in the literature involve only two parties. This works

for a single exchange, but makes it difficult to implement the proposed method in a context similar to online exchanges. In fact, in the case of an online exchange, there are many participants who intend to operate the same exchange, at the same time and at the same rate while competing for first place. This is a many-1 setting, instead of a 1-1 as in the case of Alice and Bob explained before. A visual representation of the difference can be seen in Figure 4.1. Another assumption generally made and implicit in the methods proposed in literature is that these parties already know each other before making the exchange. In reality this is a very strong assumption and that makes it difficult to implement the proposed system retaining the same security guarantees. To face this problem, a centralization tweak is operated during the implementation where the algorithm of connection and knowledge of the two participants is operated on a central server. In the context of markets, this connection algorithm is called *matching algorithm*. It is necessary to study methods that give the possibility to exchange funds in a multitude of contexts without requiring that the participants know each other beforehand, especially at a time when traditional financial methods are moving towards decentralized finance (DeFi). For these reasons, our solution is broadcasted, meaning it deal with multiple participants concurrently in a many-1 environment. Therefore it is a Broadcast Time-Lock Exchange (BTLE).

Contributions. First, we propose an off-line and decentralized matching algorithm. We then introduce a decentralized exchange mechanism, namely one that operates without the assistance of a trusted third party, leveraging time-lock puzzles as a synchronization tool. In addition, we present a novel class of time-lock puzzles based on the Pell conic.e Furthermore, we provide a comparative analysis between the classical time-lock puzzle proposed by Rivest et al. [26] and our construction. We also develop and implement both the legacy and the proposed time-lock puzzles[1]. From a theoretical standpoint, we prove the security of the proposed method within the hybrid ideal/real world simulation paradigm, considering static honest-but-curious adversaries. Finally, we conduct a detailed analysis of the protocol with respect to its time complexity.

[1] https://github.com/disnocen/dex-tlp

4.2 Related Works

This section is divided into three parts. In the first part, we will deal with the presentation of the proposed works in the literature related to the communication between two blockchains. In the second part, we will deal with the study of the presentation of results obtained in time-based mapping, while in the third part we will introduce the works on the hash time lock puzzles. This last part is necessary to make a comparison between our proposed method and the latter method.

4.2.1 Cross Chain Communication

One of the first reports on the CCC issue is the report Buterin wrote in 2016 [44]. At that time the cross-chain communication problem was also called the *interoperability* problem, hence the name in the report. In that report, Buterin classifies the main methods of interoperability present at the time. Generally an interoperable system was leveraged to enhance the scalability of the blockchain in question. Here, scalability means the increase in terms of throughtput (i.e. the number of transactions per second) correlated to an increase in the number of users. In fact, interoperability gives the possibility of rebalancing the loads between two blockchain and therefore split the work between the two. In this sense we talk about sidechains, introduced for the first time in [91]. Another way to find interoperability is to build an interoperable blockchain by design. This is the case of the ecosystems dubbed “blockchain of blockchains”. Examples of these ecosystems are Cosmos [95] and Polkadot [96]. In both projects, the idea is to create a hierarchy of blockchains where each exchange of funds is approved through the blockchain at the head of all others. For a more detailed analysis see, e.g., the work of Zamyatin et al. [43].

4.2.2 Time Release Cryptography

Here we briefly explain what is meant by time-based cryptography. For a general overview of this issue, see the detailed survey by Jaques et al. [97]. In this subsection we present only the fundamental results related to the BTLE protocol. Timed primitives came up in several contexts. In the following we distinguish between a pre-blokchain phase and a post-blockchain phase. The first generation, pre-blockchain, includes “time capsules" for key escrowing as in Bellare et al. [98], time-based

cryptographic secrets as Rivest et al. [26] and contract signing as in Boneh et al. [99]. Of those, only the last two protocols are secure against parallel processing, i.e. they use what has been defined as *inherently secure* function [100]. After the introduction of Bitcoin [101], time based cryptography had a new wave of research. In particular, the post-blockchain study of time-based cryptographic protocols is focused on *verifiable delay functions* (VDF) and *time-lock puzzles* (TLP). VDFs are a subset of TLPs: the difference is that in VDFs other people can verify the solution without the need to solve the puzzle [100]. The majority of the newer studies have focused on VDFs. Some of them proposed new protocols, such as [102], while others extended the TLP in [26] making it a VDF. The works of the latter type are that of Wesolowski [103] and that of Pietrzak [104]. These works are compared in [105]. Interestingly, the new wave of research on time based cryptography has generally left aside the traditional time-lock puzzles: we are aware of only one document on this topic, i.e. the time-lock puzzle in [106]. However BTLE does not need any outside verification of the result. That is because our puzzles have an implicit verification: if the solution is right, then the user can retrieve the funds, otherwise it can not.

4.2.3 Hash time-lock Contracts

To date, the majority of peer-to-peer exchange methods are based on the concept of Hash Time Lock Contract (HTLC). The inventor of HTLCs is considered to be Tier Nolan [47] and they are analyzed in many documents, such as the ones by Herlihy [48] and by Miraz and Donald [107]. An HTLC is a contract that uses hash-based and time-based cryptographic constructs to lock and unlock funds on chain. Participants in this kind of contract have to manually redeem funds by generating cryptographic proofs of payment before a certain date in order proceed with the protocol. The downside of this is that parties are required to stay online during the whole execution of the protocol. This is difficult for power constrained devices or in places where a stable internet connectivity can't be expected. Another downside of HTLCs is that these cryptographic primitives can not be implemented on all blockchains. To obtain the kind of lock required by the protocol a scripting language is necessary. In particular, both blockchain scripting languages need to have primitive to compute the same hash function. Many popular blockchain projects do not satisfy these requirements. For example it wouldn't be possible to make a HTLC between Tezos (which uses the Blake2b function as its principal hash function) and Bitcoin (which

uses the SHA256 function to make hashing). Finally, HTLCs can be instantiated only between two participants or entities: it is a 1-1 exchange. This makes it difficult to base a complete exchange algorithm in decentralized markets with a many-1 setting. In some works, several possibilities have been proposed to overcome some of these limitations, using, for example, AVTC as a cryptographic primitive [108] or leveraging relays and adapters [109]. However, none of these approaches simultaneously addresses all the previous issues, and none of them utilizes time-lock puzzles as a cryptographic primitive, which we instead propose in our BTLE protocol, where for the first time a solution of this kind also involving time-lock puzzles is presented.

4.3 Background

4.3.1 Time lock-puzzle

Starting from the classical time-lock puzzle construction introduced by Rivest, Shamir, and Wagner (RSW-TL) [26] (see Section 1.12.1), we extend this idea to develop a novel time-lock puzzle based on the group structure of conics. We refer to this construction as BM-TL, as it builds upon the RSA-like cryptosystem studied in [110, 111]. Both systems are not parallelizable, because they use sequential functions. Thus, there is no advantage in having more CPUs, because all computations are necessarily done only on one core of the CPU. These time-lock puzzles can be considered as a CPU-bound puzzle with a timing function and an implicit verification [112].

The idea of Rivest et al. for constructing time-lock puzzles can be naturally generalized by replacing modular exponentiation with analogous operations defined over different algebraic structures. In particular, given a field $\mathbb{F}$, the conic

$$\mathscr{C} = \{(x,y) \in \mathbb{F} \times \mathbb{F} : x^2 - Dy^2 = 1\},$$

with $D > 1$ square-free, can be equipped with the product

$$(x_1,y_1) \otimes (x_2,y_2) = (x_1x_2 + y_1y_2D, x_1y_2 + x_2y_1),$$

so that $(\mathscr{C}, \otimes)$ is an abelian group with identity $(1,0)$ and the inverse of an element (x,y) is $(x,-y)$. Considering $x^2 - D$ an irreducible polynomial over $\mathbb{F}$, one can easily observe that $\mathscr{C}$ is isomorphic to the elements of norm 1 of the field $\mathbb{A} := \mathbb{F}[X]/(x^2 - D)$. Then we can parametrize the conic by considering $P = \mathbb{A}^*/\mathbb{F}^* \cong \mathbb{F} \cup \{\alpha\}$, where $\alpha \notin \mathbb{F}$ is the point at the infinity. In this way, the induced product over P is given by

$$\begin{cases} a \odot b = \dfrac{D + ab}{a + b}, & \text{if } a + b \neq 0 \\ a \odot b = \alpha, & \text{if } a + b = 0 \end{cases}.$$

It is interesting to observe that the same parametrization can be obtained by using the line $y = \dfrac{1}{m}(x+1)$.

Considering $\mathbb{F} = \mathbb{Z}_p$ and D not a quadratic residue modulo p, then $\mathscr{C}$ is a cyclic group of order $p+1$ and thus

$$a^{\odot p+1} \equiv 1 \pmod{p}$$

for every $a \in \mathbb{Z}_p$, where the powers are evaluated with respect to the product $\odot$. Moreover, we can also define the conic $\mathscr{C}$ with points whose coordinates belong to the ring $\mathbb{Z}_n$. In this case, we do not have a group structure, however, considering $P = \mathbb{Z}_n \cup \alpha$, with $n = pq$, p and q primes, we have an analogue of the Euler's theorem:

$$a^{\odot \Psi(n)} \equiv 1 \pmod{n}, \quad \forall a \in \mathbb{Z}_n^*,$$

where $\Psi(n) = (p+1)(q+1)$ plays the role of the Euler totient function, see [110]. Now, we have all the tools for defining the BM-TL following the idea of Rivest et al. Indeed, the secret sk can be encrypted by

$$c \equiv sk + a^{\odot 2^t} \pmod{n}.$$

Knowing the factorization of n, one can efficiently compute $a^{\odot 2^t}$ evaluating first $e \equiv 2^t \pmod{\Psi(n)}$ and then $a^e \pmod{n}$. Without knowing the factorization of n, one must perform t squarings with respect to the product $\odot$.

4.3.2 Verifiable Delay Function

In the following we explain how it is possible to extend the timelock puzzles into a Verifiable Delay Function (VDF). To do that we apply the method described by Wesolowsky [103] to the BM-TL conic.

We say that a Verifiable Delay Function (VDF) implements a function $\mathscr{X} \to \mathscr{Y}$ if it is a tuple of three algorithms [105]:

- **Setup**$(\lambda, t) \to pp$ is a randomized algorithm that takes a security parameter λ and a time bound t, and outputs public parameters pp,
- **Eval**$(pp, x) \to (y, \pi)$ takes an input $x \in \mathscr{X}$ and outputs a $y \in \mathscr{Y}$ and a proof π.
- **Verify**$(pp, x, y, \pi) \to \{$ accept, reject $\}$ outputs accept if y is the correct evaluation of the VDF on input x.

The work in [103] uses the **Setup** and **Eval** phases already described in [26] and Section 4.3. We are going to describe how to extend its **Verify** phase. There are two possible types of proofs, one interactive and one non-interactive. We start by explaining the interactive one. If Primes(λ) is the set of prime numbers before λ and $h = g^{2^t}$, its steps are:

0. The verifier sends to the prover a random prime ℓ sampled uniformly from Primes(λ),
1. The prover computes $q, r \in \mathbb{Z}$ such that $2^t = q\ell + r$ with $0 \leq r < \ell$, and sends $\pi \leftarrow g^q$ to the verifier.
2. The verifier computes $r \leftarrow 2^t \pmod{\ell}$ and outputs accept if $h = \pi^l g^r$

This conclude the interactive proof. The non interactive proof uses the Fiat-Shamir heuristic in the random oracle model.

Since the creation and solution of $2^t \pmod{n}$ is the same for both the RSW-TL and BM-TL methods, the work by Wesolowsky [103] naturally extends the BM-TLP to a BM-VDF.

4.3.3 Conditional Transactions

The blockchains that have a scripting language allow users to create imperative (as in the case of Ethereum) or verifying (as in the case of Bitcoin) smart contracts. One application of this fact is the construction of conditional transactions, i.e. transactions that can be redeemed by different keys under certain conditions.

In the case of BTLE-AS we use transactions that use time as a condition (measured in terms of blocks). In particular we define a (t_1, t_2)-transaction, or transaction with parameters (t_1, t_2), a transaction such that;

- it can be redeemed by a key K_1 before time t_1
- it can be redeemed by a key K_2 after time t_1 but before time t_2
- it can be redeemed by a key K_3 after the time t_2

Specifying how to create a conditional transaction is outside the scope of the document, but examples of this type of transaction can be found in the BIP16 [113] specification for Bitcoin, but are possible in any blockchain project that has a scripting language. We use this kind of transaction in BTLE-AS (see Section 4.4.2) where, unless otherwise specified, the keys K_1 and K_3 are the same).

4.4 Design

In this section we explain how it is possible to create both a matching algorithm and an atomic swap algorithm based on time-lock puzzles. Both algorithms are part of the Broadcast Time-Lock Exchange Protocol (BTLE) and for simplicity we call BTLE-MA the matching algorithm and BTLE-AS the swap algorithm. Without loss of generality, we can say that the participant that initiates the process is selling its token in exchange of (or *to buy*) the other. We point out that it is not necessary to use both the BTLE-MA and the BTLE-AS algorithms to obtain a token swap. In particular, it is possible to add to the current swap methods based on HTLC a matching algorithm based on BTLE-MA. This facilitates the partner discovery steps in current atomic swap systems. In the following, we assume a decentralized platform where participants want to swap tokens. Each participant owns a client that can track the progress of the reference blockchain. We informally call *view*

the client's collection and ordering of states of the blockchain. This view is not necessarily the same for all participants due to network lag or possible attacks, such as an Eclipse attack that has the effect of giving a false view of the blockchain blocks. Here "false" means that the victim's view is different from the view of the majority of blockchain miners/validators. In the following we will not focus on the consequences of this attack because it has no real consequences: if the victim has a different view of the blockchain, then it is very difficult for her to have transactions accepted by miners. The existence of different views explains why BTLE-MA is necessary: there cannot be a temporal-based ordering of possible buyers and an asynchronous systems imply the absence of a shared clocks. We distinguish two classes of participants. The first is the class of *initiators*: this kind initiates the exchange by proposing the deal, i.e. the selling of its token. The initiator corresponds to a *market makers* in traditional centralized markets. The latter is the class of *exchangers*: to this class belong the possible buyers interested in an equivalent and opposite deal but that do not want to start it. These participants correspond to *market taker*. We assume there is a single initiator which we denote by A (Alice), and many possible exchangers. Since the exchangers represent Alice's partner, following the cryptography tradition we will call them all "Bobs" and, supposing they are indexed and in finite number d, we denote them as $\{B_1, B_2, \ldots, B_d\}$ and we assume d secure channels of communication between A and each one of $B_i, i = 1 \ldots d$. We also assume $\{B_1, B_2, \ldots, B_d\}$ compete at the same price level, as in traditional order books. If not, then A narrows the view to the subset of Bobs with the most favorable exchange rate for A. Note that the set $\{B_1, B_2, \ldots, B_d\}$ is completely determined by the view of A since we assume that A's view of the blockchain is the correct one. Consequently there may be other potential exchangers or some have withdrawn and A's view of potential buyers is not synchronized yet. In the protocol description we will see why neither of these two cases is a problem. Finally, in the following we treat the time-lock puzzle TLP as a blackbox which takes two inputs and then outputs a cryptographic puzzle, thanks to how we modeled the time-lock puzzle primitive in Section 4.3. The solution of the puzzle is the cleartext itself. This shows that in theory the system can use any kind of time lock puzzle, including those described in Section 4.3. In Section 4.6 we will see which one is better from a practical point of view.

4.4.1 BTLE-MA

In this round Alice has to choose the exchanger among $\{B_1, B_2, \ldots, B_d\} \leftarrow GetExchangersList()$. Here $GetExchangersList()$ is a routine on the platform that returns the addresses of the exchangers and a way to communicate with them. How to implement it is outside the scope of this document.

BTLE-MA is explained in pseudo-code in Algorithm 1. As seen in the figure, A starts by generating a random message $rand_i, i = 1 \ldots d$ for each participant in $\{B_1, B_2, \ldots, B_d\}$. Then A associates this message to the intended receiver, $rand_i \leftrightarrow B_i, i = 1, \ldots, d$. The inputs of each time-lock puzzle are the message $rand_i$ and a time in seconds. Note that in a time-lock puzzle $TB_i = TLP(rand_i, time)$, the random message is different for each B_i, but $time$ is equal for all participants: all potential exchangers must have the same chance of being able to find the solution at the same time. The randomness of the winner is determined by unpredictable factors such as network latency or the puzzle real solving time. In this sense the choice of B_j is truly random. The output is a tuple that represent the cryptographic puzzle (See Section 4.3). A performs this subroutine for all $B_i, i = 1, \ldots, d$ and then it sends all the puzzles. Finally A waits for a solution.

When A receives the first solution (i.e. the cleartext of the random message) $\widehat{rand_j}$ from some B_j, A checks that it is a valid message. Practically, this means that A verifies that the cleartext $\widehat{rand_j}$ is equal to the message $rand_j$ associated with B_j. If that is the case, A accepts the solution and B_j is the winner: from now on A will proceed to communicate only with B_j and discards all other solutions. If the message isn't valid, A waits for another solution.

We stress the fact that the entirity of the BTLE-MA routine runs off-chain. Therefore it isn't costly nor slow.

4.4.2 BTLE-AS

Let Bob be the winner of BTLE-MA and we denote it as B. This means that A will deal only with B, as in a classical token exchange. The goal of the parties in BTLE-AS is exchange their tokens in an atomic way.

Algorithm 1 BTLE-MA routine

1: $\{B_1, B_2, \ldots, B_d\} \leftarrow GetExchangersList()$
2: **for** $i = 1, \ldots, d$ **do**
3: $rand_i \leftarrow RandGen()$
4: $map(rand_i \leftrightarrow B_i)$
5: $TB_i \leftarrow TLP(rand_i, time)$
6: **end for**
7: **for** $i = 1, \ldots, d$ **do**
8: $Send : TB_i \longrightarrow B_i$
9: **end for**
10: $AcceptedSolution =$`False`
11: **while** $AcceptedSolution =$`False` **do**
12: $waitSolution()$
13: **if** $VerifySolution(sol) ==$`True` **then**
14: $AcceptedSolution =$`True`
15: $WinnerSolution = sol$
16: **end if**
17: **end while**
18: $Winner = map(WinnerSolution)$
19: return $Winner$

In the following we present the notation used to understand the protocol. At the beginning of BTLE-AS, A owns 1 `coin1`, and B owns 1 `coin2`[2]. To enforce the atomicity of the protocol, we consider the ending of the BTLE-AS protocol as "successful" only if one of this two possible endings occurs:

1. A has 1 `coin2` B has 1 `coin1`
2. A has 1 `coin1` B has 1 `coin2`

Case 1. means that both A and B were honest during the execution of the protocol, while Case 2. happens if either A or B has been dishonest or faulty.

Intuitively BTLE-AS aims to exchange private keys between A and B so that they can move their newly acquired fund to other addresses. Of course, simply exchanging private keys would easily lead to possible theft: if A is not honest, A can still use his key to move his funds even if it sent the key to B, effectively stealing B's

[2]The real exchange rates between the two tokens and how they are decided are beyond the scope of this document. We assume this kind of information can be exchanged either in the channel during BTLE-MA or the UI implementation of the protocol.

funds. Therefore we split a private key into two parts and we use the homomorphism of both the sum and external product in the elliptic curve group. Formally, given two secret keys sk^A, sk^B, an elliptic curve generator g and the relative public key pk^A and pk^B, then the key sum is homomorphic:

$$(sk^A + sk^B)g = pk^A + pk^B = (sk^A g) + (sk^B g). \tag{4.1}$$

Beside not sending the time-lock puzzle, A has another way to cheat B. In fact, A could create and send to B the time-lock puzzle of a random number instead of its private key sk_A. To avoid this problem, we introduce a zero-knowledge proof of the fact that the time-lock puzzle is hiding the right secret key, of course without revealing it.

In the following we introduce the swap and the proof protocols.

The Swap

We explain the exchange of tokens `coin1` and `coin2` with reference to Table 4.2, with notation in Table 4.1.

At first, A and B create the key pairs sk_2^A, pk_2^A and sk_1^B, pk_1^B respectively, where sk_i^j, pk_i^j are related to blockchain $BC_i, i = 1,2$ and user $j = A,B$. These key pairs are respectively the first of the two shares of A and B to redeem the exchanged funds. After this step, A and B exchange the public keys pk_2^A and pk_1^B. If this first part is successful, both A and B create ephemeral keys (sk_1^A, pk_1^A and sk_2^B, pk_2^B respectively) that represent the second of the two shares to redeem the exchanged funds. At this point A and B can create new public keys (called PK_1^B and PK_2^A respectively) to which they can send the funds. This transaction is a $(\frac{\lambda}{2}, t)$-conditional transaction as explained in Section 4.3.3where t is a generic deadline the participants A and B can agree on or can be included in a specification. Because of the way the keys PK_1^B and PK_2^A and consequently the addresses are built, neither of the two participants can redeem the funds in either blockchains at this point of the exchange. For example, A needs to know sk_2^B in order to redeem coins in the address for PK_2^A. For this reason in the second part of the exchange A and B exchange the time-lock puzzles TLP_A and TLP_B and their respective proofs, explained in Section 4.4.2 . Once the time-lock

BC_1,BC_2	Blockchain 1 and 2 with tokens `coin1` and `coin2`
G_1,G_2	the base point for the elliptic curve of BC_1 and BC_2
l_1,l_2	the base point order for the elliptic curve of BC_1 and BC_2
PK_1^A	public key for the address on blockchain BC_1 where A has the coins
PK_1^B	public key for the address on blockchain BC_1 where B receives the new coins
PK_2^A	public key for the address on blockchain BC_2 where A receives the new coins
PK_2^B	public key for the address on blockchain BC_2 where B has the coins
$sk_{1,2}^A, pk_{1,2}^A$	shares created by A for blockchain $BC_{1,2}$
$sk_{1,2}^B, pk_{1,2}^B$	shares created by B for blockchain $BC_{1,2}$

Table 4.1 Notation used in the explanation of the token exchange protocol

puzzles are opened/solved, A gets sk_2^B and B gets sk_1^A. Using λ as time unit, we observe that the second time-lock puzzle is sent later (by $1/4$ of the time unit) with a shorter opening time (i.e., $3/4$ of the time unit). This ensures that both participants, A and B, open the puzzle at about the same time. As an example, consider the conditional transaction sent from Bob to Alice:

$$\text{CondTx}\left(\frac{\lambda}{2}, t, PK_2^B \rightarrow PK_2^A\right)$$

This conditional transaction is redeemable under the following conditions:

1. By the holder of the secret key associated with PK_2^B before time $\frac{\lambda}{2}$;
2. By the holder of the secret key associated with PK_2^A after time $\frac{\lambda}{2}$ but before time t;
3. By the holder of the secret key associated with PK_2^B after time t.

The mechanism ensures that the conditional transaction operates correctly under the following conditions:

- If Alice does not send her conditional transaction before time $\frac{\lambda}{4}$, Bob retains the ability to redeem his transaction until time $\frac{\lambda}{2}$, thereby safeguarding his funds.

Alice (coin1→coin2)		Bob (coin2→coin1)
$sk_2^A \xleftarrow{\$} [1, l_2 - 1], pk_2^A = sk_2^A G_2$		$sk_1^B \xleftarrow{\$} [1, l_1 - 1], pk_1^B = sk_1^B G_1$
$rm_A \leftarrow GenRandomMessage()$		$rm_B \leftarrow GenRandomMessage()$
	$\xrightarrow{pk_2^A, rm_A}$	
	$\xleftarrow{pk_1^B, rm_B}$	
$sk_1^A \xleftarrow{\$} [1, l_1 - 1], pk_1^A = sk_1^A G_1$		$sk_2^B \xleftarrow{\$} [1, l_2 - 1], pk_2^B = sk_2^B G_2$
$PK_1^B = pk_1^A + pk_1^B = (sk_1^A + sk_1^B) G_1$		$PK_2^A = pk_2^A + pk_2^B = (sk_2^A + sk_2^B) G_2$
$hash_{A \to B} \leftarrow$SendCondTx$(\frac{\lambda}{2}, t, PK_1^A \to PK_1^B)$		$hash_{B \to A} \leftarrow$SendCondTx$(\frac{\lambda}{2}, t, PK_2^B \to PK_2^A)$
		$T_B \leftarrow TLP_B(sk_2^B, \lambda)$
		$\pi_B \leftarrow$ **ProofGen**(sk_2^B, rm_A)
	$\xleftarrow{T_B, hash_{B \to A}, \pi_B}$	
$T_A \leftarrow TLP_A(sk_1^A, \frac{3}{4}\lambda)$		
$\pi_A \leftarrow$ **ProofGen**(sk_1^A, rm_B)		
	$\xrightarrow{T_A, hash_{A \to B}, \pi_A}$	
$pk_2^B = PK_2^A - pk_2^A$		$pk_1^A = PK_1^B - pk_1^B$
ProofVer(rm_A, pk_2^B, π_B)		**ProofVer**(rm_B, pk_1^A, π_A)
open T_B and redeem coin2		open T_A and redeem coin1

Table 4.2 Protocol execution between Alice and Bob for a successful swap.

- If Alice sends her conditional transaction before time $\frac{\lambda}{4}$, she is granted a designated window to redeem Bob's funds, specifically between time $\frac{\lambda}{2}$ and t. After time t, if Alice has not redeemed the funds, Bob is permitted to withdraw them, thereby ensuring a protective measure against potential delays or malicious intent on Alice's part.

These considerations are similarly applicable to the transaction initiated by Alice towards Bob.

The Proof

In the previous section we saw how the exchange works, and we treated the proof as a blackbox. In this section we explain in details how the proof works. Here we explain how to do this assuming that the users will use one of the two time-lock puzzles described in Section 4.3. In particular we assume that for each protocol there exist two functions **ProofGen**() and **ProofVer**() such that:

- **ProofGen**$(sk, rand)$ is a deterministic algorithm that, given a secret key sk and a random message $rand$, creates a proof π.
- **ProofVer**(π, pk) is a deterministic algorithm that, given the public key pk with respect to sk and the proof π, outputs 1 if π is valid and 0 otherwise.

We start with the explanation of **ProofGen**. In case of the RSW-TL and the BM-TL time-lock puzzles, c is obtained from the formula

$$c = sk + a^{2^t}, \quad c = sk + a^{\odot 2^t}$$

respectively, where sk is the message (the secret key) the user wants to encapsulate in the puzzle, a a random number and t is the number of squarings.

To generate the proof, we leverage the fact that digital signatures ensure unforgeability, meaning that only the holder of the private key can produce a valid signature for a given message. Unforgeability does not necessarily imply that no information about the private key can be leaked (unlike zero-knowledge proofs, which guarantee no leakage at all), as some schemes may tolerate partial leakage without compromising security. However, in our scenario, the unforgeability of the signature is sufficient to ensure the validity of the proof, as only the correct holder of the private key can generate a valid signature. Thus, let **SigGen**(m, s) be the signature routine of a digital signature scheme such that given a message m and a secret key s it outputs a signature σ, and let **SigVer**(m, σ, S) the respective verification algorithm such that given a message m, a public key S and a signature σ outputs `True` if σ is the signature of message m, and `False` otherwise. Finally let P a prover and V a verifier, similar to every zero-knowledge protocol. Then the protocol works in the following way and the case of the RSW-TL can be seen in Algorithm 2.

Algorithm 2 ProofGen(sk, rm)

1: $c = sk + a^{2^t}$
2: $A \leftarrow$ **PubkeyGen**(a^{2^t})
3: $\sigma \leftarrow$ **SigGen**(rm, c)
4: $\pi = (\sigma, A)$
5: $Send(\pi)$

Assume there is an algorithm $X =$ **PubkeyGen**(x) that given a number x considers x as a ephemereal secret key and outputs the relative ephemereal public

key X. When P creates the time-lock puzzle c, it also computes the public key $A = \textbf{PubkeyGen}(a^{2^t})$ in the case of RSW-TL, or $A = \textbf{PubkeyGen}(a^{\odot 2^t})$ in the case of BM-TL. Then P computes $\sigma = \textbf{Sig}(rm, c)$, a signature of the random message rm using the time-lock puzzle result c acting as ephemereal secret key. The proof is $\pi = (\sigma, A)$. This concludes the **ProofGen** routine.

Upon receiving the time-lock puzzle an the proof π, V starts the **ProofVer** routine as seen in Algorithm 3 . V first checks that the proof works. To do that, V uses the public key $pk + A$ to verify that the message signed in σ is rm (recall that V already has pk as per the swap algorithm) using the $\textbf{SigVer}(\sigma, pk + A)$ routine. This works because $c = sk + a^{2^t}$, pk is the public key corresponding to sk and $A = \textbf{PubkeyGen}(a^{2^t})$ by construction. Consequently, using the property of homomorphism of the sum in elliptic curve cryptography, we have:

$$PubKey = pk + A = g \cdot sk + g \cdot a^{2^t} = g(sk + a^{2^t}) = g \cdot c$$

If the verification checks, then V is sure P's time-lock puzzle is correctly built and then proceeds in solving it.

Algorithm 3 ProofVer(rm, pk, π)

1: $\sigma = \pi[0]; A = \pi[1]$
2: $Pubkey = pk + A$
3: $Bool \leftarrow \textbf{SigVer}(rm, \sigma, Pubkey)$
4: **if** $Bool ==$ `True` **then**
5: $\quad Accept\ \pi$
6: **else**
7: $\quad Reject\ \pi$
8: **end if**

4.5 Security

The security of the proposed scheme relies on the difficulty of factoring large composite numbers, a well-known cryptographic assumption referred to as the *factoring assumption*. In this section, we provide a security proof in the hybrid ideal/real-world simulation model, which considers a *static honest-but-curious* adversary $\mathscr{A}$. In this context, *static* means that the adversary is given a fixed set of parties to control from the outset; honest parties remain honest, and corrupted parties remain corrupted

throughout the protocol. The *honest-but-curious* (or semi-honest, or passive) adversarial model assumes that even corrupted parties correctly follow the protocol specifications without deviation. However, the adversary gains access to the internal state of all corrupted parties, including the transcript of all messages received during the protocol execution. The adversary then attempts to use this information to derive data that should remain private.

To formalize the problem, a two-party protocol is defined as a random process that maps pairs of inputs to pairs of outputs, one for each party. This process is referred to as a *functionality* and is represented as:

$$f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*,$$

$$(x,y) \mapsto (f_1(x,y), f_2(x,y))$$

The first party, with input x, aims to obtain $f_1(x,y)$, while the second party, with input y, aims to obtain $f_2(x,y)$.

We begin with the following notation:

- Let $f = (f_1, f_2)$ be a probabilistic polynomial-time functionality, and let π be a two-party protocol for computing f.
- The view of the i-th party ($i \in \{1,2\}$) during an execution of π on input (x,y) and security parameter λ is denoted by $\mathsf{view}^i_\pi(x,y,\lambda)$.
- The output of the i-th party ($i \in \{1,2\}$) during an execution of π on input (x,y) and security parameter λ is denoted by $\mathsf{output}^i_\pi(x,y,\lambda)$.
- The joint output of both parties is denoted by:

$$\mathsf{output}_\pi(x,y,\lambda) = (\mathsf{output}^1_\pi(x,y,\lambda), \mathsf{output}^2_\pi(x,y,\lambda)).$$

We now present the following definition from [114]:

Definition 4.5.1 (Security in the presence of honest-but-curious adversaries) *Let $f = (f_1, f_2)$ be a functionality. We say that π securely computes f in the presence of static honest-but-curious adversaries if there exists a probabilistic polynomial-time simulator $\mathscr{S}$ such that:*

$$\{(\mathscr{S}(1^\lambda, x, f_1(x,y)), f(x,y))\}_{x,y,\lambda} \overset{c}{\equiv} \{(\mathsf{view}_\pi^1(x,y,\lambda), \mathsf{output}_\pi(x,y,\lambda))\}_{x,y,\lambda},$$

$$\{(\mathscr{S}(1^\lambda, y, f_2(x,y)), f(x,y))\}_{x,y,\lambda} \overset{c}{\equiv} \{(\mathsf{view}_\pi^2(x,y,\lambda), \mathsf{output}_\pi(x,y,\lambda))\}_{x,y,\lambda},$$

for all $x, y \in \{0,1\}^*$ such that $|x| = |y|$, and $\lambda \in \mathbb{N}$.

To prove security, we show that the ideal model, presented in Algorithm 4, can simulate the execution of the real-world protocol BTLE-AS. We refer to the ideal model as BM-AS-IDEAL, which operates with parties in the set $\mathscr{P} = \{A, B\}$. We restrict our analysis to scenarios with a single malicious party. We denote the corrupted party, controlled by the adversary $\mathscr{A}$, as P_I, and the honest party as P_J. Finally, *sid* denotes a unique session identifier, which ensures that each protocol execution is distinct, preventing replay attacks and preserving the integrity of the communication.

In Algorithm 4, the functionality f can be interpreted as a comprehensive set of operations performed by the Trusted Third Party, which includes:

- The secure generation and distribution of public-private key pairs.
- The verification and validation of the transaction hash (*txhash*).
- The secure generation and distribution of Time Lock Puzzles TLP_P along with their corresponding proofs π_P.

Moreover, to ensure proper synchronization of timing-related actions, especially in the context of cross-chain interactions, we introduce a *global clock model*. This model is specifically tailored to provide a unified, blockchain-native time reference for participants operating across different blockchains. Finally, for the purposes of this analysis, we assume that the devices used by the parties are not compromised and securely manage all cryptographic operations, such as storing private keys and generating random values.

Algorithm 4 BM-AS-IDEAL

1: Upon receiving **Setup**$(sid, l_1, l_2, G_1, G_2)$ from some party $\mathscr{U} \in \mathscr{P}$, check that *sid* hasn't been previously used and
   - Generate sk_i^P, pk_i^P, rm_P for $i = 1,2$ and $p \in \mathscr{P}$
   - Send $pk_2^A, rm_A, sk_1^B, pk_1^B, PK_2^A$ to B
   - Send $pk_1^B, rm_B, sk_2^A, pk_2^A, PK_1^B$ to A
2: Upon receiving **Clean**$(sid, P \in \mathscr{P}, txhash, \lambda, t)$ from some party $\mathscr{U} \in \mathscr{P}$, if t has not passed and
3: **if** second time *sid* used and *txhash* is correct for party P **then**
4: Start *time*
5: Publish to the other party $\bar{P}$:
   - $T_P \leftarrow TLP_P(sk_i^P, \lambda)$ with $i = 2$ if $P == B$, $i = 1$ otherwise
   - $\pi_P \leftarrow$ **ProofGen**$(sk_i^P, rm_{\bar{P}})$
   - *txhash*
6: **else**
7: **if** third time *sid* used and *txhash* is correct for party P and $time < \frac{\lambda}{4}$ **then**
8: publish to the other party $\bar{P}$:
   - $T_P \leftarrow TLP_P(sk_i^P, \frac{3}{4}\lambda)$ with $i = 2$ if $P == B$, $i = 1$ otherwise
   - $\pi_P \leftarrow$ **ProofGen**$(sk_i^P, rm_{\bar{P}})$
   - *txhash*
9: **end if**
10: **end if**

.

4.5.1 Global Clock Model

In the context of cross-chain atomic swaps between Blockchain_A and Blockchain_B, we address the need for a unified time reference for both participants, Alice and Bob, by proposing a *global clock model*. This model operates based on block counts rather than real-world time, providing a blockchain-native mechanism to synchronize actions across chains. Alice and Bob determine that a specified time interval has passed by observing the insertion of a predefined number of blocks.

Let:

- T_A denote the average block insertion time on Blockchain_A,
- T_B denote the average block insertion time on Blockchain_B.

Since the global clock is based on the slower of the two blockchains, we assume, without loss of generality, that Blockchain_A is the slower blockchain. Consequently, the global clock operates using Blockchain_A as its reference.

At the start of the swap process, let h_k^A and h_l^B represent the block heights of Blockchain_A and Blockchain_B, respectively. We define the key time parameters as follows:

- A transition of λ units of time corresponds to the insertion of the block at height $h^A_{k+4\lambda}$.
- A transition of t units of time corresponds to the insertion of the block at height h^A_{k+4t}.

These definitions use a coefficient of 4 to enable Alice and Bob to check for intermediate time milestones, specifically at $\frac{\lambda}{4}$ and $\frac{3\lambda}{4}$. This ensures that even if λ is not divisible by 4, there is a corresponding block insertion for each fractional milestone, allowing precise synchronization.

Although each blockchain maintains its independent block count, the global clock increments in sync with the block count of Blockchain_A. This ensures that both Alice and Bob remain synchronized in their actions throughout the atomic swap process.

To implement this global clock model effectively:

- Both Alice and Bob must have active addresses on both Blockchain_A and Blockchain_B.
- Each participant must be capable of monitoring block insertions on both blockchains, either through node connections or reliable notifications.

Thus, for reliable synchronization, we also assume that the faster blockchain (Blockchain_B) is notified each time a new block is added to Blockchain_A. This ensures a decentralized, blockchain-native timing system tailored for this specific application.

4.5.2 Proof of security

We are now ready to proceed with the proof of the following theorem:

Theorem 4.5.1 (Security of the BM-AS Protocol) *Let the functionality BM-AS-IDEAL and protocol BM-AS be as defined in Definition 4.5.1. The BM-AS protocol securely computes the BM-AS-IDEAL in the presence of one honest-but-curious adversary.*

Proof. Let $\mathscr{A}$ be a static honest-but-curious adversary. We construct a simulator $\mathscr{S}$ who invokes $\mathscr{A}$ internally to simulate the execution of the real protocol, while interacting with BM-AS-IDEAL in the ideal world.

We assume that Bob (P_I) is the corrupted party controlled by $\mathscr{A}$, while Alice (P_J) is the honest one (the case where Alice is corrupted and Bob is honest can be handled analogously). This assumption is reflected in the simulation steps below, where $\mathscr{S}$ replicates the actions of $\mathscr{A}$ in the real protocol.

The simulator $\mathscr{S}$:

1. $\mathscr{S}$ sends $\textbf{Setup}(sid, l_1, l_2, G_1, G_2)$ to BM-AS-IDEAL and receives the keys and random message.
2. $\mathscr{S}$ simulates the sending of the conditional transaction with parameters $(\frac{\lambda}{2}, t)$ and publishes the output $txhash_I$ to P_J.

3. $\mathscr{S}$ simulates TLP_I and $\textbf{ProofGen}_I$ and publishes the output to P_J.

4. $\mathscr{S}$ monitors the chain until the following occurs:

   (a) P_J sends $txhash_J$, TLP_J, and $\textbf{ProofGen}_J$ before time $\frac{\lambda}{4}$:
   P_I opens TLP_J and redeems the money from the transaction with hash $txhash_J$.

From the standpoint of $\mathscr{A}$, the simulated and real executions are identical:

- Step 2 in simulation is parallel to Step $7, 10$ of BM-AS protocol
- Step 3 in simulation is parallel to Step $8, 9, 10$ of BM-AS protocol
- The creation of the conditional transaction is the same
- Step 4(a) in the simulation is parallel to the Step $13, 16$ in BM-AS

$\mathscr{S}$ measures the times λ and t using the global clock model presented in 4.5.1. Thus, we have proved that for any adversary $\mathscr{A}$ in the real model, there exists a simulator $\mathscr{S}$ in the ideal model that can effectively simulate the execution of the real protocol, ensuring that both the adversary's view and the joint outputs of the honest and corrupted parties in the real-world execution are computationally indistinguishable from those in the ideal-world execution. Consequently, the BM-AS protocol securely computes the BM-AS-IDEAL functionality in the presence of one honest-but-curious adversary, as stated in the theorem.

Remark on Sybil Attacks: A potential challenge for the proposed protocol is its susceptibility to Sybil attacks, where malicious actors create multiple fake identities to disrupt the matching process or exploit time-lock puzzles without completing exchanges. However, we assume that adversaries are rational and aim to maximize their gains. In this context, Sybil attacks are inefficient, as splitting computational resources across multiple identities reduces the ability to solve time-lock puzzles effectively, thereby lowering the adversary's potential reward. While this work does not include specific mechanisms to resist Sybil attacks, exploring such protections could be an interesting direction for future research.

Remark on post-quantum primitives: While it is well known that the factoring assumption is susceptible to quantum attacks via Shor's algorithm, this assumption

remains justified by its current practical security within blockchain interoperability. Despite the rapid advancements in quantum computing, the deployment of post-quantum cryptographic schemes is still an ongoing process, and many standardized blockchain infrastructures continue to rely on pre-quantum assumptions, particularly elliptic curve cryptography (ECC) for key generation and digital signatures. Since ECC is not quantum-resistant, existing blockchain systems will require substantial upgrades to maintain security in a post-quantum era. Quantum threats extend beyond factoring-based schemes, as highlighted in [115], which discusses the vulnerabilities of both public-key cryptography and hash functions against quantum attacks, including those based on Shor's algorithm. This underscores the need for quantum-resistant cryptographic primitives in blockchain ecosystems.

Finally, an interesting direction for future research would be to extend the current security proof to the malicious adversary model, addressing potential deviations from the protocol and demonstrating robustness against actively adversarial behaviors.

4.6 Discussion

In this section, we analyze the exchange protocol considering three points of view. The first is that of security, then we will look at the protocol from a fee point of view and finally we will look at the speed of the protocols. We conclude that the best time-lock puzzle protocol for this purpose is the time-lock puzzle based on the Bellini-Murru protocol (BM-TL). Both the time-lock puzzles are based on already analized methods. In particular, both the RSW and BM method derive their security from the difficult problem of number factorization. BTLE is competitive with the classic atomic swap in terms of fees. This is because no on-chain operations are required: the exchange of random messages in BTLE-MA, the exchange of private key shares in BTLE-AS and the computation needed for the generation an resolution of the time-lock puzzles are done off-chain. The only transaction per blockchain is the transaction that participants make to send funds to another address. This is an advantage over the number of transactions required in an HTLC.

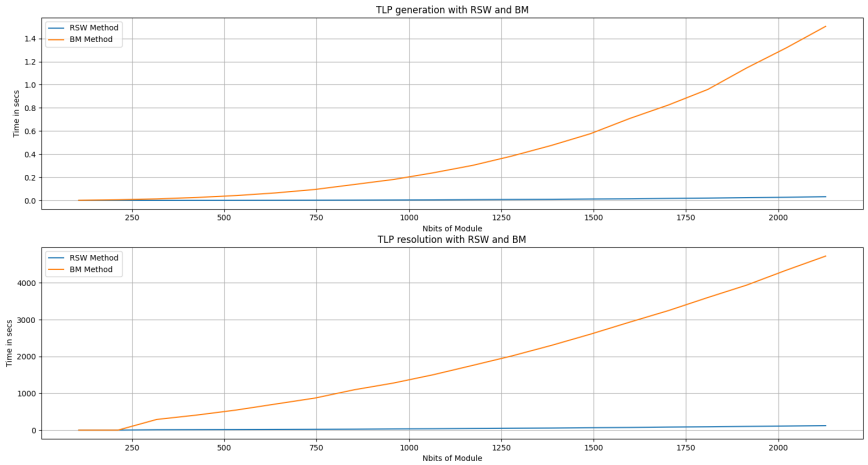

Fig. 4.2 Comparison of generation (above) and resolution (below) times of the time-lock puzzles in the RSW-TL and BM-TL cases. As seen in figure, for a scenario like the one presented in this article, on a practical level only BM-TL can be used.

Another advantage of BTLE over HTLC is that the timelock puzzle system is more flexible. While in an HTLC it is necessary to decide the waiting time between transactions as a function of the timing of the creation of blocks in both blockchains, in BTLE the timing can be shorter since BTLE does not have to do with blocks (there are no transactions sent other than the funding transaction if both parties are honest), but the only delays are related to the latency of the internet network, which are obviously much shorter We conclude this section with advice on choosing a time-lock puzzles between RSW-TL and BM-TL based on the tests we conducted. As seen in the comparison in Figure 4.2, using a squaring number $t = 10^7$ the BM-TL takes much longer than the RSW-TL method. In particular it can be seen how with a number N such that its length in bits is about 2000 bits, the resolution time of the time-lock puzzle is $\sim$100 seconds in the case of RSW-TL , while it exceeds 4000 seconds (just over an hour) in the BM-TL case. Figure 4.3 highlights this result.

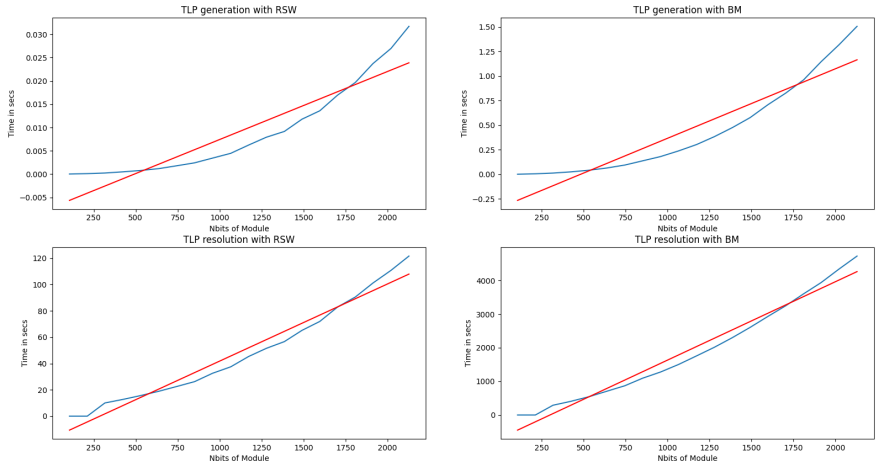

Fig. 4.3 Comparison of generation (above) and resolution (below) times of the RSW (left) and BM (right) time-lock puzzles. It is clear that the growth in terms of seconds with respect to the magnitude in bits of the modulo N is not linear. We also note that the times in the BM-TL case are about 30 times greater than the times in RSW-TL.

Furthermore, the growth in terms of seconds in both generation and resolution of the time-lock puzzle is not linear in terms of the number of bits in the modulo N. This makes it more practical to use BM-TL as fewer total squarings are required. If participants want to wait less than an hour, they can use a N module of less than 2000 bits or decrease t. Test results can be found in the Github repository[3]. There, it is possible to find the results of the tests made with both BM-TL and RSW-TL, on a core of different machines (Apple M1, AMD Series 7, Intel Xeon Skylake IBRS),

4.7 Conclusions

In this article we presented BTLE, a two-phases protocol for obtaining atomic swaps between two blockchains based on time-lock puzzles instead of HTLC which (differently from HTLC) can be used in both a 1-1 swap and a many-1 swap as found in e.g. online exchanges. In the document we have proved the feasibility of the protocol and we proposed an implementation of all the functions of BTLE that can

[3]See files whose name starts with `generation-values-from-test` at `https://github.com/disnocen/dex-tlp`

be found online[4]. In particular, time-lock puzzle constructions for both the RSW-TL and the BM-TL are presented and implemented. Furthermore, we tested BTLE using the two types of time-lock puzzles and we published the results. Our protocol is subject to quantum attacks. To address these challenges, future adaptations of our protocol could explore alternative time-lock constructions based on quantum-safe assumptions, such as lattice-based cryptography or hash-based time-lock puzzles.

[4] https://github.com/disnocen/dex-tlp

Chapter 5

Implementation of a Post-Quantum Anonymous Verifiable Credential Framework

The result presented in this chapter is a joint work with Davide Margaria, Pino Alessandro, Andrea Vesco, Giuseppe D'Alconzo, Antonio J. Di Scala and Carlo Sanna and it can be found in [116]. 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].

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 document 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.

Contributions. This work 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 document also presents our new open source implementation [117] with a proper experimental assessment of the framework defined by Bootle, Lyubashevsky, Nguyen, and Sorniotti (BLNS) [9, 10]. Overall, this document 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.

5.1 Introduction

In the introduction to this work we provide a comprehensive description of the Self-Sovereign Identity (SSI) model (which can be found in Section 3.2), the foundation of modern decentralized identity systems. In particular, Section 3.2 discusses the core principles of the SSI model, such as the roles of issuers, holders, and verifiers within the *Triangle of Trust*. While the SSI model offers a promising foundation for decentralized identity, its reliance on traditional cryptographic techniques presents significant challenges in the context of emerging quantum technologies. In fact, *anonymous credential schemes* based on traditional cryptography, such as CL [62] and BBS^+ [63], are feasible from an implementation standpoint, but are not suitable considering the development of Cryptographically Relevant Quantum Computers (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. For this reason, we explore the limitations of traditional cryptographic methods in the context of quantum computing and examine the ongoing research in the area of Post-Quantum ZKPs. We also discuss how these emerging techniques could be integrated with SSI to ensure the privacy and security of identity management systems in a post-quantum world.

5.2 PQ Anonymous Credential

Anonymous credential schemes based on Post-Quantum Cryptography (PQC) have only recently been proposed (not before 2022) [9, 10, 118, 119], with the exception of [120], which is impractical as the signature size is at least 670 MB [118]. Overall, the literature on PQ anonymous credentials is still in its infancy and none of the schemes address implementation issues. To the best of our knowledge, only a single implementation of the scheme in [9, 10] is publicly available, the LaZer library [121]. 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 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 documents 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 etal in [118] present a lattice-based framework for anonymous credentials, improving the previous state-of-the-art [120]. The size of a proof is around 640 KB. Lai etal in [119] build on top of [118] 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 [9, 10] 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 [122] and they use the Ajtai commitment [123]. Both [118] and [119] exploit the *signature scheme with efficient protocol* paradigm introduced in the seminal work of Camenisch and Lysyanskaya [62] to build the anonymous credential framework,

and both [119] and [9] use the sampling trapdoor algorithm described in [124]. In addition, the ISIS$_f$ assumption on which the BLNS framework [9] is based is a very natural generalization of the underlying problem upon classic lattice-based signature schemes such as [124] are based, and it is also similar to other recently, and independently, proposed lattice assumptions [125]. Under specific choices of f, the new assumption is proven to be equivalent to SIS, a well-known assumption introduced by Ajtai in [123], on which many constructions base their security, like the Dilithium signature [126] relying on the variant SelfTargetMSIS.

In light of these considerations, we have chosen the BLNS framework [9, 10] because it excels in (*a*) having the shortest proof dimension, resulting in the smallest size for a VC (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 [127], also implemented in the FALCON digital signature [128] selected by NIST for future standardization. The BLNS framework [9] 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 $\mathscr{L}$. 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 $\mathscr{L}$ such that the difference $s - v_m$ is short enough, 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 [129], 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.

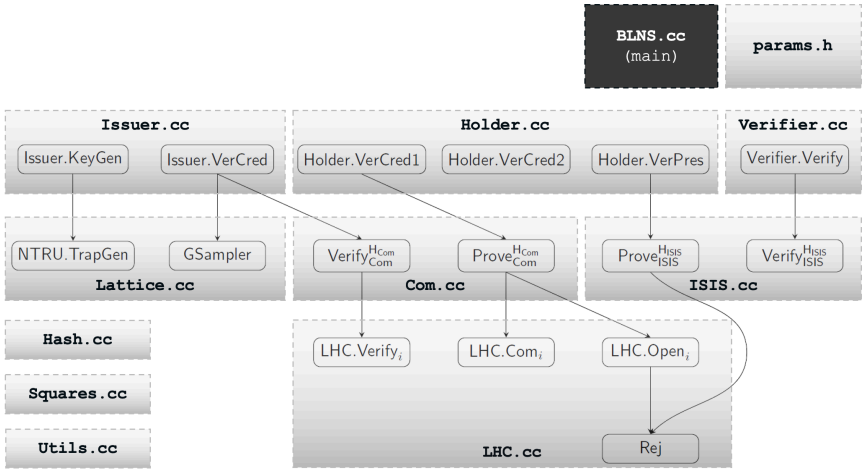

Fig. 5.1 Architecture and dependency graph of BLNS core functions implementation.

Since there are no PQ versions of the accumulator-based solution, we built on top of the timestamp approach in [84] that is quantum-secure by construction (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, [61, 130]. Furthermore, this approach enables a *rich revocation semantic*, since the credential can be partially revoked and/or updated (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.

5.3 Tailoring the BLNS Framework to PQ VCs

5.3.1 General Principles and High-level Architecture

Figure 5.1 presents the architecture and the dependency graph of BLNS core function implementation. The resulting credential system consists of three actors as in Figure 3.1: 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 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 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.

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.

5.3.2 Issuing Protocol

Figure 5.2 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.

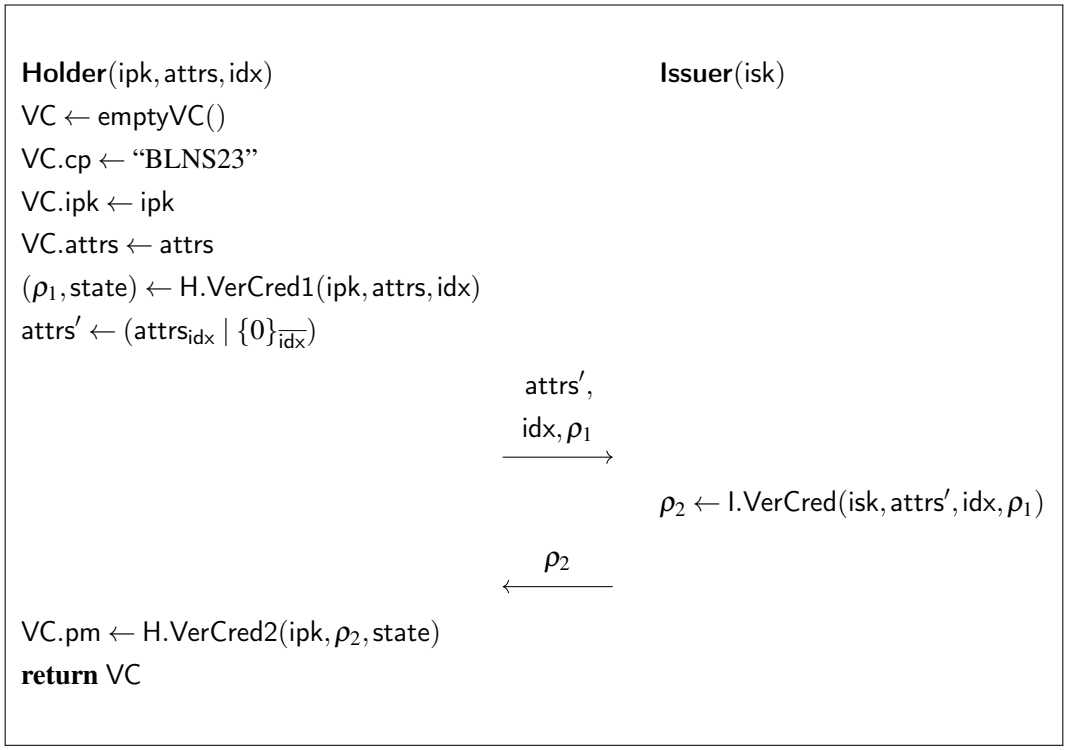

Fig. 5.2 Issuing protocol.

5.3.3 Verification Protocol

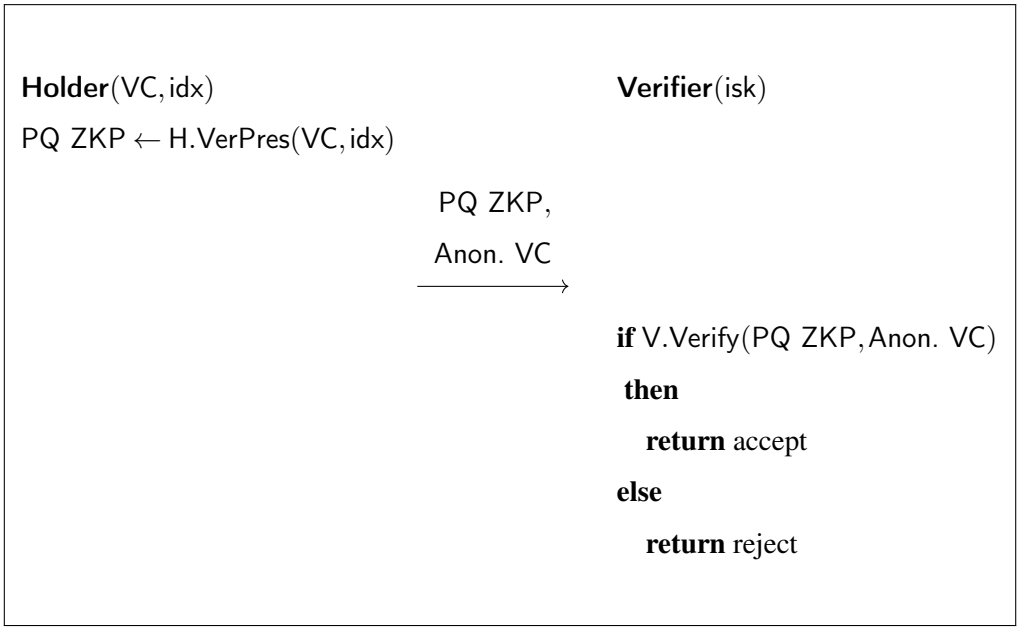

Fig. 5.3 Verification protocol.

Figure 5.3 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.

5.3.4 Security Parameters

Table 5.1 shows the parameters of the framework defined in [10]. For the security level of 128 bit, it reports both the original parameters proposed in [10] (first row) and the ones inspired by the LaZer paper [121] (second row). Note that [10] does not report any parameters sets for the higher security levels (ie 192 and 256 bit). We estimated these values starting from the FALCON parameters [128]. FALCON defines the parameters for the key generation (ie 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.

Table 5.1 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

5.4 Software Implementation

5.4.1 Key Implementation Choices and External Dependencies

We implemented the BLNS framework in C++ language, following the original pseudocode in [9, 10] 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 relying 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 5.1, we used the ones in the second row (inspired by the LaZer paper [121]) 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 [131] and FLINT [132]. Taking into account the comparative analysis reported in [133], we selected the NTL library mainly for its simpler usability and higher expected performance. Moreover, NTL [131] only requires the GMP [134] 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 [128]. 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.

5.4.2 Core Functions and Preliminary Performance

Figure 5.1 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 (ie `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 (ie $\mathsf{Prove}_{\mathsf{Com}}$, $\mathsf{Verify}_{\mathsf{Com}}$, $\mathsf{Prove}_{\mathsf{ISIS}}$, $\mathsf{Verify}_{\mathsf{ISIS}}$). 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 [9], 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 5.2 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.

Table 5.2 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	Holder.VerCred1	173.55	3064.23	629.56	467.15	475.98
Issuing	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	Holder.VerPres	317.67	2316.57	543.91	422.67	260.84
Protocol	Verifier.Verify	120.39	265.11	155.29	149.13	21.57

Concerning the initialization of the protocols, the Issuer.KeyGen is faster (approx. 5 ms) than the Holder.Init (ie 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 5.2 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 [9] 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 (ie an average of 4.6 and 2.5 repetitions, respectively) and, thus, affects the current performance of Holder.VerCred1 and Holder.VerPres.

5.5 Conclusions

This document 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 [117] 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 (eg investigate alternatives to reduce the impact of the rejection sampling) and the efficiency of the underlying algorithms under multiple configurations (eg 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.

Chapter 6

On Delegation of Verifiable Presentations - Extended Version

The work presented in this chapter is a oint work with Andrea Gangemi and Andrea Flamini and is an extended version of the document [135] presented at 3rd International Workshop on Trends in Digital Identity (TDI 2025). It includes additional technical details, extended security proofs, and new results that were not part of the original conference publication.

In this work, we address the problem of constructing *delegations of verifiable presentations* derived from both verifiable and anonymous credentials. Our goal is to enable a credential holder (the *delegator*) to securely authorize another party (the *delegatee*) to present a credential on their behalf. We introduce the notion of a *verifiable presentation delegation scheme*, formalizing the core algorithms for *delegation issuance*, *delegated presentation*, and *presentation verification*. We identify and rigorously define the essential security properties that such schemes must satisfy—namely, *correctness*, *unforgeability*, and, in the case of anonymous credentials, *unlinkability*. We then propose two concrete delegation schemes: the first is compatible with verifiable credentials supported in the *European Digital Identity Wallet Architecture Reference Framework (EUDI ARF)*, while the second supports anonymous credentials and is instantiated using *BBS signatures*. We formally prove that both constructions satisfy the required security properties. Finally, we discuss how our schemes can be instantiated in practical ecosystems, focusing in particular

on their integration within the *European Blockchain Services Infrastructure (EBSI)* and the *EUDI framework*.

Contributions. Unlike the delegation of issuance of anonymous credentials, a topic widely studied in the cryptographic literature, for instance [136–141], to the best of our knowledge, delegation of VPs has not been formalized in the existing literature. However, in order to ensure the secure deployment of the delegation of VP functionality, it is necessary to formally define both its syntax and the security properties that it must satisfy to enable a security analysis of its instantiations. For this reason, we fill this gap and provide a formal definition of VP delegation scheme as an extension that is built on top of a verifiable credential scheme. We define the syntax of a VP delegation scheme and we identify the security properties that it must satisfy, namely the *correctness*, that guarantees that legitimate delegators can delegate third parties to perform a presentation on their behalf; the *unforgeability of delegated presentations*, that captures that an adversary should not be able to present a delegation for which it is not a legitimate delegatee, and should not be able to generate a delegation for a delegator payload that is not certified by the credentials it controls; finally the *unlinkability* that captures the ability to generate delegations, and delegated presentations that can not be linked to a specific credential issued by the issuer. Then, we describe a VP delegation scheme that can be applied to mdoc VCs and another one that can be applied to ACs: for the sake of concreteness, the VP delegation scheme is instantiated with BBS anonymous credentials [142, 74, 15], but it can be easily generalized to any AC. We prove the security of our VP delegation schemes according to our security notion. We prove the unforgeability of both schemes (in particular, we observe that the proof of the scheme based on anonymous credentials is remarkably simpler than the one based on mdoc VCs) and we prove the unlinkability of the scheme based on AC. To conclude, we briefly discuss the compatibility of our scheme in the context of the EUDI and EBSI frameworks. For simplicity, we assume each party is provided with a single VC/AC.

6.1 Introduction

In this work, we focus on one of the features that can be enabled by the use of verifiable credentials: the ability to delegate the presentation of verifiable credentials

to a different holder without involving any trusted third party. The relevance of this feature is attested by its inclusion in the EUDI Wallet Reference Implementation Roadmap [143].

Delegation of VPs A credential holder (the delegator), using its VC or AC, can create a delegation that instructs another holder (the delegatee) to act on its behalf to execute a *specific and predetermined* operation while interacting with a verifier. VP delegation is developed as a mechanism allowing one to cover use cases such as: someone authorizes a family member to pick up a prescription from the pharmacy, an elderly person authorizes their adult child to represent them in a public online service, or a person delegates an intermediary to perform financial operations on its behalf. For simplicity, in this work, we consider a system in which every user is provided a "main credential", like an ID card, by a trusted issuer that every holder can use to delegate to other holders. Our definition of a VP delegation scheme builds on top of such a VC or AC scheme. In particular, we describe a delegation mechanism that allows a delegator to generate delegations that must specify:

- the *delegatee identifier*, a statement that the identity (or the VC/AC) of the delegatee must satisfy. For example, the delegatee identifier could be "the delegatee is over 18", then any over 18 holder would be allowed to present such delegation, or more specific statements such as "the delegatee name is *Name* and their surname is *Surname*", in such a case only a holder named *Name Surname* would be allowed to present such delegation. This identifier is sent by the delegatee to the delegator before the latter produces the delegation;
- the *scope* for which the delegation is being created. This might include a period of validity, an identifier of the verifier to which it must be presented, and an identifier of the operation that the delegatee must perform;
- the *delegator payload*, a statement about the delegator's identity that contains (at least) the information the verifier would request from any holder to perform the operation specified in the scope;
- a *proof* that the delegator identity (or VC/AC) satisfies the delegator payload.

The delegatee, holding the delegation, interacts with the verifier to present the delegation and executes the operations specified in the scope. The delegatee proves

to be an authorized delegatee by showing, using its own credential, that it satisfies the delegatee identifier information specified in the delegation. Finally, the verifier checks that the delegation and the presentation of the delegation are valid and accepts (or rejects) the delegated presentation, allowing the delegatee to perform (or preventing the delegatee from performing) the operation specified in the scope.

In this work, we focus on the class of VCs schemes that are currently adopted in the EUDI Wallet Architecture Reference Framework (ARF) [53, Annex 2, Topic 12], or are considered for possible future adoption. According to the EUDI ARF, the supported VC schemes must have one of the following formats: mdoc, described in [57, 56], SD-JWT, described in [144], or W3C VCDM, described in [145]. These different verifiable credential formats are based on the same cryptographic mechanism that we abstract in this section, and, for simplicity, we will refer to it as *mdoc VC* (desribed in Section 3.4). The VC scheme considered for possible future adoption by the EUDI ARF is the BBS anonymous credential scheme [15, 142] (described in Section 3.7), as also advised in [55], which has recently attracted significant attention in the cryptographic literature [74, 146–153]. Anonymous credentials [154, 58] are the tool recommended for addressing the linkability problem holding for mdoc VCs. In the case of BBS anonymous credentials[1], the unlinkability property is achieved through a NIZKP that allow the prover to demonstrate possession of a valid credential while revealing no information beyond the disclosed statement.

6.2 VP Delegation Scheme: Definition and Security Notions

In this section, we define the notion of a *VP delegation scheme*, which is built on top of a VC scheme, as defined in Definition 3.3.1. A VP delegation scheme must define the following algorithms: (1) a delegation issuance algorithm, to allow the delegator D to create a delegation del, (2) a delegation verification algorithm to check the validity of a delegation, (3) a delegation presentation algorithm, to allow the delegatee Δ to present del to a verifier, generating a presentation pres, and (4) to verify

[1] As for all the schemes that follow the framework of Camenisch and Lysyanskaya [58] like [155, 156, 142, 74].

the delegated presentation pres. We formally define the input-output specifications of each algorithm that constitutes a VP delegation scheme.

Definition 6.2.1 (VP delegation scheme) *A* VP delegation scheme $\mathscr{VPDS}$*, built on top of a VC or AC scheme (*$\mathsf{IssuerSetup}, \mathsf{CredIssuance}, \mathsf{CredPresentation}, \mathsf{PresVer}$*), is defined by the the following algorithms*[2]*:*

- Delegation issuance*:* $\underbrace{(\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}})}_{\mathsf{del}} \xleftarrow{\$} \mathsf{DelegIssuance}(\mathsf{cred}_{\mathsf{D}}, \mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}})$*,*

  where:

  - Δ_{ID} *is the* delegatee identity *which is a statement that must be satisfied by the delegatee presenting* del *(and its credential).* Δ_{ID} *represents a set of attributes with the associated indices that the delegatee must reveal when presenting the delegation. With a slight abuse of notation we can say that, being* $\{a_i\}_{i\in[l]}$ *the attributes in the credential,* $\Delta_{\mathsf{ID}} = \{a_i\}_{i\in\Delta_{\mathsf{ID}}}$*;*
  - scope *is the* delegation scope*, describing the operation that can be performed by the delegatee interacting with specified verifiers* [3]*;*
  - DP *is the* delegator payload *containing the attributes that the delegator must disclose about its identity (e.g. it is entitled to withdraw a specific drug) when it creates its delegation. As for* Δ_{ID}*, we write* $\mathsf{DP} = \{a_i\}_{i\in\mathsf{DP}}$*;*
  - π_{DP} *is a proof that the holder knows a credential* $\mathsf{cred}_{\mathsf{D}}$ *for the claims in* DP*. The proof must be bound to* scope *and* Δ_{ID}*.*

  See Section 6.1 for a better intuition about the semantic meaning of these fields.

- Delegation verification*:* $\{0,1\} \xleftarrow{\$} \mathsf{DelegVer}(\mathsf{del}, \mathsf{pk}_{\mathsf{Iss}})$*.*

  This algorithm is executed by a delegatee (or by a verifier) to verify the validity of the delegation.

- Delegated presentation*:* $\underbrace{(\mathsf{del}, \pi_{\mathsf{del}})}_{\mathsf{pres}} \xleftarrow{\$} \mathsf{DelegPres}(\mathsf{del}, \mathsf{cred}_{\Delta}, \mathsf{nonce})$*.*

[2] In the rest of this work, the public parameters pp are omitted from the input of most described algorithms for notational simplicity.

[3] Note that these information must be encoded using a dictionary or scheme which subsequently allows verifiers of delegated presentations to verify it. A more detailed analysis of this aspect is out of scope and will be treated in a future work.

The verifier sends to the delegatee a random nonce *which is used to guarantee the freshness of the delegated presentation. After parsing* del *as* $(\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}})$*, the delegatee computes* π_{del} *which is a proof of knowledge of a credential* cred_{Δ} *satisfying the delegatee identity* Δ_{ID} *included in* del*.*

- Delegated presentation verification*:* $\{0,1\} \xleftarrow{\$} \mathsf{DelegPresVer}(\mathsf{pres}, \mathsf{pk}_{\mathsf{Iss}})$.

  This is the verification algorithm executed by the verifier. Parse pres *as* $(\mathsf{del}, \pi_{\mathsf{del}})$*. Upon receiving the proof* π_{del} *and the delegation* del*, the verifier checks that*

  - $1 \leftarrow \mathsf{DelegVer}(\mathsf{del}, \mathsf{pk}_{\mathsf{Iss}})$*;*
  - π_{del} *is a valid proof for the value* Δ_{ID} *in* del*;*
  - scope *included in* del*, which is part of* pres*, is satisfied.*

  If these checks pass, the delegated presentation is accepted, and the algorithm outputs 1.

Figure 6.1 describes the interaction flow between the delegator D, the delegatee Δ and the Verifier V.

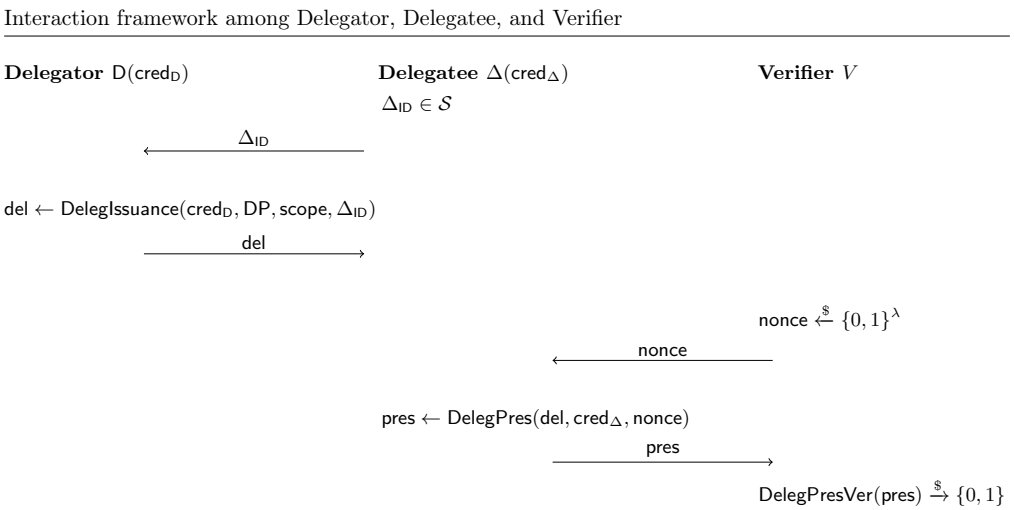

Fig. 6.1 Interactions between the delegator D, the delegatee Δ, and the Verifier V. According to the scenario, the delegatee might send to the delegator some information Δ_{ID} about its identity, so that the delegator can use it as an input for the algorithm DelegIssuance.

We now define the security properties for VP delegation schemes.

Correctness Property. The correctness property specifies that a delegation algorithm always allows an honest delegator (who generates a delegation for a DP consistent with its identity), to generate a delegation del and delegate another user. Then, if the user is an honest delegatee (its identity satisfies Δ_{ID}), it can always present del satisfying scope to a verifier.

Definition 6.2.2 (Correctness of VP delegation scheme) *Given a VP delegation scheme*

$$\mathscr{VPDS} = (\mathsf{DelegIssuance}, \mathsf{DelegVer}, \mathsf{DelegPres}, \mathsf{DelegPresVer}),$$

we say that the scheme is correct if $1 \leftarrow \mathsf{DelegPresVer}(\mathsf{pres})$ *whenever:*

- $\mathsf{del} \xleftarrow{\$} \mathsf{DelegIssuance}(\mathsf{cred}_{\mathsf{D}}, \mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}})$ *where* $\mathsf{cred}_{\mathsf{D}}$ *satisfies the statements contained in* DP *(which is contained in* del*);*
- $\mathsf{pres} \xleftarrow{\$} \mathsf{DelegPres}(\mathsf{del}, \mathsf{cred}_{\Delta}, \mathsf{nonce})$*, where* cred_{Δ} *satisfies the statements contained in* Δ_{ID}.

Unforgeability Property. We consider two notions of unforgeability: the unforgeability of the delegation algorithm DelegIssuance, which means that an adversary cannot forge a delegation from a credential it does not possess, and the unforgeability of the delegation presentation algorithm DelegPres, which means that an adversary cannot present a delegation that is not issued to its identity (i.e., its identity does not satisfy Δ_{ID}). We capture these two notions of unforgeability in a single experiment.

The experiment captures that an adversary $\mathscr{A}$, after receiving the public key of the issuer $\mathsf{pk}_{\mathsf{Iss}}$, the public parameters pp, and performing a training, cannot forge a delegation presentation. The training considers an adversary that can learn information from the system in the following ways: it can (1) corrupt a polynomial number of users, (2) receive a polynomial number of delegations from users it does not control, and (3) verify a polynomial number of delegated presentations (i.e., receive a polynomial number of presentations of its choice). These operations are modeled by allowing $\mathscr{A}$ to query a polynomial number of credentials to the oracle $\mathsf{O}_{\mathsf{iss}}$, of delegations to the oracle $\mathsf{O}_{\mathsf{del}}$ and of presentations of delegations to the oracle $\mathsf{O}_{\mathsf{pres}}$.

Note that in a system supporting the delegation of VPs, the adversary can see not only delegated presentations, but also presentations of verifiable credentials (generated using the algorithm $\mathsf{CredPresentation}(\mathsf{cred}, \mathsf{stmt})$). However, we omit the queries for credentials presentations because we assume that credentials presentations can be seen as delegated presentations associated to an empty delegation $(\mathsf{del}, \mathsf{nonce}) = (\perp, \perp)$.

Experiment 4 ($\mathsf{Exp}_{\mathscr{A}}^{\mathsf{DelPresUnforgeability}}(1^{\lambda})$) *The experiment is given by the following phases.*

Setup phase *$\mathscr{A}$ receives from the challenger of the experiment the public key* $\mathsf{pk}_{\mathsf{Iss}}$ *of the issuer and the public parameters* pp *of the underlying VC or AC scheme.*

Training phase *$\mathscr{A}$ can interact with random oracles* $\mathsf{O}_{\mathsf{iss}}, \mathsf{O}_{\mathsf{del}}$ *and* $\mathsf{O}_{\mathsf{pres}}$*, to which it can send a polynomial number of issuing queries. Each query has the following input-output specification:*

1. *Issuing queries to* $\mathsf{O}_{\mathsf{iss}}$*: $\mathscr{A}$ can query for the issuance of a credential for a set of attributes* $\{a_i\}_{i\in[l]}$ *of its choice; the oracle computes a credential* $\mathsf{cred} \leftarrow \mathsf{CredIssuance}(\{a_i\}_{i\in[l]}, \mathsf{sk}_{\mathsf{Iss}})$*, stores* cred *to the* credential table CT *and sends it to $\mathscr{A}$.*
2. *Delegation queries to* $\mathsf{O}_{\mathsf{del}}$*: $\mathscr{A}$ can query* $\mathsf{O}_{\mathsf{del}}$ *for the issuance of a delegation for* $(\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP})$ *of its choice; the oracle computes a delegation* $\mathsf{del} \leftarrow (\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}})$*, stores* del *in the* delegation table DT *and sends it to $\mathscr{A}$.*
3. *Delegated presentation queries to* $\mathsf{O}_{\mathsf{pres}}$*: $\mathscr{A}$ can query for the presentation of a delegation for the values* $(\mathsf{del}, \mathsf{nonce})$ *of its choice; the oracle* $\mathsf{O}_{\mathsf{pres}}$ *generates a presentation* pres *for* $(\mathsf{del}, \mathsf{nonce})$*, stores* pres *to the* presentation table PT *and sends it to $\mathscr{A}$.*

Forgery phase *$\mathscr{A}$ eventually outputs a delegated presentation* $\mathsf{pres}^{\star} = (\mathsf{del}^{\star}, \pi_{\mathsf{del}^{\star}})$.

Winning conditions *Parse* $\mathsf{pres}^{\star} = (\mathsf{del}^{\star}, \pi_{\mathsf{del}^{\star}})$ *as* $(\Delta_{\mathsf{ID}}{}^{\star}, \mathsf{scope}^{\star}, \mathsf{DP}^{\star}, \pi_{\mathsf{DP}^{\star}}, \pi_{\mathsf{del}^{\star}})$.

The adversary wins the experiment if $1 \leftarrow \mathsf{DelegPresVer}(\mathsf{pres}^{\star})$ *and at least one of the following conditions is satisfied:*

1. forgery of the delegation*:* $\mathsf{del}^{\star} = (\Delta_{\mathsf{ID}}{}^{\star}, \mathsf{scope}^{\star}, \mathsf{DP}^{\star}, \pi_{\mathsf{DP}^{\star}})$ *is forged, i.e.*

- *it is not a delegation queried by* $\mathscr{A}$ *to* $\mathsf{O}_{\mathsf{del}}$*, i.e.* $\mathsf{del}^\star \notin \mathsf{DT}$[4]*;*
- $\mathsf{del}^\star$ *is not generated using any credential issued to* $\mathscr{A}$*.*

2. forgery of the delegated presentation*:* $\pi_{\mathsf{del}^\star}$ *is forged, i.e.*
   - $\forall \mathsf{pres} \in \mathsf{PT}$, $\mathsf{pres} = (\mathsf{del}, \pi_{\mathsf{del}})$, $\pi_{\mathsf{del}} \neq \pi_{\mathsf{del}^\star}$*;*
   - $\pi_{\mathsf{del}^\star}$ *is not generated using any credential issued to* $\mathscr{A}$*.*

Definition 6.2.3 (Unforgeability of VP delegation scheme) *We say that a VP delegation scheme is unforgeable if for any PPT adversary* $\mathscr{A}$ *executing* $\mathsf{Exp}_{\mathscr{A}}^{\mathsf{DelPresUnforgeability}}(1^\lambda)$*,*

$$\Pr\left[\mathscr{A} \textit{ wins } \mathsf{Exp}_{\mathscr{A}}^{\mathsf{DelPresUnforgeability}}(1^\lambda)\right] \leq \nu(\lambda),$$

where $\nu(\lambda)$ *is negligible in the security parameter* λ*.*

Unlinkability Property. We consider two notions of *unlinkability*. The first concerns the DelegIssuance algorithm, ensuring that, given a delegation generated from one of two possible delegator credentials, an adversary cannot distinguish which credential was used. The second relates to the DelegPres algorithm, which guarantees that, given a delegated presentation generated from one of two possible delegatee credentials, an adversary cannot determine which credential produced it. We formalize these two notions of unlinkability through two corresponding experiments.

Experiment 5 ($\mathsf{Exp}_{\mathscr{A}}^{\mathsf{DelegIssuanceUnlinkability}}(1^\lambda)$**)**

1. *The adversary* $\mathscr{A}$ *generates the public parameters* pp *for the verifiable credential scheme together with the issuer key pair* $(\mathsf{sk}_{\mathsf{Iss}}, \mathsf{pk}_{\mathsf{Iss}})$ *and send* pp *and* $\mathsf{pk}_{\mathsf{Iss}}$ *to the challenger.*
2. *The challenger samples a random bit* $b \xleftarrow{\$} \{0,1\}$*.*
3. $\mathscr{A}$ *generates and sends to the challenger* Δ_{ID}*,* DP*,* scope *and* cred_{D_0} *and* cred_{D_1}*, satisfying* DP*.*
4. *The challenger runs* $\mathsf{del}_b \xleftarrow{\$} \mathsf{DelegIssuance}(\mathsf{cred}_{D_b}, \mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}})$*.*

[4]In the constructions we will present, this implies that $\pi_{\mathsf{DP}^\star}$ is different from any π_{DP} generated by $\mathsf{O}_{\mathsf{del}}$.

5. *The challenger sends* del_b *to* $\mathscr{A}$.
6. $\mathscr{A}$ *outputs a bit* b'.
7. $\mathscr{A}$ *wins if* $b' = b$.

If the adversary has access to a random oracle (according to the VP delegation scheme definition), it can send random oracle queries before sending to the challenger $\mathsf{del}, \mathsf{nonce}, \mathsf{cred}_{\mathsf{D}_0}$ *and* $\mathsf{cred}_{\mathsf{D}_1}$.

Definition 6.2.4 (Unlinkability of the delegation) *A VP delegation scheme satisfies* unlinkability of the delegation *if for any PPT adversary* $\mathscr{A}$ *executing* $\mathsf{Exp}_{\mathscr{A}}^{\mathsf{DelegIssuanceUnlinkability}}(1^\lambda)$,

$$\left|\Pr\left[\mathscr{A} \text{ wins } \mathsf{Exp}_{\mathscr{A}}^{\mathsf{DelegIssuanceUnlinkability}}(1^\lambda)\right] - \frac{1}{2}\right| = \left|\Pr[b = b'] - \frac{1}{2}\right| \leq \nu(\lambda),$$

where $\nu(\lambda)$ *is negligible in the security parameter* λ.

Experiment 6 ($\mathsf{Exp}_{\mathscr{A}}^{\mathsf{DelegPresUnlinkability}}(1^\lambda)$)

1. *The adversary* $\mathscr{A}$ *generates the public parameters* pp *for the verifiable credential scheme together with the issuer key pair* $(\mathsf{sk}_{\mathsf{Iss}}, \mathsf{pk}_{\mathsf{Iss}})$ *and send* pp *and* $\mathsf{pk}_{\mathsf{Iss}}$ *to the challenger.*
2. *The challenger samples a random bit* $b \xleftarrow{\$} \{0,1\}$.
3. *The adversary* $\mathscr{A}$ *generates and sends to the challenger* del, nonce *and* cred_{Δ_0} *and* cred_{Δ_1}, *satisfying* Δ_{ID}.
4. *The challenger runs* $\mathsf{pres}_b \xleftarrow{\$} \mathsf{DelegPres}(\mathsf{del}, \mathsf{cred}_{\Delta_b}, \mathsf{nonce})$.
5. *The challenger sends* pres_b *to* $\mathscr{A}$.
6. $\mathscr{A}$ *outputs a bit* $\bar{b}$.
7. $\mathscr{A}$ *wins if* $\bar{b} = b$.

If the adversary has access to a random oracle (according to the VP delegation scheme definition), it can send random oracle queries before sending to the challenger $\mathsf{del}, \mathsf{nonce}, \mathsf{cred}_{\Delta_0}$ *and* cred_{Δ_1}.

Definition 6.2.5 (Unlinkability of the delegated presentation) *A VP delegation scheme satisfies* unlinkability of the delegated presentation *if for any PPT adversary $\mathscr{A}$ executing* $\mathsf{Exp}^{\mathsf{DelegPresUnlinkability}}_{\mathscr{A}}(1^\lambda)$,

$$\left|\Pr\left[\mathscr{A} \textit{ wins } \mathsf{Exp}^{\mathsf{DelegPresUnlinkability}}_{\mathscr{A}}(1^\lambda)\right] - \frac{1}{2}\right| = \left|\Pr[\bar{b} = b] - \frac{1}{2}\right| \leq \nu(\lambda),$$

where $\nu(\lambda)$ is negligible in the security parameter λ.

6.3 VP Delegation Schemes from mdoc VCs and BBS ACs

In this section, we first create an instance of a VP delegation scheme according to Definition 6.2.1 that is built on top of the mdoc VC scheme, as described in Definition 3.4.1. Then, we repeat the same idea to create an instance of a VP delegation scheme built on top of the BBS AC scheme, as described in Definition 3.7.1.

In the former case, the delegator and the delegatee each have a VC issued by the issuer of the following form:

$$\mathsf{cred}_{\mathsf{D}} = \left((\sigma_{\mathsf{D}}, \{\mathsf{com}_{\mathsf{D},i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}}), \{a_{\mathsf{D},i}\}_{i\in[l]}, \{\mathsf{salt}_{\mathsf{D},i}\}_{i\in[l]}, \mathsf{sk}_{\mathsf{cred}_{\mathsf{D}}}\right),$$

$$\mathsf{cred}_{\Delta} = \left((\sigma_{\Delta}, \{\mathsf{com}_{\Delta,i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\Delta}}), \{a_{\Delta,i}\}_{i\in[l]}, \{\mathsf{salt}_{\Delta,i}\}_{i\in[l]}, \mathsf{sk}_{\mathsf{cred}_{\Delta}}\right).$$

Note that in this case all the proofs introduced in Definition 6.2.1 are digital signatures.

In the latter case, instead, the delegator and the delegatee each have a BBS credential issued by the issuer of the following form:

$$\mathsf{cred}_{\mathsf{D}} = \left(\sigma_{\mathsf{D}}, \{a_{\mathsf{D},i}\}_{i\in[l]}\right),$$

$$\mathsf{cred}_{\Delta} = \left(\sigma_{\Delta}, \{a_{\Delta,i}\}_{i\in[l]}\right).$$

where $\sigma_{\mathsf{D}} = (A_{\mathsf{D}}, e_{\mathsf{D}})$ and $\sigma_{\Delta} = (A_{\Delta}, e_{\Delta})$. In this case, all the proofs are non-interactive zero-knowledge proofs.

Definition 6.3.1 (mdoc VP Delegation Scheme) *Given the mdoc VC scheme defined in Definition 3.4.1, which uses the digital signature scheme $\mathscr{DS}$ and a cryptographic hash function H, we can define the following delegation scheme (see Definition 6.2.1).*

- Delegation issuance*:* $\underbrace{(\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}})}_{\mathsf{del}} \xleftarrow{\$} \mathsf{DelegIssuance}(\mathsf{cred}_{\mathsf{D}}, \mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}})$.

  On input $(\mathsf{cred}_{\mathsf{D}}, \mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}})$*, the delegator computes* π_{DP} *as a variant of a presentation of* $\mathsf{cred}_{\mathsf{D}}$ *which is bound to* DP *but also to* scope *and* Δ_{ID}*:*

  - $\mathsf{pres}' \leftarrow ((\sigma_{\mathsf{D}}, \{\mathsf{com}_{\mathsf{D},i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}}), \{\mathsf{salt}_{\mathsf{D},i}\}_{i\in\mathsf{DP}})$*;*
  - $\sigma' \xleftarrow{\$} \mathsf{Sign}((\mathsf{pres}'||\mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}}), \mathsf{sk}_{\mathsf{cred}_{\mathsf{D}}})$ *and* $\pi_{\mathsf{DP}} \leftarrow (\mathsf{pres}', \sigma')$*;*
  - *return* $\mathsf{del} \leftarrow ((\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}}))$.[5]

- Delegation verification*:* $\{0,1\} \xleftarrow{\$} \mathsf{DelegVer}(\mathsf{del}, \mathsf{pk}_{\mathsf{Iss}})$.

  To verify the delegation, parse:

$$\mathsf{del} \rightarrow (\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}}), \quad \pi_{\mathsf{DP}} \rightarrow (\mathsf{pres}', \sigma'),$$

$$\mathsf{pres}' \rightarrow ((\sigma_{\mathsf{D}}, \{\mathsf{com}_{\mathsf{D},i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}}), \{\mathsf{salt}_{\mathsf{D},i}\}_{i\in\mathsf{DP}}).$$

  Then, perform the following checks:

  - *Verify the signature of the issuer:* $1 \leftarrow \mathsf{Vf}(\sigma_{\mathsf{D}}, (\{\mathsf{com}_{\mathsf{D},i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}}), \mathsf{pk}_{\mathsf{Iss}})$. *This means that the delegation is created from a valid VC issued by* $\mathsf{pk}_{\mathsf{Iss}}$*;*
  - *Check that* $\mathsf{com}_{\mathsf{D},i} = H(a_{\mathsf{D},i}||\mathsf{salt}_{\mathsf{D},i}), \forall i \in \mathsf{DP}$. *This means that the delegation has a* DP *consistent with the credential used to generate it;*
  - *verify the signature* σ' *of* pres' *using the public key* $\mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}}$*:*

$$1 \leftarrow \mathsf{Vf}(\sigma', (\mathsf{pres}'||\mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}}), \mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}}).$$

[5] Note that π_{DP} is structured as a presentation of the mdoc VC which is bound, via σ', to the specific $\mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}}$ that will be part of the delegation del.

This means that the delegation (for scope *and* Δ_{ID}*) was created by the holder of the associated credential.*

- Delegated presentation*:* $\underbrace{(\mathsf{del}, \pi_{\mathsf{del}})}_{\mathsf{pres}} \xleftarrow{\$} \mathsf{DelegPres}(\mathsf{del}, \mathsf{cred}_{\Delta}, \mathsf{nonce})$.

  After parsing $\mathsf{del} \rightarrow (\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}})$*, the delegatee computes* π_{del} *as a presentation of* Δ_{ID} *using* cred_{Δ} *which is bound to* del*:*

  – *compute* $\mathsf{pres}'' \leftarrow ((\sigma_{\Delta}, \{\mathsf{com}_{\Delta,i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\Delta}}), \{\mathsf{salt}_{\Delta,i}\}_{i\in\Delta_{\mathsf{ID}}}, \mathsf{nonce})$*;*

  – *the delegatee signs* pres'' *computing* $\sigma'' \xleftarrow{\$} \mathsf{Sign}((\mathsf{pres}''||\mathsf{del}), \mathsf{sk}_{\mathsf{cred}_{\Delta}})$*, sets* $\pi_{\mathsf{del}} \leftarrow (\mathsf{pres}'', \sigma'')$ *and returns* $\mathsf{pres} \leftarrow (\mathsf{del}, \pi_{\mathsf{del}})$.

- Delegated presentation verification*:* $\{0,1\} \xleftarrow{\$} \mathsf{DelegPresVer}(\mathsf{pres}, \mathsf{pk}_{\mathsf{Iss}})$.

  Parse $\mathsf{pres} \rightarrow (\mathsf{del}, \pi_{\mathsf{del}})$*, where* $\pi_{\mathsf{del}} \rightarrow (\mathsf{pres}'', \sigma'')$ *and* $\mathsf{pres}'' \rightarrow ((\sigma_{\Delta}, \{\mathsf{com}_{\Delta,i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\Delta}}), \{\mathsf{salt}_{\Delta,i}\}$
  The verifier checks that

  – *the delegation is valid, i.e.* $1 \leftarrow \mathsf{DelegVer}(\mathsf{del})$*;*

  – π_{del} *is a valid presentation of* cred_{Δ} *for the attributes in* Δ_{ID} *specified in* del*, i.e.:*

    1. *the signature of the issuer is valid:* $1 \leftarrow \mathsf{Vf}(\sigma_{\Delta}, (\{\mathsf{com}_{\Delta,i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\Delta}}), \mathsf{pk}_{\mathsf{Iss}})$.
    2. $\mathsf{com}_{\Delta,i} = H(a_{\Delta,i}||\mathsf{salt}_{\Delta,i})\ \forall i \in \Delta_{\mathsf{ID}}$*;*
    3. *the signature* σ'' *of* pres'' *is valid using* $\mathsf{pk}_{\mathsf{cred}_{\Delta}}$*:* $1 \leftarrow \mathsf{Vf}(\sigma'', (\mathsf{pres}''||\mathsf{del}), \mathsf{pk}_{\mathsf{cred}_{\Delta}})$*;*

  – *the value* scope *included in* del *is satisfied according to the associated verification procedure.*

  If these checks pass, the delegated presentation is accepted, and the algorithm outputs 1.

To clearly define the instantiation of our BBS-based delegation scheme, we first formalize the notation used to represent the selective disclosure of attributes. In our setting, we consider two distinct selective disclosures: one performed by the delegator and one by the delegatee. Specifically, we define:

- $\mathsf{Rev}_{\mathsf{D}}$ as the set of indices of the attributes disclosed by the delegator, identified by DP.

- $\mathsf{Hid}_\mathsf{D} = [\ell] \setminus \mathsf{Rev}_\mathsf{D}$ as the set of indices corresponding to the delegator's hidden attributes.
- Rev_Δ as the set of indices of the attributes disclosed by the delegatee, identified by Δ_ID.
- $\mathsf{Hid}_\Delta = [\ell] \setminus \mathsf{Rev}_\Delta$ as the set of indices corresponding to the delegatee's hidden attributes.

Definition 6.3.2 (BBS VP Delegation Scheme) *Given the BBS AC scheme defined in Definition 3.7.1 and a cryptographic hash function H, we can define the following delegation scheme (see Definition 6.2.1).*

- Delegation issuance: $\underbrace{(\Delta_\mathsf{ID}, \mathsf{scope}, \mathsf{DP}, \pi_\mathsf{DP})}_{\mathsf{del}} \xleftarrow{\$} \mathsf{DelegIssuance}(\mathsf{cred}_\mathsf{D}, \mathsf{DP}, \mathsf{scope}, \Delta_\mathsf{ID})$.

  On input $(\mathsf{cred}_\mathsf{D}, \mathsf{DP}, \mathsf{scope}, \Delta_\mathsf{ID})$ *the delegator* D *computes a non-interactive zero-knowledge proof* π_DP*, which is bound to* DP, scope *and* Δ_ID*, to prove that* DP *is included in* $\mathsf{anoncred}_D$*:*

  1. *generates the commitment of the underlying sigma protocol* $(\bar{A}_\mathsf{D}, \bar{B}_\mathsf{D}, U_\mathsf{D})$*:*
     - $r_1 \leftarrow \mathbb{Z}_p^*$;
     - $\bar{A}_\mathsf{D} \leftarrow A_\mathsf{D}^{r_1}, \quad \bar{B}_\mathsf{D} \leftarrow C(\mathbf{a}_\mathsf{D})^{r_1} \bar{A}_\mathsf{D}^{-e_\mathsf{D}}$;
     - $\alpha_1, \beta_1 \leftarrow \mathbb{Z}_p, \quad \delta_{1,i} \leftarrow \mathbb{Z}_p \quad \forall i \in \mathsf{Hid}_\mathsf{D}$;
     - $U_\mathsf{D} \leftarrow \bar{A}_\mathsf{D}^{\alpha_1} \bar{B}_\mathsf{D}^{\beta_1} \prod_{i \in \mathsf{Hid}_\mathsf{D}} \mathbf{h}[i]^{\delta_{1,i}}$.
  2. *computes the challenge* $c_1 \leftarrow H(\bar{A}_\mathsf{D}, \bar{B}_\mathsf{D}, U_\mathsf{D}, \mathsf{DP}, \mathsf{scope}, \Delta_\mathsf{ID})$;
  3. *computes a response* z_1*:*
     - $s_1 \leftarrow \alpha_1 + r_1^{-1} \cdot e_\mathsf{D} \cdot c_1$;
     - $n_1 \leftarrow \beta_1 + r_1^{-1} \cdot c_1$;
     - $u_{1,i} \leftarrow \delta_{1,i} - \mathbf{a}_\mathsf{D}[i]\, c_1 \quad \forall i \in \mathsf{Hid}_\mathsf{D}$;
     - $z_1 \leftarrow (s_1, n_1, (u_{1,i})_{i \in \mathsf{Hid}_\mathsf{D}})$.
  4. *sets* $\pi_\mathsf{DP} \leftarrow (\bar{A}_\mathsf{D}, \bar{B}_\mathsf{D}, c_1, z_1)$;[6]
  5. *returns* $\mathsf{del} \leftarrow (\Delta_\mathsf{ID}, \mathsf{scope}, \mathsf{DP}, \pi_\mathsf{DP})$.

[6] Note that from $\bar{A}_\mathsf{D}, \bar{B}_\mathsf{D}, c_1$ and z_1 the verifier can reconstruct U_D.

- Delegation verification*:* $\{0,1\} \xleftarrow{\$} \mathsf{DelegVer}(\mathsf{del}, \mathsf{pk}_{\mathsf{Iss}})$.

  To verify the delegation, parse:

$$\mathsf{del} \to (\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}}), \quad \pi_{\mathsf{DP}} \to (\bar{A}_{\mathsf{D}}, \bar{B}_{\mathsf{D}}, c_1, z_1).$$

  Then, the delegatee:

  1. *computes* $\tilde{U}_{\mathsf{D}}$

$$\tilde{U}_{\mathsf{D}} \leftarrow \bar{A}_{\mathsf{D}}^{s_1} \bar{B}_{\mathsf{D}}^{n_1} \prod_{i \in \mathsf{Hid}_{\mathsf{D}}} \mathbf{h}[i]^{u_{1,i}} C_{\mathsf{Rev}_{\mathsf{D}}}(\mathbf{a}_{\mathsf{D}})^{-c_1};$$

  2. *checks if*

$$c_1 \stackrel{?}{=} H(\bar{A}_{\mathsf{D}}, \bar{B}_{\mathsf{D}}, \tilde{U}_D, \mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}}),$$
$$\mathbf{e}(\bar{A}_{\mathsf{D}}, X_2) \stackrel{?}{=} \mathbf{e}(\bar{B}_{\mathsf{D}}, g_2).$$

- Delegated presentation*:* $\underbrace{(\mathsf{del}, \pi_{\mathsf{del}})}_{\mathsf{pres}} \xleftarrow{\$} \mathsf{DelegPres}(\mathsf{del}, \mathsf{cred}_{\Delta}, \mathsf{nonce})$.

  After receiving the nonce*, the delegatee computes a non-interactive zero-knowledge proof* π_{del}*, which is bound to* del*, to prove that* Δ_{ID} *is included in* cred_{Δ}*, in the following way:*

  1. *generates the commitment of the underlying sigma protocol* $(\bar{A}_{\Delta}, \bar{B}_{\Delta}, U_{\Delta})$*:*
     - $r_2 \leftarrow \mathbb{Z}_p^*$*;*
     - $\bar{A}_{\Delta} \leftarrow A_{\Delta}^{r_2}, \quad \bar{B}_{\Delta} \leftarrow C(\mathbf{a}_{\Delta})^{r_2} \bar{A}_{\Delta}^{-e_{\Delta}}$*;*
     - $\alpha_2, \beta_2 \leftarrow \mathbb{Z}_p, \quad \delta_{2,i} \leftarrow \mathbb{Z}_p \quad \forall\, i \in \mathsf{Hid}_{\Delta}$*;*
     - $U_{\Delta} \leftarrow \bar{A}_{\Delta}^{\alpha_2} \bar{B}_{\Delta}^{\beta_2} \prod_{i \in \mathsf{Hid}_{\Delta}} \mathbf{h}[i]^{\delta_{2,i}}$.
  2. *computes the challenge* $c_2 \leftarrow H(\bar{A}_{\Delta}, \bar{B}_{\Delta}, U_{\Delta}, \mathsf{del}, \mathsf{nonce})$*;*
  3. *computes a response* z_2*:*
     - $s_2 \leftarrow \alpha_2 + r_2^{-1} \cdot e_{\Delta} \cdot c_2$*;*
     - $n_2 \leftarrow \beta_2 + r_2^{-1} \cdot c_2$*;*
     - $u_{2,i} \leftarrow \delta_{2,i} - \mathbf{a}_{\Delta}[i]\, c_2 \quad \forall\, i \in \mathsf{Hid}_{\Delta}$*;*
     - $z_2 \leftarrow (s_2, n_2, (u_{2,i})_{i \in \mathsf{Hid}_{\Delta}})$.
  4. *sets* $\pi_{\mathsf{del}} \leftarrow (\bar{A}_{\Delta}, \bar{B}_{\Delta}, c_2, z_2)$

5. *returns* $\mathsf{pres} \leftarrow (\mathsf{del}, \pi_{\mathsf{del}})$.

- Delegated presentation verification*:* $\{0,1\} \xleftarrow{\$} \mathsf{DelegPresVer}(\mathsf{pres}, \mathsf{pk}_{\mathsf{Iss}})$.

  Parse

  $$\mathsf{pres} \rightarrow (\mathsf{del}, \pi_{\mathsf{del}}), \quad \mathsf{del} \rightarrow (\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}}),$$
  $$\pi_{\mathsf{DP}} \rightarrow (\bar{A}_{\mathsf{D}}, \bar{B}_{\mathsf{D}}, c_1, z_1), \qquad \pi_{\mathsf{del}} = (\bar{A}_{\Delta}, \bar{B}_{\Delta}, c_2, z_2).$$

  The verifier checks that

  1. *the delegation is valid, i.e.* $1 \leftarrow \mathsf{DelegVer}(\mathsf{del}, \mathsf{pk}_{\mathsf{Iss}})$*;*
  2. π_{del} *is a valid proof for the attributes in* Δ_{ID} *specified in* del*:*
     - *computes* $\tilde{U}_{\Delta}$*:*

       $$\tilde{U}_{\Delta} \leftarrow \bar{A}_{\Delta}^{s_2} \bar{B}_{\Delta}^{n_2} \prod_{i \in \mathsf{Hid}_{\Delta}} \mathbf{h}[i]^{u_{2,i}} C_{\mathsf{Rev}_{\Delta}}(\mathbf{a}_{\Delta})^{-c_2};$$

     - *checks that*

       $$c_2 \stackrel{?}{=} H(\bar{A}_{\Delta}, \bar{B}_{\Delta}, \tilde{U}_{\Delta}, \mathsf{del}, \mathsf{nonce}),$$
       $$\mathbf{e}(\bar{A}_{\Delta}, X_2) \stackrel{?}{=} \mathbf{e}(\bar{B}_{\Delta}, g_2).$$

  3. *the value* scope *included in* π_{DP} *is satisfied according to the associated verification procedure.*

  If these checks pass, the delegated presentation is accepted, and the algorithm outputs 1.

6.4 Security Analysis

In this section we prove that

- the VP delegation scheme described in Definition 6.3.1 satisfies the security properties of correctness (Definition 6.2.2) and unforgeability (Definition 6.2.3).
- the VP delegation scheme described in Definition 6.3.2 satisfies the security properties of correctness (Definition 6.2.2), unforgeability (Definition 6.2.3)

and unlinkability of the delegated presentation (Definition 6.2.5). The unlinkability of the delegation issuance (Definition 6.2.4) can be proved using a similar argument.

6.4.1 Correctness

Theorem 6.4.1 *The delegation scheme described in Definition 6.3.1 instantiated using the digital signature $\mathscr{DS}$ is correct, assuming that the digital signature $\mathscr{DS}$ is correct and that H is modeled as a random oracle.*

It is easy to see that the VP delegation scheme is correct. In fact, the delegator always computes a proof π_{DP} to include in del, and, since the digital signature $\mathscr{DS}$ is correct, del is a valid delegation. Then, if the delegation is presented by a delegatee whose identity matches with the identity Δ_{ID} included in del, the delegatee can always successfully present the delegation, again because $\mathscr{DS}$ is correct.

Theorem 6.4.2 *The delegation scheme described in Definition 6.3.2 instantiated using the digital signature BBS is correct, assuming that H is modeled as a random oracle.*

In this case, the correctness follows directly from the correctness property of the underlying NIZKP for BBS signatures.

6.4.2 Unforgeability

We observe that in the definition of the Experiment 4, it is not well defined what we mean by $\mathsf{del}^\star$ (or $\pi_{\mathsf{del}^\star}$) "is not generated using any credential issued to $\mathscr{A}$". In fact, this sentence has a different meaning according to the amount of information a presentation reveals about the credential used to generate it. More specifically, if we consider the delegation of presentation generated using mdoc VCs (as defined in Definition 6.3.1), the presentation and delegation contain an identifier of the VC used to generate them, namely $\mathsf{pk}_{\mathsf{cred}}$, and all the commitments to the attributes. Therefore, in this case, we say that the presentation is not generated using any credential issued to $\mathscr{A}$ if the information that identifies the credential used to generate the presentation does not match any of the credentials issued to $\mathscr{A}$. For the delegation of anonymous

presentations, as those built on top of BBS credential (as defined in Definition 6.3.2), the meaning is different: since the presentation hides the credential used to generate it, and the only link between the presentation (or delegation) and the credential used to generate it are the issuer of the credential $\mathsf{pk}_{\mathsf{Iss}}$ and the attribute revealed, we say that a forgery has not been generated using any credential issued to $\mathscr{A}$ if the adversary has never been issued a credential by $\mathsf{pk}_{\mathsf{Iss}}$ for the attributes revealed in the forgery.

In the security proof of unforgeability of the delegation scheme for mdoc VCs (as defined in Definition 6.3.1), we must introduce a technicality that simplifies the description of our reduction. In particular, we show that the adversary $\mathscr{A}$ to the unforgeability of the VP delegation scheme can be used to build a reduction to a *variant* of the strong unforgeability under chosen message attacks (SUF-CMA) of the digital signature $\mathscr{DS}$. In this variant, which we refer to as *v-SUF-CMA*, the reduction can open a polynomial number of sessions of standard SUF-CMA experiments (each associated with a different public key with the same public parameters pp), and the reduction wins the v-SUF-CMA experiment if it produces a forgery for at least one of the public keys provided by the challenger of the experiment. Informally, the queries that the adversary can make are (1) queries for the opening of a new session identified by a counter i, for which it receives a public key pk_i, and (2) signing queries specifying a message and one of the public keys it has previously received (m, pk_j), for which it receives a valid signature σ of m under the public key pk_j. It is easy to see that an adversary who can win the SUF-CMA experiment can be turned into an adversary of the v-SUF-CMA experiment, and the success probability remains the same. However, it is also easy to see that an adversary of v-SUF-CMA can be used as a subroutine for a reduction to the SUF-CMA with a loss in the probability of success proportional to the number of sessions the adversary can open.

This technicality is needed to capture that an adversary of the delegation scheme for mdoc VCs could win the experiment by forging the public key of the issuer, or by forging the credential public keys, for which it does not know the secret counterpart, that it sees during the unforgeability experiment training (for example the public keys included in the delegations queried to $\mathsf{O}_{\mathsf{del}}$ and in the presentations queried to $\mathsf{O}_{\mathsf{pres}}$). Therefore, this issue does not arise when evaluating the security of the delegation scheme for BBS anonymous credentials (where π_{del} and π_{pres} are NIZKP of a BBS credential under $\mathsf{pk}_{\mathsf{Iss}}$ and do not contain any other pk), because it reduces

to the strong unforgeability of the BBS signature scheme instantiated with the public key $\mathsf{pk}_{\mathsf{Iss}}$. This remarkably simplifies the security analysis of VP delegation schemes built on top of BBS ACs, compared to the mdoc VCs.

Theorem 6.4.3 *The delegation scheme described in Definition 6.3.1 is unforgeable according to Definition 6.2.3 assuming that the digital signature $\mathscr{DS}$ is strongly v-UF-CMA (v-SUF-CMA) unforgeable and that H is a collision resistant cryptographic hash function*[7].

We prove that if there exists an adversary $\mathscr{A}$ of Experiment 4 who can win the experiment with non-negligible probability, then we can use $\mathscr{A}$ to build a reduction $\mathscr{B}$ to the v-SUF-CMA experiment for the underlying digital signature scheme $\mathscr{DS}$, which wins the experiment with non-negligible probability, or that breaks the collision resistance of the underlying hash function.

Setup $\mathscr{B}$ sends a query for a new public key in the v-SUF-CMA experiment. $\mathscr{C}$ sends to $\mathscr{B}$ a public key pk and public parameters pp. $\mathscr{B}$ sets $\mathsf{pk}_{\mathsf{Iss}} \leftarrow \mathsf{pk}$, and forwards $(\mathsf{pp}, \mathsf{pk}_{\mathsf{Iss}})$ to the adversary $\mathscr{A}$.

Training phase We show how $\mathscr{B}$ answers to the queries $\mathscr{A}$ can send during the training phase.

- Queries to $\mathsf{O}_{\mathsf{iss}}$: $\mathscr{A}$ sends in input a set of attributes $\{a_i\}_{i\in[l]}$. $\mathscr{B}$ performs the following operations:
  1. it generates a key pair $(\mathsf{sk}_{\mathsf{cred}}, \mathsf{pk}_{\mathsf{cred}})$[8], the salts $\{\mathsf{salt}_i\}_{i\in[l]}$, computes a commitment $\mathsf{com}_i = \mathsf{RO}(a_i||\mathsf{salt}_i)$ (where RO is a random oracle programmed by $\mathscr{B}$) to $a_i, \forall i \in [l]$;
  2. it sends a signing query to $\mathscr{C}$ with input $((\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}}), \mathsf{pk}_{\mathsf{Iss}})$[9] : $\mathscr{C}$ returns a signature σ of $(\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}})$ using $\mathsf{sk}_{\mathsf{Iss}}$.
  3. sends to $\mathscr{A}$ the credential $\mathsf{cred} \leftarrow ((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}}), \{a_i\}_{i\in[l]}, \{\mathsf{salt}_i\}_{i\in[l]}, \mathsf{sk}_{\mathsf{cred}})$ and adds cred to CT.

[7] To prove unforgeability we need the commitment scheme used in mdoc VCs (Definition 3.4.1) to be binding. The hiding property is useful to enable the selective disclosure of attributes.

[8] For clarity of presentation, the subscripts D (delegator) and Δ (delegatee) are omitted in the notation. The credential issuance procedure is the same for both roles and can be applied to either party.

[9] Note that this is the format for a signing query in the v-SUF-CMA experiment, given by the message to be signed, and the public key that will verify the signature.

- Queries to $\mathsf{O}_{\mathsf{del}}$: $\mathscr{A}$ can query for a delegation for the values $(\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP})$ of its choice. Upon receiving a query $\mathscr{B}$ performs the following operations:
  1. generates a set of random attributes $\{a_{\mathsf{D},i}\}_{i\in[l]}$ which satisfies DP, and computes the commitments com_i as before generating the salts and programming the random oracle;
  2. opens a new session of SUF-CMA with $\mathscr{C}$ from which it receives a public key $\mathsf{pk}_{\mathsf{cred}_\mathsf{D}}$ and sends a signing query $((\{\mathsf{com}_{\mathsf{D},i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_\mathsf{D}}), \mathsf{pk}_{\mathsf{Iss}})$ for a signature with $\mathsf{sk}_{\mathsf{Iss}}$ of $(\{\mathsf{com}_{\mathsf{D},i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_\mathsf{D}})$, and receives from $\mathscr{C}$ the signature σ_D;
  3. sends a signing query $((\mathsf{pres}', \mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}}), \mathsf{pk}_{\mathsf{cred}_\mathsf{D}})$ to $\mathscr{C}$ for a signature with $\mathsf{sk}_{\mathsf{cred}_\mathsf{D}}$ of $\mathsf{pres}' = ((\sigma_\mathsf{D}, \{\mathsf{com}_{\mathsf{D},i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_\mathsf{D}}), \{\mathsf{salt}_{\mathsf{D},i}\}_{i\in\mathsf{DP}})$. $\mathscr{C}$ returns the signature σ' ;
  4. $\mathscr{B}$ sets $\pi_{\mathsf{DP}} = (\sigma', \mathsf{pres}')$ sends to $\mathscr{A}$ the delegation $\mathsf{del} = (\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}})$ and adds del to DT.

  It is important that the reduction $\mathscr{B}$ does not choose the secret key $\mathsf{sk}_{\mathsf{cred}_\mathsf{D}}$ of the credential used to create the delegation del because the adversary $\mathscr{A}$ might create a forgery using the same public key $\mathsf{pk}_{\mathsf{cred}}$ and the reduction must be able to exploit that forgery.
- Queries to $\mathsf{O}_{\mathsf{pres}}$: $\mathscr{A}$ can query for a delegated presentation giving in input $(\mathsf{del}, \mathsf{nonce})$ of its choice. Upon receiving a query $\mathscr{B}$ performs the following operations:
  1. generates a set of random attributes $\{a_{\Delta,i}\}_{i\in[l]}$ which satisfies Δ_{ID}, and computes the commitments com_i as before generating the salts and programming the random oracle;
  2. opens a new session of SUF-CMA with $\mathscr{C}$ from which it receives a public key $\mathsf{pk}_{\mathsf{cred}_\Delta}$ and sends a signing query $((\{\mathsf{com}_{\Delta,i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_\Delta}), \mathsf{pk}_{\mathsf{Iss}})$ for a signature with $\mathsf{sk}_{\mathsf{Iss}}$ of $(\{\mathsf{com}_{\Delta,i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_\Delta})$, and receives from $\mathscr{C}$ the signature σ;
  3. $\mathscr{B}$ sends a signing query $(\mathsf{pres}''||\mathsf{del}, \mathsf{pk}_{\mathsf{cred}_\Delta})$ to $\mathscr{C}$ or a signature with $\mathsf{sk}_{\mathsf{cred}_\Delta}$ of $\mathsf{pres}'' = ((\sigma_\Delta, \{\mathsf{com}_{\Delta,i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_\Delta}), \{\mathsf{salt}_{\Delta,i}\}_{i\in\Delta_{\mathsf{ID}}}, \mathsf{nonce})$. $\mathscr{C}$ returns the signature σ''.
  4. $\mathscr{B}$ sends $\mathsf{pres} = (\mathsf{del}, (\sigma'', \mathsf{pres}''))$ to $\mathscr{A}$ and stores pres in PT.

Note that $\mathscr{B}$ knows only the secret keys associated to the credential issued to $\mathscr{A}$. The other $\mathsf{sk}_{\mathsf{cred}}$ are not known neither to $\mathscr{B}$, nor to $\mathscr{A}$. Therefore if $\mathscr{A}$ manages to forge the corresponding public keys $\mathsf{pk}_{\mathsf{cred}}$ that are included in the delegations or delegated presentations, the reduction can re-use such forgeries to win the v-SUF-CMA experiment.

Forgery Phase $\mathscr{A}$ sends to $\mathscr{B}$ a delegated presentation $\mathsf{pres}^\star = (\mathsf{del}^\star, \pi_{\mathsf{del}^\star})$ as its forgery.

We argue that this presentation is a forgery, i.e. it satisfies at least one of the winning conditions of Experiment 4.

Instantiation and analysis of the winning condition of Experiment 4 We now focus on the winning condition of Experiment 4, and we rewrite it considering the experiment instantiated with the ARF-Compliant VP delegation Scheme in Definition 6.3.1.

The adversary's output is a presentation $\mathsf{pres}^\star = (\mathsf{del}^\star, \pi_{\mathsf{del}^\star})$, with $\mathsf{del}^\star = ({\Delta_{\mathsf{ID}}}^\star, \mathsf{scope}^\star, \mathsf{DP}^\star, \pi_{\mathsf{DP}^\star})$. We recall the structure of the proofs $\pi_{\mathsf{DP}^\star}$ and $\pi_{\mathsf{del}^\star}$.

1. $\pi_{\mathsf{DP}^\star}$ is a presentation of a credential for the attributes specified in $\mathsf{DP}^\star$ containing also $\mathsf{scope}^\star$ and ${\Delta_{\mathsf{ID}}}^\star$:

$$\pi_{\mathsf{DP}^\star} = \Bigg(\underbrace{((\sigma_{\mathsf{D}}, \{\mathsf{com}_{\mathsf{D},i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}}), \{\mathsf{salt}_{\mathsf{D},i}\}_{i\in \mathsf{DP}^\star})}_{\mathsf{pres}'},\ \sigma' = \mathsf{Sign}((\mathsf{pres}', \mathsf{DP}^\star, \mathsf{scope}^\star, {\Delta_{\mathsf{ID}}}^\star), \mathsf{sk}_{\mathsf{cred}_{\mathsf{D}}}) \Bigg).$$

with σ' being the signature of pres' using the secret key associated to $\mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}}$ and σ_{D} is the signature of $(\{\mathsf{com}_{\mathsf{D},i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}})$ (part of $\mathsf{cred}_{\mathsf{D}}$) using $\mathsf{sk}_{\mathsf{Iss}}$.

2. $\pi^\star_{\mathsf{del}}$ is a presentation of a credential cred_Δ for the attributes in Δ_{ID} bounded to $\mathsf{del}^\star$ and $\mathsf{nonce}^\star$, in particular:

$$\pi_{\mathsf{del}^\star} = \Bigg(\underbrace{((\sigma_\Delta, \{\mathsf{com}_{\Delta,i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_\Delta}), \{\mathsf{salt}_{\Delta,i}\}_{i\in{\Delta_{\mathsf{ID}}}^\star}, \mathsf{nonce}^\star)}_{\mathsf{pres}''},\ \sigma'' = \mathsf{Sign}((\mathsf{pres}'', \mathsf{del}^\star), \mathsf{sk}_{\mathsf{cred}_\Delta}) \Bigg).$$

with σ'' being the signature of pres'' using the secret key associated to $\mathsf{pk}_{\mathsf{cred}_\Delta}$ and σ_Δ is the signature of $(\{\mathsf{com}_{\Delta,i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_\Delta})$ (part of cred_Δ) using $\mathsf{sk}_{\mathsf{Iss}}$.

We make the following observation.

The challenger of the unforgeability experiment (Experiment 4) generates public keys $\mathsf{pk}_{\mathsf{cred}}$ and signs them (together with a set of commitments $\{\mathsf{com}_i\}_{i\in[l]}$) using $\mathsf{sk}_{\mathsf{Iss}}$ as a consequence of:

1. issuance queries: in this case, the challenger also gives to the adversary the associated secret key and adds the credential to CT;
2. delegation queries: in this case, the challenger also signs pres' in π_{DP} using the secret key $\mathsf{sk}_{\mathsf{cred}}$ associated to $\mathsf{pk}_{\mathsf{cred}}$, adds the delegation to DT, and does not reveal $\mathsf{sk}_{\mathsf{cred}}$ to the adversary.
3. delegated presentation queries: in this case, the issuer also signs pres'' in π_{del} using the secret key $\mathsf{sk}_{\mathsf{cred}}$ associated to $\mathsf{pk}_{\mathsf{cred}}$, adds the delegation to PT, and does not reveal $\mathsf{sk}_{\mathsf{cred}}$ to the adversary.

Note that the challenger of Experiment 4 never signs public keys it has not generated .[10]

The adversary wins the experiment if $1 \leftarrow \mathsf{DelegPresVer}(\mathsf{pres}^\star)$ and at least one of the following conditions is satisfied:

1. *forgery of the delegation*: $\mathsf{del}^\star$ is forged, i.e.
   - it is not a delegation queried by $\mathscr{A}$ to $\mathsf{O}_{\mathsf{del}}$, i.e. $\mathsf{del}^\star \notin \mathsf{DT}$;
   - $\mathsf{del}^\star$ was not generated using the credentials issued to $\mathscr{A}$ meaning that

   Given $\mathsf{del}^\star = ({\Delta_{\mathsf{ID}}}^\star, \mathsf{scope}^\star, \mathsf{DP}^\star, \pi_{\mathsf{DP}^\star})$ with

$$\pi_{\mathsf{DP}^\star} = \left(\mathsf{pres}'^\star, \sigma'^\star = \mathsf{Sign}((\mathsf{pres}', \mathsf{DP}^\star, \mathsf{scope}^\star, {\Delta_{\mathsf{ID}}}^\star), \mathsf{sk}^\star_{\mathsf{cred}_\mathsf{D}})\right),$$

$$\mathsf{pres}'^\star = ((\sigma^\star_\mathsf{D}, \{\mathsf{com}^\star_{\mathsf{D},i}\}_{i\in[l]}, \mathsf{pk}^\star_{\mathsf{cred}_\mathsf{D}}), \{\mathsf{salt}^\star_{\mathsf{D},i}\}_{i\in\mathsf{DP}}).$$

   we say that $\mathsf{del}^\star$ was not generated using any credential issued to $\mathscr{A}$ if $\forall\, \mathsf{cred} \in \mathsf{CT}$, $\mathsf{cred} = \big((\sigma_\mathsf{D}, \{\mathsf{com}_{\mathsf{D},i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_\mathsf{D}}), \{a_{\mathsf{D},i}\}_{i\in[l]}, \{\mathsf{salt}_{\mathsf{D},i}\}_{i\in[l]}, \mathsf{sk}_{\mathsf{cred}_\mathsf{D}}\big)$,

[10] For the sake of clarity, note that in the security proof, the challenger of Experiment 4 is the reduction $\mathscr{B}$. The reduction does not know the secret key $\mathsf{sk}_{\mathsf{Iss}}$ but can ask for signatures of messages of its choice using $\mathsf{sk}_{\mathsf{Iss}}$ sending queries to $\mathscr{C}$, the challenger of the v-SUF-CMA experiment.

$$(\mathsf{pres}'^{\star}, \mathsf{DP}^{\star}{}^{11}) \neq \left((\sigma_{\mathsf{D}}, \{\mathsf{com}_{\mathsf{D},i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}}), \{\mathsf{salt}_{\mathsf{D},i}\}_{i\in\mathsf{DP}^{\star}}, \{a_{\mathsf{D},i}\}_{i\in\mathsf{DP}^{\star}}\right),$$

because these are the information of the credential used to generate a delegation that appear in the delegation itself.

The public key $\mathsf{pk}^{\star}_{\mathsf{cred}_{\mathsf{D}}}$ in $\mathsf{del}^{\star}$ has been generated either by $\mathscr{A}$, by $\mathscr{B}$, or by $\mathscr{C}$.

$\mathsf{pk}^{\star}_{\mathsf{cred}_{\mathsf{D}}}$ is generated by $\mathscr{A}$. In this case none of the credentials issued to $\mathscr{A}$ is related to $\mathsf{pk}^{\star}_{\mathsf{cred}_{\mathsf{D}}}$. This means that the signature $\sigma^{\star}_{\mathsf{D}}$ of $(\{\mathsf{com}^{\star}_{\mathsf{D},i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}})$ must be a forgery of $\mathsf{pk}_{\mathsf{Iss}}$ because $\mathscr{C}$ signs with $\mathsf{sk}_{\mathsf{Iss}}$ only public keys generated by $\mathscr{B}$ or $\mathscr{C}$ (according to our reduction definition);

$\mathsf{pk}^{\star}_{\mathsf{cred}_{\mathsf{D}}}$ is generated by $\mathscr{B}$. $\mathsf{pk}^{\star}_{\mathsf{cred}_{\mathsf{D}}}$ was included in a credential issued by $\mathsf{O}_{\mathsf{iss}}$, but the delegation $\mathsf{del}^{\star}$ is not generated using this credential. This means that one of the following events happens:

(a) the commitments $\{\mathsf{com}^{\star}_{\mathsf{D},i}\}_{i\in[l]} \neq \{\mathsf{com}_{\mathsf{D},i}\}_{i\in[l]} \vee \sigma^{\star}_{D} \neq \sigma_{D}$: in this case the adversary has broken the v-SUF-CMA of the $\mathsf{pk}_{\mathsf{Iss}}$;

(b) $\mathsf{DP}^{\star} \neq \{a_{\mathsf{D},i}\}_{i\in\mathsf{DP}^{\star}}$: in this case it is possible to reduce to the binding property of the commitment scheme (and hence to the collision resistance of H);

(c) $\{\mathsf{salt}^{\star}_{\mathsf{D},i}\}_{i\in\mathsf{DP}} \neq \{\mathsf{salt}_{\mathsf{D},i}\}_{i\in\mathsf{DP}}$ are different, and in this case it is possible to reduce to the collision resistance of H.

$\mathsf{pk}^{\star}_{\mathsf{cred}_{\mathsf{D}}}$ is generated by $\mathscr{C}$. $\mathsf{pk}^{\star}_{\mathsf{cred}_{\mathsf{D}}}$ has been generated by $\mathscr{C}$ during the generation of the $\mathsf{pk}_{\mathsf{Iss}}$, or to generate the response to queries to $\mathsf{O}_{\mathsf{del}}$ or $\mathsf{O}_{\mathsf{pres}}$. Then, neither $\mathscr{A}$ nor $\mathscr{B}$ know the secret counterpart $\mathsf{sk}_{\mathsf{cred}_{\mathsf{D}}}$. Also, since $\mathsf{del}^{\star} \notin \mathsf{DT}$, the challenger has never signed with $\mathsf{sk}^{\star}_{\mathsf{cred}_{\mathsf{D}}}$ the message $((\sigma_{\mathsf{D}}, \{\mathsf{com}_{\mathsf{D},i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}}), \{\mathsf{salt}_{\mathsf{D},i}\}_{i\in\mathsf{DP}^{\star}}, \mathsf{DP}^{\star}, \mathsf{scope}^{\star}, \Delta_{\mathsf{ID}}{}^{\star})$, therefore σ', included in $\pi_{\mathsf{DP}^{\star}}$, is a forgery of $\mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}}$.

In this way we have shown that our reduction can reduce to the v-SUF-CMA of the signature scheme, to the binding property of the commitment scheme or to the collision resistance of the underlying hash function.

2. *forgery of the delegated presentation*: $\pi_{\mathsf{del}^{\star}}$ is forged, i.e.

[11] Depending on the context, $\mathsf{DP}^{\star}$ denotes either the set of revealed attributes appearing in the delegation or the corresponding set of indices selecting those attributes.

- it is not a presentation queried by $\mathscr{A}$, i.e. $\mathsf{pres}^\star \notin \mathsf{PT}$;
- $\pi_{\mathsf{del}^\star}$ was not generated using the credentials issued to $\mathscr{A}$.

Given

$$\pi_{\mathsf{del}^\star} = \left(\mathsf{pres}'', \sigma'' = \mathsf{Sign}((\mathsf{pres}'', , \mathsf{del}^\star), \mathsf{sk}_{\mathsf{cred}_\Delta})\right),$$

$$\mathsf{pres}'' = ((\sigma_\Delta, \{\mathsf{com}_{\Delta,i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_\Delta}), \{\mathsf{salt}_{\Delta,i}\}_{i\in\Delta_{\mathsf{ID}}}, \mathsf{nonce})$$

we say $\pi_{\mathsf{del}^\star}$ was not generated using any credential issued to $\mathscr{A}$ if $\forall \mathsf{cred} \in \mathsf{CT}$, $\mathsf{cred} = \big((\sigma_\Delta, \{\mathsf{com}_{\Delta,i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_\Delta}), \{a_{\Delta,i}\}_{i\in[l]}, \{\mathsf{salt}_{\Delta,i}\}_{i\in[l]}, \mathsf{sk}_{\mathsf{cred}_\Delta}\big)$, defining $\overline{\mathsf{pres}''}$ removing $\mathsf{nonce}^\star$ from pres'',

$$\big(\overline{\mathsf{pres}''}, {\Delta_{\mathsf{ID}}}^\star\big) \neq \big((\sigma_\Delta, \{\mathsf{com}_{\Delta,i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_\Delta}), \{\mathsf{salt}_{\Delta,i}\}_{i\in{\Delta_{\mathsf{ID}}}^\star}, \{a_{\Delta,i}\}_{i\in{\Delta_{\mathsf{ID}}}^\star}\big)$$

Using a similar observation, it is possible to show that even in this case our reduction can reduce to the v-SUF-CMA of the underlying signature scheme or the binding property of the commitment scheme or the collision resistance of the hash function used to define the commitment.

This allows us to prove that an $\mathscr{A}$ winning the unforgeability of the delegation scheme experiment with non-negligible probability, can be used as a subroutine for a reduction winning one of the following experiments with non-negligible probability:

- the strong v-UF-CMA experiment of the underlying signature scheme or
- the collision resistance of the underlying hash function used to instantiate the random oracle.

means that at least one of the public keys that $\mathscr{B}$ has received by $\mathscr{C}$, corresponding to the sessions opened in the v-SUF-CMA experiment, has been forged. Therefore $\mathscr{B}$ win the v-SUF-CMA experiment, and this concludes the security proof.

Theorem 6.4.4 *The delegation scheme described in Definition 6.3.2 is unforgeable according to Definition 6.2.3 under the assumption that BBS is strongly unforgeable* [12]

[12] Which holds under q-Strong Diffie-Hellman assumption [74].

Since in this case the proofs included in the delegations and delegated presentations are NIZKPs bound to a specific context (for delegations $(\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP})$, whereas for delegated presentations a delegation del), we can leverage on the existence of a simulator that, by programming the random oracle can generate proofs that are indistinguishable from real proofs. Therefore, in this case, there is not need for the reduction to generate the credentials used to generate the proofs π_{DP} and π_{del}. This remarkably simplifies the security analysis making it more linear, and enabling the design of a reduction to the unforgeability of the BBS signature scheme.

We prove that if there exists an adversary $\mathscr{A}$ of Experiment 4 who can win the experiment with non-negligible probability, then we can use $\mathscr{A}$ to build a reduction $\mathscr{B}$ to the SUF-CMA experiment of the BBS signature scheme, which wins the experiment with non-negligible probability. The reduction $\mathscr{B}$ interacts with the challenger $\mathscr{C}$ of the SUF-CMA experiment and simulates the challenger of Experiment 4.

Setup $\mathscr{B}$ receives the public parameters pp and the public key $\mathsf{pk}_{\mathsf{Iss}}$ of the BBS signature scheme from the challenger $\mathscr{C}$ of the SUF-CMA experiment. $\mathscr{B}$ forwards $(\mathsf{pp}, \mathsf{pk}_{\mathsf{Iss}})$ to the adversary $\mathscr{A}$.

Training phase We show how $\mathscr{B}$ answers to the queries $\mathscr{A}$ can send during the training phase.

- Queries to $\mathsf{O}_{\mathsf{iss}}$: $\mathscr{A}$ sends in input a set of attributes $\{a_i\}_{i\in[l]}$. $\mathscr{B}$ performs the following operations:
  1. it sends a signing query to $\mathscr{C}$ with input $\{a_i\}_{i\in[l]}$: $\mathscr{C}$ returns a signature σ of $\{a_i\}_{i\in[l]}$ generated using $\mathsf{sk}_{\mathsf{Iss}}$.
  2. $\mathscr{B}$ sends to $\mathscr{A}$ the credential $\mathsf{cred} \leftarrow (\sigma, \{a_i\}_{i\in[l]})$ and adds cred to CT.
- Queries to $\mathsf{O}_{\mathsf{del}}$: $\mathscr{A}$ can query for a delegation for the values $(\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP})$ of its choice. Upon receiving a query $\mathscr{B}$ performs the following operations:
  1. runs the simulator of the presentation protocol generating a proof $\pi_{\mathsf{DP}} = (\bar{A}_{\mathsf{D}}, \bar{B}_{\mathsf{D}}, c_1, z_1)$ for the statement DP by programming the random oracle H on the input $(\bar{A}_{\mathsf{D}}, \bar{B}_{\mathsf{D}}, U_{\mathsf{D}}, \mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}})$. This operation overwrites the random oracle only with negligible probability because the values $\bar{A}_{\mathsf{D}}, \bar{B}_{\mathsf{D}}, U_{\mathsf{D}}$ generated by the simulator have super-logarithmic min-entropy [74].

2. $\mathscr{B}$ sends to $\mathscr{A}$ the delegation $\mathsf{del} = (\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}})$ and adds del to DT.

- Queries to $\mathsf{O}_{\mathsf{pres}}$: $\mathscr{A}$ can query for a delegated presentation giving in input $(\mathsf{del}, \mathsf{nonce})$ of its choice. Upon receiving a query $\mathscr{B}$ performs the following operations:

  1. runs the simulator of the presentation protocol generating a proof $\pi_{\mathsf{del}} = (\bar{A}_\Delta, \bar{B}_\Delta, c_2, z_2)$ for the statement Δ_{ID} by programming the random oracle H on the input $(\bar{A}_\Delta, \bar{B}_\Delta, U_\Delta, \mathsf{del})$. This operation overwrites the random oracle only with negligible probability because the values $\bar{A}_\Delta, \bar{B}_\Delta, U_\Delta$ generated by the simulator have super-logarithmic min-entropy [74].
  2. $\mathscr{B}$ sends to $\mathscr{A}$ the presentation $\mathsf{pres} = (\mathsf{del}, \pi_{\mathsf{del}})$ and adds pres to PT.

Forgery Phase $\mathscr{A}$ sends to $\mathscr{B}$ a delegated presentation $\mathsf{pres}^\star = (\mathsf{del}^\star, \pi_{\mathsf{del}^\star})$ as its forgery.

To be a valid forgery, this presentation must satisfy at least one of the winning conditions of Experiment 4.

Instantiation and analysis of the winning condition of Experiment 1 Note that, since we are describing a VP delegation scheme that is built on top of an anonymous credential scheme, the proofs generated by delegator and delegatee hide the anonymous credential used to generate it. Therefore, in this case the statement "$\mathsf{del}^\star$ (or $\pi_{\mathsf{del}^\star}$) is not generated using any credential issued to $\mathscr{A}$" means that none of the issued credentials can be used to generate the associated proof π_{DP} (or indeed $\pi_{\mathsf{del}^\star}$) included in $\mathsf{del}^\star$ (or in $\mathsf{pres}^\star$), i.e. none of the credentials issued to the adversary satisfy the statements, or set of attributes in DP (or Δ_{ID}).

We consider separately the two cases:

1. $\mathsf{del}^\star$ *is forged, i.e.* $\mathsf{del}^\star \notin \mathsf{DT}$ *and* $\forall \mathsf{cred} \in \mathsf{CT}$*, being* $\mathbf{A}$ *the set of attributes in* cred*, then* $\mathsf{DP}^\star \not\subseteq \mathbf{A}$.

   Since the proof $\pi_{\mathsf{DP}^\star}$ is a NIZKP obtained by applying the Fiat-Shamir transform to a sigma protocol, it is possible to run the Fiat-Shamir extractor and extract a BBS signature. This is done by rewinding $\mathscr{A}$ to the moment it has sent the random oracle query associated to the

challenge associated to the proof $\pi_{\mathsf{DP}^\star}$, and changing the associated digest. With non-negligible probability $\mathscr{A}$ will produce another valid forgery returning two transcript of the sigma protocol for the same first message but two different challenges. From this $\mathscr{B}$ can extract a witness, i.e. a BBS signature $\sigma^\star$, for the statement DP. But the adversary was never issuer a BBS signature for the attributes in DP and this means that $\mathscr{B}$ has never received from $\mathscr{C}$ the signature $\sigma^\star$ and can win the SUF-CMA experiment for BBS signatures by sending $\sigma^\star$ to $\mathscr{C}$.

2. $\pi_{\mathsf{del}^\star}$ *is forged,* $\forall \mathsf{pres} \in \mathsf{PT}$, $\mathsf{pres} = (\mathsf{del}, \pi_{\mathsf{del}})$, $\pi_{\mathsf{del}} \neq \pi_{\mathsf{del}^\star}$ *and* $\forall \mathsf{cred} \in \mathsf{CT}$*, being* $\mathbf{A}$ *the set of attributes in* cred*, then* ${\Delta_{\mathsf{ID}}}^\star \not\subseteq \mathbf{A}$. This follows from a similar argument as the previous item. It is possible to show that the reduction can extract a BBS signature $\sigma^\star$ for the attributes in Δ_{ID} that were never issued to $\mathscr{A}$, and therefore it was not generated by $\mathscr{C}$. This means that $\mathscr{B}$ can win the SUF-CMA experiment for BBS signatures by returning to $\mathscr{C}$ the signature $\sigma^\star$.

6.4.3 Unlinkability

In this section, we focus on proving the *unlinkability* of delegated presentations for the delegation scheme described in Definition 6.3.2. We note that the unlinkability of the delegation issuance proof follows from the same argument, since the interactive proof used in DelegIssuance employs the same building blocks as the one used in DelegPres.

The proof is presented showing the existence of a probabilistic polynomial-time simulator $\mathscr{S}$ that can generate a simulated presentation indistinguishable from that produced by an honest delegatee holding a valid BBS credential. In fact, if such a simulator exists, this implies that no adversary can distinguish between two honest delegated presentations derived from different credentials, as long as they are consistent with the same Δ_{ID}. We formalize this argument by explicitly describing an indistinguishability experiment and the associated security definition.

This explicit description is useful to relate two kinds of definition of unlinkability that are used in the literature: one stronger, which requires the existence of a simulator that generates presentations that are indistinguishable from honestly generated ones, and a weaker one where the adversary generates and sends to the challenger two

credentials and it must guess which of the two credentials was used to generate a presentation it receives from the challenger.

Indistinguishability Experiment and Definition. The goal of this experiment is to ensure that the delegated presentation generated using a real anonymous credential is indistinguishable from one generated by a simulator $\mathscr{S}$ that does not possess the credential. The existence of such a simulator implies that the delegated presentation does not reveal any information about the underlying credential. In particular, we will show that if no efficient adversary can win the indistinguishability experiment with non-negligible advantage, then no adversary can win the unlinkability experiment with non-negligible advantage — that is, no adversary can distinguish which of two credentials was used to produce a given delegated presentation with probability significantly greater than $\frac{1}{2}$.

Experiment 7 ($\mathsf{Exp}^{\mathsf{Indistinguishability}}_{\mathscr{A}}(1^{\lambda})$)

1. *The adversary $\mathscr{A}$ generates the public parameters* pp *for the verifiable credential scheme together with the issuer key pair* $(\mathsf{sk}_{\mathsf{Iss}}, \mathsf{pk}_{\mathsf{Iss}})$ *and send* pp *and* $\mathsf{pk}_{\mathsf{Iss}}$ *to the challenger.*
2. *The challenger samples a random bit* $b' \xleftarrow{\$} \{0,1\}$*;*
3. *The adversary $\mathscr{A}$ generates and sends to the challenger* del, nonce *and* cred_{Δ} *satisfying* Δ_{ID}*;* [13]
4. *The challenger computes* pres *as follows:*
   - *If* $b' = 0$*, computes* $\mathsf{pres} \xleftarrow{\$} \mathscr{S}(\mathsf{del}, \mathsf{nonce})$
   - *If* $b' = 1$*, computes* $\mathsf{pres} \xleftarrow{\$} \mathsf{DelegPres}(\mathsf{del}, \mathsf{cred}_{\Delta}, \mathsf{nonce})$.
5. *The challenger sends* pres *to* $\mathscr{A}$.
6. *$\mathscr{A}$ outputs a bit* $\bar{b}'$.
7. *$\mathscr{A}$ wins if* $\bar{b}' = b'$.

[13] The challenger verifies that: del is valid, and that cred_{Δ} satisfies Δ_{ID} and is valid under $\mathsf{pk}_{\mathsf{Iss}}$. The verification algorithm for a VC is not defined in this work, however every VC scheme that we consider has a natural associated algorithm to verify the correct issuance of the credential.

If the adversary has access to a random oracle (according to the VP delegation scheme definition), it can send random oracle queries before sending to the challenger $\mathsf{del}, \mathsf{nonce}$ *and* cred_{Δ}. *In that case, if* $b' = 0$ *the random oracle will be simulated by* $\mathscr{S}$*, otherwise the queries are sent to a real random oracle.*

Definition 6.4.1 (Indistinguishability of the Delegated Presentation) *A VP delegation scheme satisfies* indistinguishability of the delegated presentation *if for any probabilistic polynomial-time (PPT) adversary* $\mathscr{A}$ *executing the experiment* $\mathsf{Exp}^{\mathsf{Indistinguishability}}_{\mathscr{A}}(1^{\lambda})$,

$$\left|\Pr\left[\mathscr{A} \textit{ wins } \mathsf{Exp}^{\mathsf{Indistinguishability}}_{\mathscr{A}}(1^{\lambda})\right] - \frac{1}{2}\right| = \left|\Pr[\bar{b}' = b'] - \frac{1}{2}\right| \leq \nu(\lambda),$$

where $\nu(\lambda)$ *is a negligible function in the security parameter* λ.

Theorem 6.4.5 *The delegation scheme described in Definition 6.3.2 satisfies* indistinguishability of the delegated presentation *as defined in Definition 6.4.1.*

We construct a simulator $\mathscr{S}$ that emulates the delegated presentation by programming the random oracle RO.

- **Input.** The simulator takes as input the public parameters of the system, the issuer's public key $\mathsf{pk}_{\mathsf{Iss}}$, and a pair $(\bar{A}_c, \bar{B}_c)$ that satisfies the relation $\bar{A}_c^{\mathsf{sk}_{\mathsf{Iss}}} = \bar{B}_c$ [142, 74].
- **Random Oracle queries.** At the beginning, the internal hash table HT is empty. Whenever a query is made to the random oracle RO:
  - If the input has already been queried, the simulator returns the corresponding value stored in the hash table HT;
  - If the input x is new, the simulator samples a random value $c \xleftarrow{\$} \mathbb{Z}_p$, stores (x, c) in HT, and returns x.

The simulator $\mathscr{S}$ proceeds as follows:

- Sample a random exponent $\gamma \xleftarrow{\$} \mathbb{Z}_p$.

- Compute:
$$\bar{A} \leftarrow \bar{A}_c^{\gamma}, \qquad \bar{B} \leftarrow \bar{B}_c^{\gamma}.$$

- Sample random values:
$$s, n \xleftarrow{\$} \mathbb{Z}_p, \qquad u_i \xleftarrow{\$} \mathbb{Z}_p \quad \forall i \in \mathsf{Hid}.$$

- Set the randomness vector:
$$\mathbf{z} \leftarrow (s, n, (u_i)_{i \in \mathsf{Hid}}).$$

- Sample a random challenge:
$$c \xleftarrow{\$} \mathbb{Z}_p.$$

- Compute:
$$U \leftarrow \bar{A}^s \cdot \bar{B}^n \cdot \prod_{i \in \mathsf{Hid}} \mathbf{h}[i]^{u_i} \cdot C_{\mathsf{Rev}}(\mathbf{a})^{-c}.$$

The simulator $\mathscr{S}$ then programs the random oracle RO on the tuple $(\bar{A}, \bar{B}, U, \mathsf{del}, \mathsf{nonce})$, by storing $((\bar{A}, \bar{B}, U, \mathsf{del}, \mathsf{nonce}), c)$ in HT. This ensures that any future query with the same input will consistently return the fixed challenge c.

The simulator $\mathscr{S}$ then constructs the proof and the corresponding delegated presentation as follows:

- It sets the proof π_{del} to:
$$\pi_{\mathsf{del}} \leftarrow (\bar{A}, \bar{B}, c, \mathbf{z});$$

- It assembles the final presentation pres as:
$$\mathsf{pres} \leftarrow (\mathsf{del}, \pi_{\mathsf{del}})$$
and sends it to the adversary.

Note that the simulator $\mathscr{S}$ is constructed to ensure that the output presentations are correct and distributed as real presentations.

In particular,
$$\bar{A}^{\mathsf{sk}_{\mathsf{Iss}}} = (\bar{A}_c^{\gamma})^{\mathsf{sk}_{\mathsf{Iss}}} = (\bar{A}_c^{\mathsf{sk}_{\mathsf{Iss}}})^{\gamma} = \bar{B}_c^{\gamma} = \bar{B}$$

therefore, the pairing check

$$\mathbf{e}(\bar{A}, X_2) = \mathbf{e}(\bar{B}, g_2)$$

remains satisfied after randomization, and the pair $(\bar{A}, \bar{B})$ is distributed uniformly at random as in real presentations. In addition, the first message U is chosen uniformly at random as a consequence of the sampling of the responses $\mathbf{z}$ and the challenge c

$$U = \bar{A}^s \cdot \bar{B}^n \cdot \prod_{i \in \mathsf{Hid}} \mathbf{h}[i]^{u_i} \cdot C_{\mathsf{Rev}}(\mathbf{a})^{-c}.$$

and since the simulator programs the random oracle on input $(\bar{A}, \bar{B}, U, \mathsf{del}, \mathsf{nonce})$ to c, the verification still holds. Consequently, all verification checks are satisfied, and the simulated presentation is indistinguishable from a real one. This means that, in Experiment 7, an adversary trying to distinguish a simulated presentation (obtained running $\mathscr{S}$) from a real one (obtained running DelegPres) cannot succeed with probability greater than $\frac{1}{2}$. Therefore, it satisfies the indistinguishability of delegated presentations as defined in Definition 6.4.1.

6.4.4 Collision Handling in the Random Oracle Simulation

In the proof of Theorem 6.4.5, we constructed a simulator $\mathscr{S}$ that emulates the delegated presentation by programming the random oracle RO. The correctness of this simulation relies on the ability of $\mathscr{S}$ to consistently answer all random oracle queries without overwriting previously assigned outputs in the internal hash table HT. The purpose of this section is to rigorously analyze this event and to show that the probability of such a collision occurring is negligible in the security parameter λ.

Probability of Collision The adversary $\mathscr{A}$ is allowed to make a polynomial number of queries to RO. For each new random oracle query for input x made by $\mathscr{A}$, the simulator $\mathscr{S}$ samples a random challenge $c \xleftarrow{\$} \mathbb{Z}_p$ and stores the pair (x, c) in a hash table HT. Also, during the simulation of a delegation presentation, the simulator $\mathscr{S}$ internally generates a fresh random value $c \in \mathbb{Z}_p$ and sets U accordingly so that the RO query $(\bar{A}, \bar{B}, U, \mathsf{del}, \mathsf{nonce})$ will map to c. If $\mathscr{A}$ has previously queried RO with this exact input, and the random oracle has already assigned a different value $c' \neq c$ to this input, then the simulator cannot overwrite the hash table and the simulation

fails. However, since the challenge c is chosen uniformly at random from $\mathbb{Z}_p$, and the adversary is limited to a polynomial number of RO queries, the probability of such a collision is bounded above by a negligible function in the security parameter λ.

Let $Q_1, \ldots, Q_q$ denote the q distinct queries made by $\mathscr{A}$ to RO (with q polynomial in λ). The simulator fails if, for some $j \in [q]$, the an input $Q_j = (\bar{A}_j, \bar{B}_j, U_j, \mathsf{del}_j, \mathsf{nonce}_j)$ matches the tuple $(\bar{A}', \bar{B}', U', \mathsf{del}', \mathsf{nonce}')$ constructed by the simulator in the simulation of the presentation. Since del and nonce are chosen by the adversary, while $\bar{A}', \bar{B}', U'$ are generated by the simulator, the probability that the simulator generates a tuple that matches one of the adversary's previous queries can be bounded as follows. Observe that $\bar{B}$ is computed as $\bar{B} = \bar{A}^{\mathsf{sk}_{\mathsf{Iss}}}$. Therefore, once $\bar{A}$ is fixed, $\bar{B}$ is uniquely determined. As a result, for a fixed pair $(\mathsf{del}, \mathsf{nonce})$, the simulator's internally constructed tuple

$$(\bar{A}', \bar{B}', U', \mathsf{del}, \mathsf{nonce})$$

can coincide with a query Q_j made by the adversary only if[14]

$$U = U' \quad \text{and} \quad \bar{A} = \bar{A}' \quad \Rightarrow \quad \bar{B} = \bar{B}'.$$

Since U and $\bar{A}$ are sampled independently and uniformly at random from a group of prime order p, the joint probability that $U = U'$ and $\bar{A} = \bar{A}'$ for fixed $(\mathsf{del}, \mathsf{nonce})$ is exactly $\frac{1}{p^2}$. Applying the union bound over all q queries made by the adversary to the random oracle:

$$\Pr[\text{Simulator fails}] \leq \sum_{j=1}^{q} \Pr[(\bar{A}, \bar{B}, U, \mathsf{del}, \mathsf{nonce}) = Q_j] \leq \frac{q}{p^2}.$$

Since $q = \text{poly}(\lambda)$ and p is exponential in the security parameter λ, the probability that the Simulator fails is negligible in λ.

Theorem 6.4.6 *The delegation scheme described in Definition 6.3.2 satisfies unlinkability*[15] *of the delegated presentation as defined in Definition 6.2.5.*

[14] Note that this is a necessary condition for a collision between the tuples.

[15] In the delegation scheme described in Definition 6.3.1, unlinkability does not hold for delegations and delegated presentations. Specifically, the delegator's and delegate's presentations are structured as

$$\mathsf{pres}' \leftarrow \big((\sigma_{\mathsf{D}}, \{\mathsf{com}_{\mathsf{D},i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}}), \{\mathsf{salt}_{\mathsf{D},i}\}_{i\in\mathsf{DP}}\big), \qquad \mathsf{pres}'' \leftarrow \big((\sigma_{\Delta}, \{\mathsf{com}_{\Delta,i}\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\Delta}}), \{\mathsf{salt}_{\Delta,i}\}_{i\in\Delta_{\mathsf{ID}}}, \mathsf{nonce}\big).$$

We prove the theorem via a reduction. Specifically, we assume that there exists a probabilistic polynomial-time adversary $\mathscr{A}$ that wins the unlinkability experiment $\mathsf{Exp}^{\mathsf{DelegPresUnlinkability}}_{\mathscr{A}}(1^\lambda)$ with non-negligible probability. We then construct a reduction $\mathscr{R}$ that uses $\mathscr{A}$ to win the indistinguishability experiment $\mathsf{Exp}^{\mathsf{Indistinguishability}}_{\mathscr{R}}(1^\lambda)$ with non-negligible probability. This would contradict the result proven in Theorem 6.4.5.

Consider the probabilistic polynomial-time simulator $\mathscr{S}$ introduced in the proof of Theorem 6.4.5, which, given as input del and nonce, outputs a simulated delegated presentation pres^* that is computationally indistinguishable from a real delegated presentation pres generated by an honest delegatee using a valid anonymous credential $\mathsf{anoncred}_\Delta$, i.e.,

$$\mathscr{S}(\mathsf{del},\mathsf{nonce}) \to \mathsf{pres}^* \approx_c \mathsf{DelegPres}(\mathsf{del},\mathsf{cred}_\Delta,\mathsf{nonce}).$$

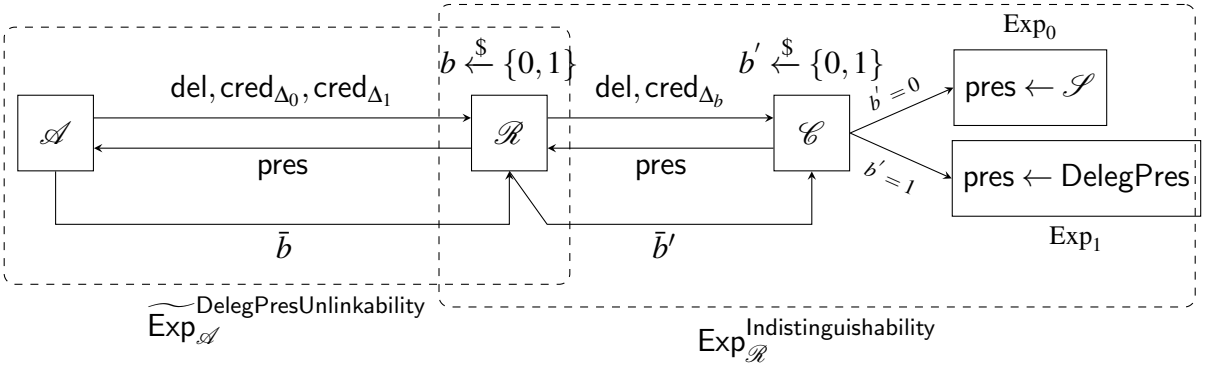

Fig. 6.2 Overview of the interaction between the adversary $\mathscr{A}$, the challenger $\mathscr{C}$, and the reduction $\mathscr{R}$. The reduction $\mathscr{R}$ first chooses a bit b to select one of the two credentials provided by $\mathscr{A}$. It then receives either a simulated presentation (if $b' = 0$) or a real presentation (if $b' = 1$) corresponding to the chosen credential. The adversary $\mathscr{A}$ tries to guess the bit b based on the presentation received. Based on $\mathscr{A}$'s guess $\bar{b}$, the reduction $\mathscr{R}$ outputs its own guess $\bar{b}'$ for b': if $\bar{b} = b$, $\mathscr{R}$ guesses that the presentation is real ($b' = 1$); otherwise, it guesses simulated ($b' = 0$).

Assume, by contradiction, that there exists an adversary $\mathscr{A}$ that wins $\mathsf{Exp}^{\mathsf{DelegPresUnlinkability}}_{\mathscr{A}}(1^\lambda)$ with probability $\frac{1}{2} + \varepsilon$, where ε is non-negligible in the security parameter λ. Recall from Experiment 6 that this corresponds to the setting where the presentation is always generated from a real credential.

Since pres' and pres'' include $\mathsf{pk}_{\mathsf{cred}_\mathsf{D}}$ and $\mathsf{pk}_{\mathsf{cred}_\Delta}$, respectively, multiple presentations by the same party are trivially linkable.

We now introduce a variant of the Experiment 6, denoted by $\widetilde{\mathsf{Exp}}^{\mathsf{DelegPresUnlinkability}}_{\mathscr{A}}$, in which the presentation pres may be either real or simulated depending on a random bit b', as illustrated in Figure 6.2. The formal description of this variant is given in the following experiment.

Experiment 8 ($\widetilde{\mathsf{Exp}}^{\mathsf{DelegPresUnlinkability}}_{\mathscr{A}}(1^{\lambda})$)

1. *The adversary $\mathscr{A}$ generates the public parameters* pp *for the verifiable credential scheme together with the issuer key pair* $(\mathsf{sk}_{\mathsf{Iss}}, \mathsf{pk}_{\mathsf{Iss}})$ *and send* pp *and* $\mathsf{pk}_{\mathsf{Iss}}$ *to the challenger.*
2. *The challenger samples a random bit* $b' \xleftarrow{\$} \{0,1\}$.
3. *$\mathscr{A}$ generates and sends to the challenger* del, nonce *and* cred_{Δ_0} *and* cred_{Δ_1}, *satisfying* Δ_{ID}.
4. *$\mathscr{C}$ samples a random bit* $b' \xleftarrow{\$} \{0,1\}$ *and computes* pres *as follows:*
   - *If* $b' = 0$, *computes* $\mathsf{pres} \xleftarrow{\$} \mathscr{S}(\mathsf{del}, \mathsf{nonce})$.
   - *If* $b' = 1$, *samples a random bit b and computes* $\mathsf{pres} \xleftarrow{\$} \mathsf{DelegPres}(\mathsf{del}, \mathsf{cred}_{\Delta_b}, \mathsf{nonce})$.
5. *$\mathscr{C}$ sends* pres *to $\mathscr{A}$.*
6. *$\mathscr{A}$ outputs a bit* $\bar{b}$.
7. *$\mathscr{A}$ wins if* $\bar{b} = b$.

If the adversary has access to a random oracle (according to the VP delegation scheme definition), it can send random oracle queries before sending to the challenger $\mathsf{del}, \mathsf{nonce}$, cred_{Δ_0} and cred_{Δ_1}.

Next, we proceed to compute the probability that the adversary $\mathscr{A}$ wins $\widetilde{\mathsf{Exp}}^{\mathsf{DelegPresUnlinkability}}_{\mathscr{A}}$:

$$\begin{aligned}
\Pr\left[\mathscr{A} \text{ wins } \widetilde{\mathsf{Exp}}_{\mathscr{A}}^{\mathsf{DelegPresUnlinkability}}\right] &= \Pr\left[\bar{b} = b\right] = \\
&= \Pr\left[\bar{b} = b \mid b' = 0\right] \cdot \Pr\left[b' = 0\right] \\
&\quad + \Pr\left[\bar{b} = b \mid b' = 1\right] \cdot \Pr\left[b' = 1\right] = \\
&= \Pr\left[\bar{b} = b \mid b' = 0\right] \cdot \tfrac{1}{2} \\
&\quad + \Pr\left[\mathscr{A} \text{ wins } \mathsf{Exp}_{\mathscr{A}}^{\mathsf{DelegPresUnlinkability}}\right] \cdot \tfrac{1}{2} = \\
&= \tfrac{1}{2} \cdot \tfrac{1}{2} + \left(\tfrac{1}{2} + \varepsilon\right) \cdot \tfrac{1}{2} \\
&= \tfrac{1}{2} + \tfrac{\varepsilon}{2}.
\end{aligned}$$

This reflects the fact that adversary $\mathscr{A}$ has an advantage ε only when $b' = 1$, since otherwise it guesses at random.

Next, consider the probability that the bit $b' = 1$ conditioned on the event that $\mathscr{A}$ guesses correctly, i.e., $\bar{b} = b$. By Bayes' theorem, we have

$$\begin{aligned}
\Pr\left[b' = 1 \mid \bar{b} = b\right] &= \frac{\Pr\left[\bar{b} = b \mid b' = 1\right] \Pr[b' = 1]}{\Pr\left[\bar{b} = b\right]} = \\
&= \frac{\left(\frac{1}{2} + \varepsilon\right) \cdot \frac{1}{2}}{\frac{1}{2} + \frac{\varepsilon}{2}} = \frac{\frac{1}{2} + \varepsilon}{1 + \varepsilon} = \frac{1}{2} \cdot \frac{1 + 2\varepsilon}{1 + \varepsilon} = \frac{1}{2}\left(1 + \frac{\varepsilon}{1 + \varepsilon}\right) = \frac{1}{2} + \frac{1}{2} \cdot \frac{\varepsilon}{1 + \varepsilon}.
\end{aligned}$$

Since ε is non-negligible and positive, and since

$$\frac{\varepsilon}{1 + \varepsilon} > \frac{\varepsilon}{2} \qquad \forall\ 0 < \varepsilon \leq \frac{1}{2},$$

we obtain that

$$\Pr\left[b' = 1 \mid \bar{b} = b\right] > \frac{1}{2} + \frac{\varepsilon}{4},$$

which is strictly greater than $\frac{1}{2}$ by a non-negligible quantity.

Consequently, $\mathscr{R}$ can define its own guess for b' as

$$\bar{b'} = \begin{cases} 1 & \text{if } \bar{b} = b, \\ 0 & \text{if } \bar{b} \neq b. \end{cases}$$

Finally, we compute the probability that $\mathscr{R}$ wins $\mathsf{Exp}^{\mathsf{Indistinguishability}}_{\mathscr{R}}$:

$$\begin{aligned}
\Pr\left[\mathscr{R} \text{ wins } \mathsf{Exp}^{\mathsf{Indistinguishability}}_{\mathscr{R}}\right] &= \Pr\left[\bar{b'} = b'\right] = \\
&= \Pr\left[\bar{b'} = 0 \mid b' = 0\right] \cdot \Pr\left[b' = 0\right] \\
&\quad + \Pr\left[\bar{b'} = 1 \mid b' = 1\right] \cdot \Pr\left[b' = 1\right] = \\
&= \Pr\left[\bar{b} \neq b \mid b' = 0\right] \cdot \tfrac{1}{2} \\
&\quad + \Pr\left[\bar{b} = b \mid b' = 1\right] \cdot \tfrac{1}{2} = \\
&= \Pr\left[\bar{b} \neq b \mid b' = 0\right] \cdot \tfrac{1}{2} \\
&\quad + \Pr\left[\mathscr{A} \text{ wins } \mathsf{Exp}^{\mathsf{DelegPresUnlinkability}}_{\mathscr{A}}\right] \cdot \tfrac{1}{2} \\
&= \tfrac{1}{2} \cdot \tfrac{1}{2} + \left(\tfrac{1}{2} + \varepsilon\right) \cdot \tfrac{1}{2} = \\
&= \tfrac{1}{2} + \tfrac{\varepsilon}{2}.
\end{aligned}$$

Therefore, any non-negligible advantage ε that $\mathscr{A}$ has in winning $\mathsf{Exp}^{\mathsf{DelegPresUnlinkability}}_{\mathscr{A}}$ implies an advantage of $\frac{\varepsilon}{2}$ for an adversary $\mathscr{R}$ in winning $\mathsf{Exp}^{\mathsf{Indistinguishability}}_{\mathscr{R}}$. This contradicts the fact that, as proven in Theorem 6.4.5, the simulator $\mathscr{S}$ produces presentations that are computationally indistinguishable from real ones. We therefore conclude that our VP delegation scheme achieves *unlinkability* of delegated presentations.

Analogously, the unlinkability of the delegation issuance phase can be established using the same proof strategy. This is because the underlying zero-knowledge protocol used during delegation issuance is structurally identical to the one employed in delegated presentations, allowing for a simulator to generate indistinguishable transcripts.

6.5 Instantiation of Our VP Delegation Scheme for mdoc VCs in the EUDI and EBSI Frameworks

In this section we focus on the VP delegation scheme described in Definition 6.3.1, since mdoc VCs are the credentials supported by the EUDI ARF at the time of writing. The VP delegation scheme for mdoc VCs that we have described and analyzed can be integrated into existing ecosystems such as the EUDI or EBSI frameworks, without defining new data structures, only new verification procedures. We sketch some considerations describing possible solutions for that. The delegation del can be a *VC issued by the delegator* that has as attributes the components we have described, namely $\mathsf{scope}, \Delta_{\mathsf{ID}}, \mathsf{DP}$ and π_{DP}. The delegator signs this credential (as an issuer) using the same secret key used to generate the π_{DP} as specified in Definition 6.3.1. This additional signature on the whole delegation is needed for compatibility reasons, because every VC must be signed by the issuer, in this case the delegator. This special credential must also contain the $\mathsf{pk}_{\mathsf{cred}}$ of the credential that the delegatee will use to present the delegation. In this sense, the delegatee must choose in advance the single-use credential it will use to generate the presentation. When the delegatee creates a delegated presentation, it presents del (which is structured as a VC) disclosing all its attributes, namely, $\mathsf{scope}, \Delta_{\mathsf{ID}}, \mathsf{DP}$ and π_{DP}; the delegatee then creates a verifiable presentation π_{del} using its own VC, associated to $\mathsf{pk}_{\mathsf{cred}}$ communicated to the delegator, revealing the attributes included in Δ_{ID}. The outcome of the delegated presentation is derived from the combination of these two presentations. The only modification to the verification protocol is that the verifier must check that π_{DP} is indeed a valid presentation of the statement DP and that the presentation π_{del} created by the delegatee is a valid presentation of Δ_{ID}.

In EBSI, the only entities entitled to issue credentials are legal persons whose DID (a reference to, at least, their public key) is registered in the Trusted Issuer Registry (TIR)[16]. This means that credential holders who are physical persons are not allowed to issue credentials and therefore cannot generate the delegation as we have mentioned above. If the delegator is only a physical person we must consider two cases:

- the delegatee is a legal person: in this case, the delegator can generate the attributes for the delegation VC $\mathsf{scope}, \Delta_{\mathsf{ID}}, \mathsf{DP}$ and π_{DP} and sends them to the

[16] https://hub.ebsi.eu/apis/pilot/trusted-issuers-registry/v4

delegatee who generates the VC containing this information and signs it in turn. Note that this signature is only useful to guarantee the compatibility with the credential format and to create a VC issued by an entity registered in the TIR. The delegatee can choose not to sign it, which is perfectly fine, but cannot change the information sent by the delegator because π_{DP} is cryptographically bound to $\mathsf{scope}, \Delta_{\mathsf{ID}}$ and DP.

- if the delegatee is also only a physical person, the delegator must have the delegation VC signed by a third party registered in the TIR, which may be the issuer of the credential used to generate the delegation or a third party with this specific role.

6.6 Conclusions

In this work, we introduced the notion of verifiable presentation delegation scheme, enabling a credential holder to securely authorize a third party to present credentials on their behalf. We formalized the definition of a VP delegation scheme, defining its core algorithms and essential security properties, including correctness, unforgeability, and unlinkability. We proposed two concrete constructions: one compatible with verifiable credentials in the EUDI ARF, and another supporting anonymous credentials based on BBS signatures, and proved their security. Finally, we discussed the practical applicability of our schemes, highlighting their integration within real-world ecosystems such as EBSI and EUDI frameworks.

Chapter 7

Conclusions

This work has investigated fundamental problems at the intersection of cryptography, decentralized systems, and digital identity. The three contributions presented in this thesis address complementary challenges, ranging from swap protocols across different blockchains, to the design and implementation of post-quantum anonymous credential system, and the delegation of digital credential presentations.

The first contribution of this thesis lies in the design of *BTLE (Broadcast Time-Lock Exchange Protocol)*, a novel protocol for decentralized cross-chain exchange. BTLE departs from classical approaches based on Hash Time-Locked Contracts (HTLCs) by leveraging time-lock puzzles as a synchronization primitive, enabling exchanges in scenarios where scripting capabilities are limited or incompatible across different blockchains. The protocol is structured in two phases: a matching phase (BTLE-MA), which enables competition among multiple market takers, and a swap phase (BTLE-AS), which ensures a fair exchange between the selected parties. A key technical contribution of this work is the introduction of a new class of time-lock puzzles based on Pell conic computations, proposed as an alternative to the classical construction by Rivest, Shamir, and Wagner. Experimental results show that the proposed time-lock puzzle based on the Pell conic outperforms the classical construction for the intended application. In particular, for a modulus of 2000 bits, the RSW-TL scheme solves the puzzle in approximately 100 seconds, whereas the proposed BM-TL construction requires over 4000 seconds. This substantial increase in delay is achieved while significantly reducing the number of squaring operations. In addition, the protocol is fully implemented and evaluated, showing that BTLE can

effectively operate off-chain and overcome important limitations of HTLC-based designs.

The second contribution focuses on the design and implementation of *Post-Quantum Anonymous Verifiable Credential* systems within the Self-Sovereign Identity (SSI) paradigm. While anonymous credential schemes based on classical assumptions (e.g., CL or BBS+) are well understood and practically efficient, they are vulnerable to quantum computers. This motivates the transition toward post-quantum constructions. In this context, the work provides a comprehensive analysis of existing post-quantum anonymous credential frameworks, identifying practical limitations and trade-offs. Building on this analysis, a concrete framework based on recent lattice-based constructions is selected, adapted to the verifiable credential data model, and implemented in an open-source library. The resulting system achieves 128-bit security against quantum adversaries and supports selective disclosure of attributes.

Finally the third contribution of this thesis focuses on the delegation of verifiable presentations of digital credentials. While delegation has been extensively studied in the context of credential issuance, the delegation of *verifiable presentations* has not been formally addressed in the literature. This work fills this gap by introducing the notion of a *verifiable presentation delegation scheme*, providing a formal definition of its syntax and identifying the fundamental security properties it must satisfy, namely correctness, unforgeability, and unlinkability. Two concrete instantiations of are presented: the first compatible with standard verifiable credentials, including those defined in the European Digital Identity Wallet Architecture Reference Framework (EUDI ARF), while the second supports BBS anonymous credentials. Both schemes are formally proven secure according to the proposed definitions, and their practical applicability is discussed in the context of infrastructures such as EBSI and the EUDI framework.

Several directions for future work emerge naturally from this research. On the side of decentralized exchange, further work is required to analyze the security of the proposed scheme under stronger adversarial models and to investigate its integration with emerging cross-chain interoperability protocols. In the context of post-quantum credentials, improving the efficiency and scalability of zero-knowledge proofs remains a key challenge. Finally, the notion of delegated presentations can be extended to more complex scenarios, such as hierarchical delegations, revocation mechanisms, and composability with other cryptographic protocols.

References

[1] Politecnico di Milano. Blockchain & Web3 Observatory. https://www.osservatori.net/en/research/active-observatories/blockchain-web3.

[2] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[3] Victor S Miller. Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques*, pages 417–426. Springer, 1986.

[4] Jorge Guajardo and Christof Paar. Efficient algorithms for elliptic curve cryptosystems. In *Annual International Cryptology Conference*, pages 342–356. Springer, 1997.

[5] Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. Ripemd-160: A strengthened version of ripemd. *Lecture Notes in Computer Science*, 02 2000.

[6] Wouter Penard and Tim van Werkhoven. On the secure hash algorithm family. 2008.

[7] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Keccak. Cryptology ePrint Archive, Paper 2015/389, 2015.

[8] H. W. Lenstra and Carl Pomerance. A rigorous time bound for factoring integers. *Journal of the American Mathematical Society*, 5(3):483–516, 1992.

[9] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Alessandro Sorniotti. A Framework for Practical Anonymous Credentials from Lattices. Cryptology ePrint Archive, Paper 2023/560, 2023. https://eprint.iacr.org/2023/560.

[10] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Alessandro Sorniotti. A Framework for Practical Anonymous Credentials from Lattices. In *CRYPTO 2023*, pages 384–417, Santa Barbara (CA), August 2023.

[11] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plancon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. Cryptology ePrint Archive, Paper 2022/284, 2022.

[12] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography*. 2023. https://toc.cryptobook.us/.

[13] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, 2008. Available at http://www.cs.stanford.edu/~xb/joc07/.

[14] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. Cryptology ePrint Archive, Paper 2008/136, 2008.

[15] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55, 08 2004. https://doi.org/10.1007/978-3-540-28628-8_3.

[16] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. volume 3152/2004, pages 56–72, 08 2004.

[17] Stefano Tessaro and Chenzhi Zhu. Revisiting bbs signatures. In *Advances in Cryptology – EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, page 691–721, Berlin, Heidelberg, 2023. Springer-Verlag.

[18] Paulo SLM Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In *Security in Communication Networks: Third International Conference, SCN 2002 Amalfi, Italy, September 11–13, 2002 Revised Papers 3*, pages 257–267. Springer, 2003.

[19] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2014.

[20] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1992.

[21] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1987.

[22] Dan Boneh and Victor Shoup. A Graduate Course in Applied Cryptography, 2023. Last accessed: 2025-10-10.

[23] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '89, page 239–252, Berlin, Heidelberg, 1989. Springer-Verlag.

[24] Jan Camenisch and Anna Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. volume 2045/2001, pages 93–118, 05 2001.

[25] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.

[26] Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release Crypto. Technical report, 1996.

[27] Ronald Rivest, Adi Shamir, and David Wagner. Time-lock puzzles and timed-release crypto. 03 2001.

[28] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009.

[29] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

[30] Ikuya Takashima. *Ripple: The Ultimate Guide to the World of Ripple XRP, Ripple Investing, Ripple Coin, Ripple Cryptocurrency, Cryptocurrency*. CreateSpace Independent Publishing Platform, North Charleston, SC, USA, 2018.

[31] Bin Cao, Xuesong Wang, Weizheng Zhang, Houbing Song, and Zhihan Lyu. A many-objective optimization model of industrial internet of things based on private blockchain. *IEEE Network*, 34:78–83, 09 2020.

[32] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. Hyperledger fabric. In *Proceedings of the Thirteenth EuroSys Conference*. ACM, apr 2018.

[33] Richard Brown, James Carlyle, Ian Grigg, and Mike Hearn. Corda: An introduction, 09 2016.

[34] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, page 173–186, USA, 1999. USENIX Association.

[35] Karl Wüst and Arthur Gervais. Do you need a blockchain? In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 45–54, 2018.

[36] Paula Fraga-Lamas and Tiago M. Fernández-Caramés. A review on blockchain technologies for an advanced and cyber-resilient automotive industry. *IEEE Access*, 7:17578–17598, 2019.

[37] Mohammad Chowdhury, Alan Colman, Ashad Kabir, Jun Han, and Paul Sarda. Blockchain versus database: A critical analysis. pages 1348–1353, 08 2018.

[38] Sin Kuang Lo, Xiwei Xu, Yin Kia Chiam, and Qinghua Lu. Evaluating suitability of applying blockchain. pages 158–161, 11 2017.

[39] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Annual international cryptology conference*, pages 139–147. Springer, 1992.

[40] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In *Secure information networks*, pages 258–272. Springer, 1999.

[41] Adam Back et al. Hashcash-a denial of service counter-measure. 2002.

[42] Poonam Rani and Rajul Bhambay. A Comparative Survey of Consensus Algorithms Based on Proof of Work. In *Emerging Technologies in Data Mining and Information Security*, pages 261–268. Springer, 2023.

[43] Alexei Zamyatin, Mustafa Al-Bassam, Dionysis Zindros, Eleftherios Kokoris-Kogias, Pedro Moreno-Sanchez, Aggelos Kiayias, and William J Knottenbelt. SoK: Communication Across Distributed Ledgers. *ePrint IACR*, page 17, 2018.

[44] Vitalik Buterin. Chain interoperability. *R3 Research Paper*, 2016.

[45] Kaihua Qin, Liyi Zhou, Yaroslav Afonin, Ludovico Lazzaretti, and Arthur Gervais. Cefi vs. defi - comparing centralized to decentralized finance. *CoRR*, abs/2106.08157, 2021.

[46] N. Asokan, Victor Shoup, and Michael Waidner. Asynchronous protocols for optimistic fair exchange. In *Security and Privacy - 1998 IEEE Symposium on Security and Privacy, Oakland, CA, USA, May 3-6, 1998, Proceedings*, pages 86–99. IEEE Computer Society, 1998.

[47] T. Nolan. Alt chains and atomic transfers. https://archive.ph/wWzna, 2013.

[48] Maurice Herlihy. Atomic Cross-Chain Swaps. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 245–254, Egham United Kingdom, July 2018. ACM.

[49] Ujan Mukhopadhyay, Anthony Skjellum, Oluwakemi Hambolu, Jon Oakley, Lu Yu, and R. Brooks. A brief survey of cryptocurrency systems. pages 745–752, 12 2016.

[50] Gamze Yıldız Şeren, Dilek Akbaş Akdoğan, and Osman Geyik. *Cryptocurrencies and Blockchain In 4th Industrial Revolution Process: Some Public Recommendations*. 10 2019.

[51] Amendments by the European Parliament to the Commission proposal for a Regulation of the European Parliament and of the Council amending Regulation (EU) no 910/2014 as regards establishing a framework for a European Digital Identity. https://www.europarl.europa.eu/doceo/document/A-9-2023-0038_EN.html, 03 2023.

[52] Consolidated text: Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 (General Data Protection Regulation) (Text with EEA relevance). http://data.europa.eu/eli/reg/2016/679/2016-05-04, 2016.

[53] The European Digital Identity Wallet Architecture and Reference Framework, version 2.6.0, 10 2025.

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

[55] Carsten Baum, Olivier Blazy, Jan Camenisch, Jaap-Henk Hoepman, Eysa Lee, Anja Lehmann, Anna Lysyanskaya, René Mayrhofer, Hart Montgomery, Ngoc Khanh Nguyen, et al. Cryptographers' Feedback on the EU Digital Identity's ARF. *Tech. Rep.*, 2024.

[56] Mobile Driver's License (mDL) Implementation Guidelines, Version 1.2. https://www.aamva.org/topics/mobile-driver-license, 01 2023.

[57] ISO/IEC 18013-5 Personal identification - ISO-compliant driving licence - part 5: Mobile driving licence (mDL) application, 09 2021.

[58] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *SCN 2002*, volume 2576 of *LNCS*, pages 268–289, 2002.

[59] Andrea Flamini, Giada Sciarretta, Mario Scuro, Amir Sharif, Alessandro Tomasi, and Silvio Ranise. On cryptographic mechanisms for the selective disclosure of verifiable credentials. *Journal of Information Security and Applications*, 83:103789, 2024.

[60] Alex Preukschat and Drummond Reed. *Self-Sovereign Identity: Decentralized digital identity and verifiable credentials*. Manning, NY, 2021.

[61] W3C. 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/.

[62] Jan Camenisch and Anna Lysyanskaya. A Signature Scheme with Efficient Protocols. In *International Conference on Security in Communication Networks*, pages 268–289, Amalfi (IT), September 2002.

[63] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous Attestation Using the Strong Diffie-Hellman Assumption Revisited. In *International Conference on Trust and Trustworthy Computing*, pages 1–20, Vienna (AT), August 2016.

[64] David Chaum. Security without identification: transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, October 1985.

[65] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. volume 2045/2001, pages 93–118, 05 2001.

[66] Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, page 21–30, New York, NY, USA, 2002. Association for Computing Machinery.

[67] Jan Camenisch and Thomas Groß. Efficient attributes for anonymous credentials. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, page 345–356, New York, NY, USA, 2008. Association for Computing Machinery.

[68] Harsha Gardiyawasam Pussewalage and Vladimir Oleshchuk. An anonymous delegatable attribute-based credential scheme for a collaborative e-health environment. *ACM Transactions on Internet Technology*, 19:1–22, 09 2019.

[69] Chao Lin, De biao He, Hongjie Zhang, Lisong Shao, and Xinyi Huang. Privacy-enhancing decentralized anonymous credential in smart grids. *Comput. Stand. Interfaces*, 75:103505, 2021.

[70] Tomasz Truderung. Enhanced anonymous credentials for e-voting systems, 2025.

[71] P. Angin, Bharat Bhargava, Rohit Ranchal, Noopur Singh, Mark Linderman, Lotfi ben Othmane, and Leszek Lilien. An entity-centric approach for privacy and identity management in cloud computing. pages 177 – 183, 12 2010.

[72] Christian Paquin and Greg Zaverucha. U-prove cryptographic specification v1.1. Technical report, Microsoft Corporation, 2011.

[73] IBM. Idemix. https://github.com/IBM/idemix, 2022.

[74] Stefano Tessaro and Chenzhi Zhu. Revisiting BBS signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 691–721. Springer, 2023.

[75] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In *International conference on the theory and applications of cryptographic techniques*, pages 418–433. Springer, 2002.

[76] Jorn Lapon, Markulf Kohlweiss, Bart De Decker, and Vincent Naessens. Analysis of revocation strategies for anonymous idemix credentials. pages 3–17, 10 2011.

[77] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. Blacklistable anonymous credentials: blocking misbehaving users without ttps. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, page 72–81, New York, NY, USA, 2007. Association for Computing Machinery.

[78] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. Blac: Revoking repeatedly misbehaving anonymous users without relying on ttps. *ACM Trans. Inf. Syst. Secur.*, 13(4), December 2010.

[79] Stefan Brands, Liesje Demuynck, and Bart De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. volume 4586, pages 400–415, 07 2007.

[80] Toru Nakanishi, Hiroki Fujii, Yuta Hira, and Nobuo Funabiki. Revocable group signature schemes with constant costs for signing and verifying. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 93-A:50–62, 2009.

[81] Jorn Lapon, Markulf Kohlweiss, Bart De Decker, and Vincent Naessens. Analysis of revocation strategies for anonymous idemix credentials. pages 3–17, 10 2011.

[82] Jan Camenisch and Anna Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In Moti Yung, editor, *CRYPTO 2002*, pages 61–76, Berlin, Heidelberg, 2002. Springer.

[83] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In Stanisław Jarecki and Gene Tsudik, editors, *Public Key Cryptography*, pages 481–500, Berlin, Heidelberg, 2009. Springer.

[84] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. Solving Revocation with Efficient Update of Anonymous Credentials. In Juan A. Garay and Roberto De Prisco, editors, *Security and Cryptography for Networks*, pages 454–471, Berlin, Heidelberg, 2010. Springer.

[85] Fadi Barbara, Enrico Guglielmino, Nadir Murru, and Claudio Schifanella. Btle: Atomic swaps with time-lock puzzles. *Journal of Mathematical Cryptology*, 19(1):20240044, 2025.

[86] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 281–310. Springer, 2015.

[87] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California,*

USA, August 16-20, 1992, Proceedings, volume 740 of *Lecture Notes in Computer Science*, pages 139–147. Springer, 1992.

[88] Adam Back. Hashcash - A Denial of Service Counter-Measure. page 10, 1992.

[89] Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. *J. Cryptol.*, 3(2):99–111, 1991.

[90] Andrew Poelstra. On Stake and Consensus. Technical report, March 2015.

[91] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling Blockchain Innovations with Pegged Sidechains. *Whitepaper*, page 25, 2014.

[92] Johnny Dilley, Andrew Poelstra, Jonathan Wilkins, Marta Piekarska, Ben Gorlick, and Mark Friedenbach. Strong Federations: An Interoperable Blockchain Solution to Centralized Third-Party Risks. *arXiv:1612.05491 [cs]*, January 2017.

[93] Joseph Poon. Plasma : Scalable autonomous smart contracts. 2017.

[94] Fadi Barbara, Nadir Murru, and Claudio Schifanella. Towards a broadcast time-lock based token exchange protocol. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 13098, pages 243–254. Springer, 2022.

[95] Jae Kwon and Ethan Buchman. Cosmos Whitepaper. https://cosmos.network/cosmos-whitepaper.pdf, 2016.

[96] Dr Gavin Wood. POLKADOT: VISION FOR A HETEROGENEOUS MULTI-CHAIN FRAMEWORK. *Whitepaper*, 2016.

[97] Samuel Jaques, Hart Montgomery, and Arnab Roy. Time-release Cryptography from Minimal Circuit Assumptions. Technical Report 755, 2020.

[98] Mihir Bellare and Shafi Goldwasser. Encapsulated Key Escrow. Technical report, Massachusetts Institute of Technology, 1996.

[99] Dan Boneh and Moni Naor. Timed Commitments. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Mihir Bellare, editors, *Advances in Cryptology — CRYPTO 2000*, volume 1880, pages 236–254. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[100] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable Delay Functions. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, volume 10991, pages 757–788. Springer International Publishing, Cham, 2018.

[101] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. *Whitepaper*, page 9, 2008.

[102] Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. Homomorphic Time-Lock Puzzles and Applications. In *Advances in Cryptology – CRYPTO 2019*, volume 11692, pages 620–649. Springer International Publishing, Cham, 2019.

[103] Benjamin Wesolowski. Efficient verifiable delay functions. *J. Cryptol.*, 33(4):2113–2147, 2020.

[104] Krzysztof Pietrzak. Simple verifiable delay functions. In *10th innovations in theoretical computer science conference (itcs 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[105] Dan Boneh, Benedikt Bünz, and Ben Fisch. A Survey of Two Verifiable Delay Functions. Technical Report 712, 2018.

[106] Jia Liu, Tibor Jager, Saqib A. Kakvi, and Bogdan Warinschi. How to build time-lock encryption. *Des. Codes Cryptogr.*, 86(11):2549–2586, 2018.

[107] M. H. Miraz and D. C. Donald. Atomic Cross-chain Swaps: Development, Trajectory and Potential of Non-monetary Digital Token Swap Facilities. *Annals of Emerging Technologies in Computing*, 3(1):42–50, January 2019.

[108] Yacov Manevich and Adi Akavia. Cross chain atomic swaps in the absence of time via attribute verifiable timed commitments. In *IEEE 7th European Symposium on Security and Privacy*, pages 606–626, 2022.

[109] L. Lys, A. Micoulet, and M. Potop-Butucaru. R-swap: Relay based atomic cross-chain swap protocol. In *Proc. 6th Int. Symp. Algorithmic Aspects Cloud Comput.*, pages 18–37, 2021.

[110] Emanuele Bellini and Nadir Murru. An efficient and secure RSA-like cryptosystem exploiting Rédei rational functions over conics. *Finite Fields and Their Applications*, 39:179–194, May 2016.

[111] Emanuele Bellini, Nadir Murru, Antonio J Di Scala, and Michele Elia. Group law on affine conics and applications to cryptography. page 15.

[112] Isra Mohamed Ali, Maurantonio Caprolu, and Roberto Di Pietro. Foundations, Properties, and Security Applications of Puzzles: A Survey. *arXiv:1904.10164 [cs]*, April 2020.

[113] Gavin Andresen. Bip-16: Pay to script hash. Github, 2013.

[114] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. Information Security and Cryptography. Springer, 2010.

[115] Tiago M Fernandez-Carames and Paula Fraga-Lamas. Towards post-quantum blockchain: A review on blockchain cryptography resistant to quantum computing attacks. *IEEE access*, 8:21091–21116, 2020.

[116] Davide Margaria, Alessandro Pino, Andrea Vesco, Giuseppe D'Alconzo, Antonio J. Di Scala, Enrico Guglielmino, and Carlo Sanna. Implementation of a post-quantum anonymous verifiable credential framework. In *2025 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6, 2025.

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

[118] Corentin Jeudy, Adeline Roux-Langlois, and Olivier Sanders. Lattice Signature with Efficient Protocols, Application to Anonymous Credentials. Cryptology ePrint Archive, Paper 2022/509, 2022. https://eprint.iacr.org/2022/509.

[119] Qiqi Lai, Chongshen Chen, Feng-Hao Liu, Anna Lysyanskaya, and Zhedong Wang. Lattice-based Commit-Transferrable Signatures and Applications to Anonymous Credentials. Cryptology ePrint Archive, Paper 2023/766, 2023. https://eprint.iacr.org/2023/766.

[120] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Signature Schemes with Efficient Protocols and Dynamic Group franklinfrom Lattice Assumptions. In *International Conference on the Theory and Applications of Cryptology and Information Security*, pages 373–403, Hanoi (VN), December 2016.

[121] Vadim Lyubashevsky, Gregor Seiler, and Patrick Steuer. The LaZer Library: Lattice-Based Zero Knowledge and Succinct Proofs for Quantum-Safe Privacy. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 3125–3137, 2024.

[122] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022*, pages 71–101, Cham, 2022. Springer Nature Switzerland.

[123] Miklós Ajtai. Generating Hard Instances of Lattice Problems. *Electronic Colloquium on Computational Complexity*, 96(7), January 1996. https://eccc.weizmann.ac.il/report/1996/007/download.

[124] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for Hard Lattices and New Cryptographic Constructions. Cryptology ePrint Archive, Paper 2007/432, 2007. https://eprint.iacr.org/2007/432.

[125] Martin R. Albrecht, Valerio Cini, Russell W. F. Lai, Giulio Malavolta, and Sri AravindaKrishnan Thyagarajan. Lattice-Based SNARKs: Publicly Verifiable, Preprocessing, and Recursively Composable. Cryptology ePrint Archive, Paper 2022/941, 2022. https://eprint.iacr.org/2022/941.

[126] Léo Ducas et al. CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme. *IACR Transactions on Cryptographic Hardware and Embedded*

Systems, 2018(1):238–268, February 2018. https://tches.iacr.org/index.php/TCHES/article/view/839.

[127] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe P. Buhler, editor, *Algorithmic Number Theory*, pages 267–288, Berlin, Heidelberg, 1998. Springer.

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

[129] Adeline Langlois and Damien Stehlé. Worst-case to Average-case Reductions for Module Lattices. *Designs, Codes and Cryptography*, 75(3):565–599, June 2015.

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

[131] Victor Shoup. NTL: A Library for doing Number Theory. https://libntl.org.

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

[133] Victor Shoup. NTL vs FLINT, 2021. https://libntl.org/benchmarks.pdf.

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

[135] Andrea Flamini, Andrea Gangemi, Enrico Guglielmino, and Vincenzo Orabona. On Delegation of Verifiable Presentations. In *Proceedings of the 3rd International Workshop on Trends in Digital Identity (TDI 2025)*, pages 26–38, 2025.

[136] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *Advances in Cryptology-CRYPTO 2006: 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006. Proceedings 26*, pages 78–96. Springer, 2006.

[137] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In *Advances in Cryptology-CRYPTO 2009: 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 108–125. Springer, 2009.

[138] Johannes Blömer and Jan Bobolz. Delegatable attribute-based anonymous credentials from dynamically malleable signatures. In *International Conference on Applied Cryptography and Network Security*, pages 221–239. Springer, 2018.

[139] Jan Camenisch, Manu Drijvers, and Maria Dubovitskaya. Practical UC-secure delegatable credentials with attributes and their application to blockchain. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 683–699, 2017.

[140] Elizabeth C Crites and Anna Lysyanskaya. Delegatable anonymous credentials from mercurial signatures. In *Cryptographers' Track at the RSA Conference*, pages 535–555. Springer, 2019.

[141] Scott Griffy, Anna Lysyanskaya, Omid Mir, Octavio Perez Kempner, and Daniel Slamanig. Delegatable anonymous credentials from mercurial signatures with stronger privacy. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 296–325. Springer, 2024.

[142] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong Diffie Hellman assumption revisited. In *Trust 2016*, volume 9824 of *LNCS*, pages 1–20, 2016.

[143] The European Digital Identity Wallet Reference Implementation Roadmap, 10 2025.

[144] Daniel Fett, Kristina Yasuda, and Brian Campbell. Selective Disclosure for JWTs (SD-JWT). Internet-Draft draft-ietf-oauth-selective-disclosure-jwt-12, Internet Engineering Task Force, September 2024. Work in Progress.

[145] Manu Sporny, Dave Longley, and David Chadwick. Verifiable Credentials Data Model, 03 2022.

[146] Jack Doerner, Yashvanth Kondi, Eysa Lee, Abhi Shelat, and LaKyah Tyner. Threshold bbs+ signatures for distributed anonymous credential issuance. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 773–789. IEEE, 2023.

[147] Julia Hesse, Nitin Singh, and Alessandro Sorniotti. How to bind anonymous credentials to humans. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 3047–3064, 2023.

[148] Michele Orrù. Revisiting keyed-verification anonymous credentials. *Cryptology ePrint Archive*, 2024.

[149] Andrea Flamini, Eysa Lee, and Anna Lysyanskaya. Multi-holder anonymous credentials from bbs signatures. In *Annual International Cryptology Conference*, pages 325–357. Springer, 2025.

[150] Rutchathon Chairattana-Apirom and Stefano Tessaro. On the concrete security of bbs/bbs+ signatures. *Cryptology ePrint Archive*, 2025.

[151] Rutchathon Chairattana-Apirom, Franklin Harding, Anna Lysyanskaya, and Stefano Tessaro. Server-aided anonymous credentials. In *Annual International Cryptology Conference*, pages 291–324. Springer, 2025.

[152] Nicolas Desmoulins, Antoine Dumanois, Seyni Kane, and Jacques Traoré. Making bbs anonymous credentials eidas 2.0 compliant. *Cryptology ePrint Archive*, 2025.

[153] Rutchathon Chairattana-Apirom, Dennis Hofheinz, and Stefano Tessaro. Tight security for bbs signatures. *Cryptology ePrint Archive*, 2025.

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

[155] David Pointcheval and Olivier Sanders. Short randomizable signatures. In *Topics in Cryptology-CT-RSA 2016: The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29-March 4, 2016, Proceedings*, pages 111–126. Springer, 2016.

[156] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Annual international cryptology conference*, pages 56–72. Springer, 2004.