CONSECUTIVE DELEGATABLE SIGNING RIGHTS FOR THE ISSUANCE OF ANONYMOUS ATTRIBUTE-BASED CREDENTIALS

HONG YI VICTOR LI

to obtain the degree of Master of Science in Computer Science
Software Technology Track
with a 4TU specialization in Cyber Security
to be defended publicly on October 29, 2018

DELFT UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering, Mathematics & Computer Science
Department of Intelligent Systems
Cyber Security Group

Hong Yi Victor Li: *Consecutive Delegatable Signing Rights for the Issuance of Anonymous Attribute-Based Credentials*, Master of Science, © October 2018

STUDENT NUMBER:
4218906

COMMITTEE MEMBERS:
Dr. Odette Scharenborg
Dr. Zekeriya Erkin
Dr. Annibale Panichella
Dr. Ir. Zhijie Ren

SUPERVISORS:
Dr. Zekeriya Erkin
Dr. Ir. Zhijie Ren

ABSTRACT

Digital identities and credentials are gradually replacing physical documents, as they can be verified with more accuracy and efficiency. Since online privacy is becoming more crucial than ever, it is essential to preserve the privacy of individuals whenever possible. Therefore, anonymous attestation of digital credentials should be feasible, where provers can selectively disclose attributes and create abstractions over attributes in their credential, in order to solely disclose the minimum amount of information required to complete the goal of verification.

Many schemes in the field of attribute-based credentials consider a single root authority issuing credentials to provers. This is coherent to the traditional way of the issuance of credentials since the process of producing physical documents is costly to distribute to multiple issuers. Digital identities provide the opportunity for authorities to distribute credential issuance rights (consecutively) to smaller entrusted entities.

To the best of our knowledge, we propose the first protocol which combines both anonymous attestation with attribute-based credentials and the delegation of selective signing rights for the issuance of these credentials. Root authorities could delegate signing rights for selective attributes consecutively to trustees, which are able to create anonymous attribute-based credentials with the acquired attributes for provers. Verifiers are able to verify presentation tokens with solely the public key of the root authority, without gaining knowledge about the identities of the prover and intermediate delegators. We introduce three adapted signature schemes based on existing work in order to realize a concrete instantiation of the protocol. Anonymity is achieved by incorporating Schnorr's zero-knowledge proof of knowledge with bilinear pairings to efficiently prove the correctness of presentation tokens.

We realized a prototype of our concrete instantiation and optimized the verification algorithm in order to achieve optimal pairing performance. Complexity analysis of the protocol shows improvement in efficiency by aggregating attribute signatures throughout signing right delegation. Experimental results demonstrate a degree of practical feasibility for the verification of presentation tokens on commodity hardware within the challenging public transportation access control time bound of 300 ms.

PREFACE

When I started my master's thesis project, I joined the research group in the field of applied cryptography with a huge interest in blockchain. During the study of relevant literature, I also got curious about digital identity and anonymous credentials. My indecisiveness of interest resulted in a slow start and had put me into a journey filled with ups and downs. Now finally, nine months later, I finished my report on digital identity and anonymous credentials, which I did not foresee at the beginning of my masters, especially since I did not expect to conduct research in the field of cryptography at all.

First of all, I would like to express my gratitude to my thesis supervisors Zeki and Zhijie, for guiding me through the past nine months. Even though the first months were tough, they always have provided the support I needed in many ways and kept me focused on creating the work that lays in front of you today.

I also wish to acknowledge the help provided by Peter Schwabe from Radboud University, who gave advise and feedback on my work.

Furthermore, I am grateful to Zeki for providing us with an environment within the research group where we closely interacted with each other. We always had a great time, whether we had coffee or discussing cryptographic schemes. Therefore, a lot of appreciation goes out to my fellow master and PhD-students, for making the thesis period at least more enjoyable. Besides all the hard work, there was always a little room for some fun and, of course, (cheese)cake!

Additionally, I would like to express my appreciation to Zhijie for the countless brainstorming sessions, even when it did not involve blockchain and consensus algorithms, and to Gamze, Oğuzhan and Chibuike for helping me out whenever I had questions.

Finally and most importantly, I would like to thank my family and friends who have always stood by me, especially them who helped me through my thesis in any way. I am grateful to have them in my life, and I could not have achieved this work without them. Special thanks to Lars, Hari, Wielande, Tim, Wing Yi and Christina for their efforts during the creation of this work.

Victor Li
Delft, October 2018

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACRONYMS

APS Anonymous Proxy Signature

DAC Delegatable Anonymous Credential

DAAC Delegatable Attribute-Based Anonymous Credential

CL Camenisch and Lysyanskaya Signature [24]

BBS Boneh, Boyen and Shacham Signature [13]

PS Pointcheval and Sanders Signature [64]

1 INTRODUCTION

Citizens carry wallets full of physical documents in order to identify themselves for a variety of purposes. While presenting physical documents has become a force of habit, there are numerous drawbacks to this approach of identification. First of all, presenting physical documents often leaks more information to verifiers than required to complete the task at hand. As society is becoming increasingly privacy-aware [3, 71], people strive to minimize information disclosure whenever possible. Furthermore, currently used physical documents require manual inspection during verification to ensure proper unforgeability characteristics to be present. However, these need to be assessed within a short time frame, and can only be judged by visual inspection to a limited extent.

In the era of digitalization, physical documents are being replaced with digital identities and credentials [76]. Various governments have already employed initial identity solutions in a digital format, while others are currently exploring options to realize more extensive digital identities [31]. Furthermore, the current movement that is embodying Self-Sovereign Identity is striving towards the notion of user-autonomous identities with which individuals are in full control of the information they provide [29]. Users can create and use their identities freely across any number of authorities while maintaining control over their privacy. Individuals are able to make claims about themselves, where endorsements backed by relevant authorities or peers increase their credibility.

Technological progressions in digital identities and concepts embracing user autonomy for identity require more extensive cryptographic building blocks, in order to reliably employ digital identities and credentials. The research field focused on these building blocks is zooming into solutions and push towards more secure and practical underlying protocols.

Digital credentials provide the opportunity for large-scale credential providers to distribute credential issuance rights to smaller entrusted entities, in order to lessen the workload and management tasks of the root entity. Various protocols have been proposed for anonymous attestation with attribute-based credentials, as well as for the delegation of anonymous credentials or signing rights (Section 3.5). Nevertheless, current approaches either only focus on authentication, do not involve attribute-based credentials, do not offer complete privacy for provers, or lack practical efficiency.

To the best of our knowledge, we are the first to present a protocol that combines both anonymous attestation of attribute-based credentials and the delegation of selective signing rights for the issuance of these credentials. We modified three existing signature schemes in order to realize a concrete instantiation of the protocol to form a building block towards more flexible and advanced digital identities.

1.1 DIGITAL IDENTITIES AND CREDENTIALS

Due to rapid technological advancements, society is shifting towards the digitalization of all physical documents. Library cards, paper contracts, and passports are good examples of documents which will be replaced with digital substitutes. Not only are digital alternatives more trustworthy and efficient, but they also offer reduced costs for both creators and verifiers of these documents. Digital identity is a fundamental building block to achieve the realization of these digital documents.

The primary driver for the realization of digital identities is to identify entities with more accuracy, reliability, and efficiency. Governments and companies are currently investing heavily in digital identity solutions. In early 2018, it was estimated that banks alone collectively spend more than $1 billion per year on research and development in the field of digital identity [47]. The World Economic Forum estimates that $12 billion of total investments is necessary in order to realize digital identities for everyone [76].

The realization of solely digital identities is not sufficient. Entities holding digital identities often wish to prove to other parties that they have specific attributes or traits obtained from a trusted party. Attribute-based credentials enable issuers to assign values to attributes which allows them to define the traits of the prover more specifically and dynamically, e.g., "`city = Amsterdam`" or "`birth_date = 19940101`". It can be applied in a variety of use cases ranging from electronic passports to "Privacy-Aware Smart Health Services in IoT-Based Smart Cities" [46].

In various cases, the verification party solely needs to verify whether the prover has specific attributes of particular values, endorsed by a trusted party. A well-known example is where a customer visits a liquor store to purchase alcoholic beverages. Before proceeding the sale, the store is required by law to verify the age of its customers. Currently, customers have to present a physical identification card which also includes information not required for the verification of age, enabling the store to gain unintentional knowledge of other non-relevant attributes on the credential. Although data extraction unlikely with physical credentials, this would be more feasible with digital credentials, allowing the liquor

store to use the extraneous information to conduct data analysis on their customers without their consent or to reuse presented credentials anywhere else.

1.2 PRIVACY AWARENESS

Individuals are becoming more aware of privacy and thereby getting more concerned about the privacy risks associated with online activity [3, 71]. This trend is mainly fueled by three factors.

First of all, data breaches of online platforms and businesses occur more frequently. For example, LinkedIn got breached in 2012 by adversaries, leaking personal information of 117 million users [74]. Since 2013, *Have I Been Pwned* [1] has collected more than 5 billion online user accounts from data breaches. Adversaries, for instance, use the collected information to access other online platforms or to impersonate individuals. Even though breached platforms and businesses react as fast as they can, leaked information can never be recovered from the internet.

Additionally, the recent data scandal involving Facebook and Cambridge Analytica has occurred prominently in the news. Personal information has been collected by Cambridge Analytica in order to influence the opinion of the voter [72]. The misuse of personal information led to concerns among people, which is increasingly questioning the safety of their personal data and privacy.

Finally, the increasing realization of the importance to protect the privacy of citizens is also growing in politics, due to the increasing privacy risks. This led to more legislation and awareness to protect the online privacy of individuals. For instance, the General Data Protection Regulation (GDPR) law [38] went into effect in the European Union. This law protects the online privacy of individuals by enforcing new rules for consent, data collection, data processing, and profiling of individuals, thereby giving individuals more control over their private information.

1.3 SELF-SOVEREIGN IDENTITY

The emerging movement embodying Self-Sovereign Identity is striving towards user-autonomous identities, where individuals are central to the administration of identity [29]. Currently, users are locked into various identity providers who are able to deny their identity or even confirm a false identity. The reduction of power previously located at these centralized entities allows individuals to use their identities freely across any number of authorities and being

1 Have I Been Pwned https://haveibeenpwned.com/

in full control of their personal information, which inherently increases privacy.

The realization of Self-Sovereign Identity in its purest form is however challenging. Since the trust in centralized identity providers is omitted, new frameworks are essential to establish trust among entities. Determining the trustworthiness of claims is difficult since trust is established through a peer-to-peer economy. Nevertheless, relevant authorities could issue attestations to provers, which enables trust depending on the nature of the claim and the use case.

The essence of Self-Sovereign Identity is to enable individuals to store their identity data locally, instead of on many centralized repositories across the internet. Users are free to create their own identity and make claims about themselves or others, where endorsements backed by relevant authorities or peers should increase the credibility of such claims. It allows users to disclose selective information to verifiers whenever necessary. In this way, individuals are in full control of the information they provide to others and do not have to worry about the inherent risk of identity data stored at central repositories.

1.4 ANONYMOUS ATTESTATION

As privacy becomes more important in society, it is essential to preserve the privacy of an individual whenever possible. While Self-Sovereign Identity is striving for ownership of personal data, it should also be possible to enhance privacy during verification of claims as opposed to the presentation of physical credentials. The information disclosed during the presentation of claims should be reduced and only disclose the minimum amount of information that is required to complete the goal of verification. Therefore, provers should be able to attest claims selectively and anonymously and be able to apply abstraction on attributes by attesting predicates over their attributes values in the credential. In the case of purchasing alcoholic beverages, customers should be able to prove that their age is at least 18 or higher, without revealing their date of birth, any other attributes in the provided credential nor their identity to the store.

Several cryptographic schemes for attribute-based credentials with anonymous attestation have already been proposed in the scientific field (Section 3.5.1). Nevertheless, they all rely on a single central authority for the issuance of credentials. The presence of the peer-to-peer trust model within Self-Sovereign Identity motivates a similar approach, using the consecutive trust concept within anonymous claims to increase their credibility. An authority could declare trust consecutively towards smaller entrusted entities, which are able to endorse specific claims for provers on behalf of the

authority. We portray this concept to a more concrete problem scenario next.

1.5 DELEGATABLE SIGNING RIGHTS FOR CREDENTIALS ISSUANCE

In large-scale organizations, root entities often already delegate the management of specific tasks to other trusted entities. The root entity would pass selective authorization rights downwards in the hierarchy, in order to reduce its workload. This enables lower-level entities to act on behalf of the root entity for permitted actions.

The notion of task delegation can be extended to the issuance of credentials, where the root entity is a trusted authority which issues credentials to provers. Nevertheless, credential issuance from this authority occurs on a large scale. Therefore, the authority could delegate signing rights for specific attributes to its departments. The departments are able to issue credentials containing acquired attributes to provers, on behalf of the authority. Since large organizations are often structured hierarchically in multiple layers, entities which obtained signing rights can delegate the rights further downwards.

In order to preserve the privacy of provers during anonymous attestation, the verifiers of credentials should not gain any information about the intermediate delegators. Learning the identity of the intermediate delegators could potentially leak information about the prover and could allow verifiers to link multiple presentations of the same credential with more accuracy. Therefore, the verifier should be convinced that the credential is correct without disclosing any identifiable information about the delegators and the prover.

The concept of delegating rights has been explored in the field of cryptography. Research in the delegation of rights for anonymous attestation occurs in two categories: Anonymous Proxy Signatures and Delegatable Anonymous Credentials.

ANONYMOUS PROXY SINGATURES The root authority assigns a set of attributes to a user, where the attributes act as authorization rights. Users can propagate a subset of the acquired attributes to other users. It also allows users to sign messages, along with particular acquired attributes. The signature is able to prove that the message is signed by an anonymous entity which has acquired the required attributes. The identity of the prover and the intermediate delegators are not disclosed.

DELEGATABLE ANONYMOUS CREDENTIALS The root authority issues a credential to a user with various attributes. Users can

present specific attributes of their credential anonymously to authenticate themselves. It is also possible for users to propagate the received credential to other users. The user is free to choose attributes for the next user. A subsequent user can use this credential to anonymously authenticate himself by showing particular attributes assigned to him and users of prior levels, which results in an attribute chain. The credential can only be used for authentication purposes and could require complex business logic for the verifier to decide whenever to accept the attribute chain in the credential.

Both concepts incorporate the delegation of rights but are only suitable for authentication purposes. Nevertheless, schemes for selective signing right delegation for anonymous attribute-based credentials has not been introduced before. The issuance of these credentials in a hierarchical delegation setting could have interesting applications to various use cases.

1.6 USE CASES

We discuss two examples to demonstrate the potential relevance and the possibilities for signing right delegation for the issuance of anonymous attribute-based credentials.

1.6.1 *European Driver's Licences*

The first imaginable scenario is where the European Union wishes to issue electronic driver's licenses to European citizens. A driver's license would contain attributes such as `name`, `gender`, `date_of_birth`, `categories`, and `expiration_date`. As the European Union has many citizens, it is hard to issue and manage driver's licenses for each of them. Therefore, the European Union could delegate the signing rights for these attributes to nations, which in turn could delegate these rights to municipalities. When a citizen obtains the authorization to drive vehicles of a particular category, the municipality could issue a European electronic driver's license to the citizen on behalf of the European Union.

The licensee could then anonymously present particular attributes and the assigned attribute values from the electronic driver's license to verifiers, without disclosing its identity. The credential also does not disclose which nation or municipality was involved in signing the driver's license in order to preserve the privacy of the prover. The verifiers will accept the driver's licenses due to its trust relationship with the European Union.

1.6.2 *Stock Exchange*

The scheme could also be interesting in an investment banking setting. Companies within this industry can have multiple departments with many traders who submit their orders directly to be directly executed on different stock exchanges. In the financial world, many traders are closely watching the trading behavior of particular well-performing traders or investment companies in order to benefit from that information. To prevent analysis of trading behavior by competitors, anonymous trading orders can be incorporated with anonymous attribute-based credentials. In this scenario, the stock exchange would be the root authority, delegating signing rights for placing anonymous orders on the exchange. The stock exchange will delegate the signing rights in multiple delegations towards departments of traders. The department is able to issue specific trading rights to traders. The department could, for example, restrict types of stocks or the monetary limit per order by issuing a credential with specific attributes and attribute values. Traders can use this credential to anonymously place orders directly on the stock exchange. In case of unusual behavior, fraud or insider trading, the stock exchange is able to revoke anonymity in order to inspect orders, and thereby capture any misbehaving adversaries.

1.7 RESEARCH QUESTION

The conception of consecutive trust in a peer-to-peer setting within Self-Sovereign Identity encourages the idea for delegatable signing rights for the issuance of attribute-based credentials. The presented use cases exhibit the advantages for the realization of a scheme which allows the issuance of attribute-based credentials in a hierarchical fashion. Applications of the scheme to other fields and settings could be interesting as well. In this research, we attempt to solve this problem by designing and implementing a protocol for the delegation of signing rights for the issuance of anonymous attribute-based credentials. The main research question of this research is as follows:

> *How can we consecutively delegate signing rights for the issuance of attribute-based credentials, and construct practical, verifiable, and anonymous attribute-based proofs with delegation chain anonymity?*

The following sub-questions are derived from this research question:

(i) How can signing rights for attribute-based credentials be delegated in a more efficient manner opposed to existing schemes?

(ii) How can delegators issue valid attribute-based credentials on behalf of the root authority?

(iii) How can provers transform such a credential into proofs for anonymous attestation?

(iv) How can verifiers check the proofs without the knowledge of the identity of the intermediate delegators and the prover?

(v) Can we improve the efficiency of proof verification in terms of computational speed and storage size?

1.8 CONTRIBUTIONS

In this thesis, we propose a protocol for consecutive delegation of signing rights for the issuance of anonymous attribute-based credentials. The flexibility of the scheme allows delegators to selectively propagate signing rights to subsequent delegatees and enables provers to selectively present attributes, which achieves complete privacy for provers opposed to existing work. The protocol allows verifiers to check the correctness of the proofs, without learning the identities of intermediate delegators and the prover. As building blocks for the protocol, we introduce adapted three signature schemes, based on existing work. The main contributions of our work are as follows:

- A signature scheme which allows the aggregation of multi-signatures, and incorporates randomizability by multiplication, called *RBLS: Randomizable BLS Multi-Signatures with Public-Key Aggregation*.
- A structure-preserving signature scheme which allows randomizability by multiplication, called *RSPS-M: Randomizable Structure-Preserving Signatures by Multiplication*.
- A short randomizable signature scheme which incorporates both randomizability by exponentiation and multiplication, called *SRS-M: Short Randomizable Signatures by Multiplication*.
- A protocol, which incorporates the three aforementioned signature schemes as building blocks, for the consecutive delegation of signing rights for anonymous attribute-based credentials, called ENTRUST.
- A concrete instantiation of ENTRUST, implemented as a prototype and evaluated in terms of computation time and storage size, which shows feasibility to be used in practice.

1.9 RESEARCH OUTLINE

The research towards answering the research questions and leading to the mentioned contributions is documented in the rest of this work. The report is structured as follows. Chapter 2 gives the reader more background information on the landscape of digital identities and credentials. In Chapter 3, primitives are provided upon which the prior art, the modified signature schemes and our protocol ENTRUST is built, after which the prior art is discussed. Chapter 4 gives the formal definition of the protocol employing delegation of signing rights for the issuance of anonymous attribute-based credentials. Chapter 5 covers ideas, motivation, considerations, and decisions during the design of a concrete instantiation of the protocol. With the concluding remarks of the design phase in mind, we introduce three preliminary signature schemes in Chapter 6. By having the design and the preliminary signature schemes as building blocks, we give a concrete instantiation of our protocol called ENTRUST in Chapter 7. Chapter 8 evaluates the efficiency of the protocol by running experiments on a prototype. The report will be concluded in Chapter 9 with a discussion of limitations, and identifies open problems and future work.

2

BACKGROUND

In the introductory chapter, we have briefly touched upon the concepts of digital identities and credentials, anonymous credentials and self-sovereign identity. In this chapter, we give more in-depth knowledge of these concepts, in addition to the introductory chapter.

2.1 DIGITAL IDENTITIES AND CREDENTIALS

Diverse entities, such as natural persons and organizations, should have a way to identify themselves digitally. This can be realized with digital identities, which are digital representations of entities in the form of cryptographic keys and a set of attributes related to the entity [57]. Digital identities enable the authentication and verification of entities and allow entities to sign and verify digital messages. As described by Rivest et al. in 1978 [66], signatures on digital messages are essential:

> *If electronic mail systems are to replace the existing paper mail system for business transactions, "signing" an electronic message must be possible.*

Besides paper mail systems, this also applies to the replacement of physical documents and credentials. Digital signatures ensure authentication, non-repudiation, and integrity of digital messages.

Digital identity is a fundamental building block for various other digital concepts, such as digital credentials, which is a collection of claims about an entity in the form of attributes issued by a trusted authority. There are three parties in the most basic setting with digital credentials: an issuer, a prover, and a verifier. The *prover* is the entity which needs to show digital proof of a specific claim in order to get access or to obtain a benefit. This verifiable claim (or attestation) can be obtained by the *issuer*. Upon presentation of the verifiable claim by the prover, the *verifier* will check the correctness and authenticity of the proof. In this case, the verifier has a trust relationship with the issuer, since the verifier accepts all claims which are signed by the issuer. Figure 2.1 depicts this setting.

Sovrin states that two challenges need to be overcome before digital identities can be widely deployed [69]:

- Representation and verification of digital identities and credentials need to be standardized.
- Verification the source and integrity of the digital credentials need to be standardized.

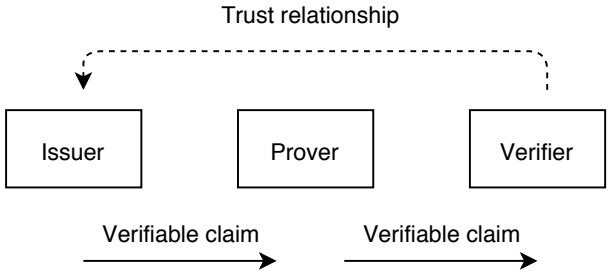

Figure 2.1: Issuer/Prover/Verifier model.

Due to the lack of a standard for the issuance and verification of digital credentials, the World Wide Web Consortium (W3C) created a working group on Verifiable Claims Data Model and Representations [77]. They have released an editor's draft [79], since the working group is still actively discussing terminology and concepts for verifiable claims.

Currently, people use physical documents to prove their identity and their issued attributes. Verifiers determine the authenticity of the document within a limited time-bound with a level of certainty based on physical aspects and unforgeability characteristics of the document. This is challenging in the digital world since the verifier inspects a digital credential, which is essentially data. Therefore digital credentials require different security properties than physical ones. During the verification of physical credentials, the verifier is incapable to directly replicate the credential. One could make a photocopy of the document, but cannot present this to others in a convincing way. Since digital identities are verified by sending data to the verifier, the verifier could store the credential to carry out data analysis or to reuse the credential somewhere else. Therefore, additional properties are required in order to prevent reuse/misuse of presented credentials.

2.2 ANONYMOUS CREDENTIALS

Anonymous claims are similar to verifiable claims. However, the identity of the prover will not be revealed to the verifier. This is advantageous for interactions between provers and verifiers where the identity of the prover is irrelevant for the task at hand. This applies for instances where provers are required to prove their age for the purchase of alcoholic beverages or to attest their membership in a particular group.

While the identity of the prover will not be disclosed to the verifier, attribute values could correlate to the identity of a person. The correlative nature of attributes could be placed in a spectrum of privacy from non-correlatable to highly correlatable. Figure 2.2 depicts the privacy spectrum of different types of attributes.

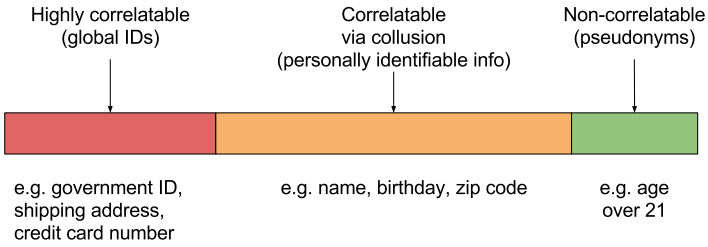

Figure 2.2: Privacy spectrum of attributes from W3C [78].

It is vital in the digital world to preserve the privacy of individuals whenever possible. In various cases, it is nonessential to present exact personal information to verifiers, while the verifier only has to know if the value of a particular attribute lies within a given range, or if the value is part of a given set of values. In addition, the verifier should not gain any information about the values of other non-relevant attributes. Therefore, the prover should be able to selectively present attributes and create abstractions on attribute values in the form of predicates. This leads to shifting attributes in the privacy spectrum towards less correlatability and minimize the disclosed information to the verifier.

Even when provers are able to present valid credentials, it is also essential for the verifier to be sure that the anonymous credentials are linked to the physical persons in reality. Since the verifier does not learn the identity of the prover, the verifier has no guarantee that the physical person owns the identity behind the anonymous credential. Measures need to be incorporated to ensure that the identity and credential have not been transferred or relayed from someone else. Identity solutions, such as ePassports [49], are attempting to mitigate this by incorporating biometrics to confirm that the physical prover is the owner of the digital identity. Nevertheless, confirmation of the relation between digital identity and physical presence is challenging, especially in the emerging concept of Self-Sovereign Identity, where users are central to the administration of their own identity.

2.3 SELF-SOVEREIGN IDENTITY

Self-Sovereign Identity is the user-autonomous concept of entities creating, managing and controlling their own identity without relying on any trusted parties/anchors such as governments and other authorities. Every entity can freely create their own digital identity and make claims about themselves and others, allowing individuals to have full control over their identity. Trust is established in a peer-to-peer economy, where endorsements backed by peers should increase the trustworthiness of claims.

The precedent concept of Self-Sovereign Identity is called "Web of Trust" and has been introduced by Phil Zimmerman in the GPG manual from 1992 [63]:

> *As time goes on, you will accumulate keys from other people that you may want to designate as trusted introducers. Everyone else will each choose their own trusted introducers. And everyone will gradually accumulate and distribute with their key a collection of certifying signatures from other people, with the expectation that anyone receiving it will trust at least one or two of the signatures. This will cause the emergence of a decentralized fault-tolerant web of confidence for all public keys.*

Christopher Allen wrote an article to define the main principles of Self-Sovereign Identity [29]. Others have adopted these main principles in assessing self-sovereign identity solutions [73]. The ten principles of self-sovereign identity according to Allen are existence, control, access, transparency, persistence, portability, interoperability, consent, minimalization and protection.

Multiple initiatives [30, 68] and various researchers [61, 70] are currently exploring the possibilities to design protocols and realize solutions towards self-sovereign identity. Hyperledger Indy [55] is a project which creates a foundation for decentralized identities. The Sovrin Foundation is working towards a framework which leverages the foundation of Hyperledger Indy to build a decentralized ecosystem with self-sovereign identity and a trust framework for users and organizations.

Nevertheless, most of the current solutions are not entirely self-sovereign, since it is hard to bootstrap a vast ecosystem without any trust anchors. Therefore it is hard to determine the trustworthiness of other identities since everyone is able to create identities and claims freely. In order to realize self-sovereign identity in the purest form, researchers are exploring trust estimation frameworks [65].

3

PRIMITIVES AND PRIOR ART

As the prior work and this research are built upon existing concepts and scheme, we introduce the primitives which are used and required to understand the rest of this report. First, various cryptographical concepts and assumptions are discussed upon which the prior art and this work are built. Finally, we discuss and analyze the prior art of anonymous credentials and delegatable signing rights.

3.1 BILINEAR GROUPS

A *bilinear group* is a tuple $\Lambda = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ where:

- $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are multiplicative cyclic groups of prime order q;
- g_1 and g_2 generate $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively;
- $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficient non-degenerate bilinear map;
- for all $X \in \mathbb{G}_1$, $Y \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_q$, $e\left(X^a, Y^b\right) = e(X, Y)^{ab}$.

$\mathsf{BGGen}(1^\kappa)$ is a bilinear-group generator with an polynomial-time algorithm that takes a security parameter κ, and outputs a bilinear group tuple $\Lambda = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$.

3.1.1 *Types of Bilinear Pairings*

Bilinear pairings are categorized into three types [48]:
Type-1 (Symmetric): $\mathbb{G}_1 = \mathbb{G}_2$.
Type-2 (Asymmetric): $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is an efficiently computable homomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. Efficient secure hashing to elements in $\mathbb{G}_2$ is not possible.
Type-3 (Asymmetric): $\mathbb{G}_1 \neq \mathbb{G}_2$ and there exists no efficiently computable homomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$. Secure hashing is possible to both $\mathbb{G}_1$ and $\mathbb{G}_2$.

Type-1 pairings were mostly used at the beginning of pairing-based cryptography but have been increasingly replaced by Type-3 pairings. Type-3 pairings appear to be more efficient [27] and are compatible with computational assumptions, such as the *Decision Diffie-Hellman* problem (Assumption 3) in both $\mathbb{G}_1$ and $\mathbb{G}_2$. Therefore in this work, we only consider Type-3 bilinear pairings.

3.2 COMPUTATIONAL PROBLEMS

The very nature of cryptography depends on computationally hard problems, where it should take adversaries an infeasible amount of computation beyond the reach of current and foreseeable future to break a cryptographic scheme. In this section, we give an overview of several computational hard problems, which will be referred to in the rest of the thesis.

Assumption 1 (Discrete Logarithm (DL) Problem): The *discrete logarithm* assumption in a group $\mathbb{G}$ with generator g states that given (g, g^a) for random $a \in \mathbb{Z}_q$, it is hard to a polynomial-time adversary to find $a' \in \mathbb{Z}_q$, such that $g^{a'} = g^a$.

Assumption 2 (Static Diffie-Hellman (SDH) Problem): The *Static Diffie-Hellman* assumption in a group $\mathbb{G}$ states that given (g, h, g^a) for $g, h \in \mathbb{G}$ and a random $a \in \mathbb{Z}_q$, it is hard to a polynomial-time adversary to find $a' \in \mathbb{Z}_q$, such that $h^{a'} = h^a$.

Assumption 3 (Decision Diffie–Hellman (DDH) Problem): The *Decision Diffie–Hellman* assumption in a group $\mathbb{G}$ with generator g states that given (g^r, g^s, g^t) for random $r, s \in \mathbb{Z}_q$, it is hard to decide whether $t = rs$ or t is random.

This assumption does not hold for symmetric pairings, since the decision is easy to make by checking $e(g^r, g^s) \stackrel{?}{=} e(g^{rs}, g)$.

Assumption 4 (Symmetric External Diffie–Hellman (SXDH) Problem): For $\Lambda = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, the *DDH* assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

Assumption 5 (Linear Problem): The *Linear* assumption [37] in a symmetric group $(q, \mathbb{G}, \mathbb{G}_T, e, g)$ states that given $(g^a, g^b, g^{ac}, g^{bd}) \in \mathbb{G}$ for random $a, b, c, d \in \mathbb{Z}_q$, it is hard to compute g^{c+d}.

Assumption 6 (Decision Linear (DLIN) Problem): The *Decision Linear* assumption [13] in a symmetric group $(q, \mathbb{G}, \mathbb{G}_\mathbb{T}, e, g)$ states that given $(g^a, g^b, g^{ra}, g^{sb}, g^t)$ for random $a, b, r, s \in \mathbb{Z}_q$, it is hard to decide whether $t = r + s$ or t is random.

Assumption 7 (Computational co-Diffie-Hellman (co-CDH) Problem): The Computational co-Diffie-Hellmann problem [16] states that given $(g_1, g_2, g_1^a, g_1^b, g_2^b)$ for random $a, b \in \mathbb{Z}_q$, it is hard to compute g_1^{ab} in polynomial-time.

3.3 RANDOM ORACLE MODEL

A random oracle can be seen as a black box, which takes a bit string of arbitrary length as input. When the random oracle has never observed the input before, it outputs a completely new output,

random and uniform over a predetermined range, and retains the input-output combination. Every time the random oracle has seen the same input as before, it outputs the same random output for that input.

In the ideal environment, the random oracle outputs a unique random output for every unique input, where it would not be possible to obtain the same output for two different inputs. In practice, hash functions are used and treated as random oracles [32]. Due to the nature of hash functions, they can never fully act as random oracles, since hash functions are algorithms which scrambles the input data to a random-appearing output. Therefore hash functions can never guarantee that there will be no output collisions for every two distinct inputs.

Bellare and Rogaway were the first to describe random oracles as primitives for cryptographic protocols [7], and how hash functions could model the behavior of random oracles.

Researchers do not favor schemes in the Random Oracle Model [58], since schemes in this model are secure in a theoretical sense, but the security of random oracles cannot be completely realized in practice. Therefore, researchers rather avoid random oracles or transform schemes to other security models.

Nevertheless, schemes in the Random Oracle Model appear to require less storage space and are faster in computation [58]. Although hash functions might not offer a full guarantee on non-colliding inputs, it is still widely used to verify the integrity of data with high probability.

3.4 ZERO-KNOWLEDGE PROOFS

Zero-Knowledge Proofs are proofs which could convince other parties with high probability that the sender knows a secret value, without revealing it. The concept of proof of knowledge has been formally introduced by Feige et al. [39], while Goldwasser et al. [50] were the first to introduce the formal definition of zero-knowledge.

3.4.1 *Schnorr's Zero-Knowledge Proof of Knowledge*

Schnorr's Zero-Knowledge Proof of Knowledge is a three-stage protocol, which allows a prover to convince the verifier that he knows a secret value, without disclosing the secret value. The protocol consists of the commitment, challenge and response phase. These phases will be repeated until the verifier is convinced that the prover knows the hidden value. Algorithm 1 depicts the three-phase protocol. In this example, the prover proves the knowledge of the secret exponent x in the public element $A = g^x$.

Algorithm 1 Schnorr's Zero-Knowledge Proof of Knowledge Protocol

Prover		Verifier
Choose $t \xleftarrow{R} \mathbb{Z}_q$		
Compute $T = g^t$	$\xrightarrow{T}$	
	$\xleftarrow{c}$	Choose $c \xleftarrow{R} \mathbb{Z}_q$
$s = x \cdot c + t$	$\xrightarrow{s}$	
		Accept if: $g^s \stackrel{?}{=} A^c T$

Using the Fiat-Shamir heuristic [40], the interactive protocol can be transformed into a non-interactive protocol, where a hash function will generate the challenge for the prover. Algorithm 2 shows the algorithm of the Schnorr's Zero-Knowledge Proof of Knowledge protocol with Fiat-Shamir transform to prove the knowledge of the secret exponent x in $A = g^x$, where Λ is the common reference string containing public parameters.

Algorithm 2 Schnorr's Zero-Knowledge Proof of Knowledge Protocol with Fiat-Shamir Transform

1: **procedure** CREATE(Λ, x)
2: $A \leftarrow g^x$
3: $t \xleftarrow{R} \mathbb{Z}_q$
4: $T \leftarrow g^t$
5: $c \leftarrow H(A||T)$
6: $s \leftarrow t + c \cdot x$
7: **return** (s, T)
8: **end procedure**
9: **procedure** VALIDATE($\Lambda, A, (s, T)$)
10: $c \leftarrow H(A||T)$
11: **return** $g^s = A^c T$
12: **end procedure**

3.4.2 *Groth-Sahai Proofs*

Groth and Sahai realized an efficient and modular proof system [53], which allows the creation of non-interactive zero-knowledge proofs and non-interactive witness-indistinguishable proofs over a large class of statements over bilinear groups. The most interesting type of the Groth-Sahai proofs for bilinear pairings is the pairing-product equation, which consists of products of pairings applied to the variables and constants from the cyclic group.

The idea behind pairing-product pairings in Groth-Sahai proofs is that bilinear pairings with variable elements $\mathcal{X}_1, \ldots, \mathcal{X}_n \in \mathbb{G}_1, \mathcal{Y}_1, \ldots, \mathcal{Y}_n \in \mathbb{G}_2$ in the form of

$$\prod_{i=1}^{n} e\left(\mathcal{X}_i, \mathcal{Y}_i\right)$$

can always be rewritten as

$$\prod_{i=1}^{n} e\left(\mathcal{A}_i, \mathcal{Y}_i\right) \cdot \prod_{i=1}^{n} e\left(\mathcal{X}_i, \mathcal{B}_i\right) \cdot \prod_{i=1}^{n} e\left(\mathcal{X}_i, \mathcal{Y}_i\right)^{\gamma_i} = t_T\,,$$

where $\mathcal{A}_i \in \mathbb{G}_1, \mathcal{B}_i \in \mathbb{G}_2, t_T \in \mathbb{G}_T$, and $\gamma_i \in \mathbb{Z}_q$ are constants. This pairing-product equation can be rewritten, such that it could proof the knowledge over the variables, without disclosing them.

Groth and Sahai generalizes this pairing-product equation for future use of function e, where matrices might be applicable. This pairing-product equation with variable elements $\mathcal{X}_1, \ldots, \mathcal{X}_m \in \mathbb{G}_1, \mathcal{Y}_1, \ldots, \mathcal{Y}_n \in \mathbb{G}_2$ is in the form of

$$\prod_{i=1}^{n} e\left(\mathcal{A}_i, \mathcal{Y}_i\right) \cdot \prod_{i=1}^{m} e\left(\mathcal{X}_i, \mathcal{B}_i\right) \cdot \prod_{i=1}^{m} \prod_{j=1}^{n} e\left(\mathcal{X}_i, \mathcal{Y}_j\right)^{\gamma_{ij}} = t_T\,,$$

determined by constants $\mathcal{A}_j \in \mathbb{G}_1, \mathcal{B}_i \in \mathbb{G}_2, t_T \in \mathbb{G}_T$, and $\gamma_{ij} \in \mathbb{Z}_q$.

Groth-Sahai proofs can be applied to a wide range of statements over bilinear pairing, which makes it very modular to combine with other schemes. There is a wide variety of schemes which incorporates this proof system [1, 5, 6, 45].

While Groth-Sahai proofs are efficient in a complexity theoretical sense, it is not practical due to the large credential size and many pairing operations to verify the proof [20]. There is a significant performance gap between schemes which use Groth-Sahai proofs and schemes which are secure in the Random Oracle Model (Section 3.3). Due to the large overhead in computation and storage, it seems that Groth-Sahai proofs in the current form are not efficient enough in practice.

3.5 PRIOR ART

The concepts of anonymous credentials and consecutive delegation of signing rights has been studied before by other researchers in the field of cryptography. The chapter gives an overview of the existing techniques and schemes, where anonymous credentials and delegation of signing rights are realized. We analyze the shortcomings of the prior art and conclude with the observations from the analysis.

3.5.1 *Anonymous Credential Schemes*

According to Camenisch et al. [22–24], there are three requirements in order to realize anonymous credentials. It is sufficient to incorporate a commitment scheme, a signature scheme and efficient protocols to (1) proof equality of two committed values; (2) getting a signature on a committed value; and (3) proving the knowledge of a signature on the committed value.

Camenisch and Lysyanskaya [24] provide a commitment and signature scheme, which supports these requirements for Type-1 bilinear pairings. In the paper, they also show how their schemes lead to an anonymous credential system. The authors incorporate Pedersen commitments for the commitment of values. *CL-Signatures* has been widely used by other researchers, which try to improve or extend the signature scheme [13, 21] or to incorporate the flexible signature scheme in their own protocols [9, 11, 28].

Au et al. [2] modified *BBS-Signatures* [13] for the realization of anonymous credentials. The scheme has similar properties to CL-Signatures but is more efficient in signature size and verification costs. However, the BBS-Signature cannot be randomized, and therefore the prover must commit to the signature and prove that he has a valid signature for the commitments.

P-Signatures are introduced by Belenkiy et al. [5] and incorporates Groth-Sahai zero-knowledge proofs. Therefore, the scheme does not rely on the Random Oracle model and allows non-interactive zero-knowledge proofs. As shown in Section 3.4.2, Groth-Sahai are efficient in theory but are not practical due to many expensive pairing operations and a large credential size.

Recently, Pointcheval and Sanders [64] introduced a new signature scheme with the properties for anonymous credentials, which allows signatures on commitments to be of constant size. In addition, the paper also features a scheme which allows sequential aggregation of signatures. The scheme outperforms CL-Signatures and BBS-Signatures in terms of credential size and verification costs. Like the CL-Signature, the *PS-Signatures* allow randomization of the signature by simply exponentiating the signature elements with a random scalar. The scheme requires Type-3 bilinear pairings, which are more efficient and secure compared to Type-1 bilinear pairings.

Several concrete anonymous credential solutions have been introduced. Microsoft developed U-Prove [60], which allows the issuance of single-show credentials. U-Prove provides untraceability, which disallows the issuer and verifier to collude with each other to reveal the identity of the prover during the presentation. Nevertheless, when the credential is shown twice, the verifier is able to link these presentations. Also, the protocol does not support

predicate proofs over attribute values, which allows provers to abstract personal information in credentials.

IBM has introduced Identity Mixer (Idemix) [56], which uses Camenisch-Lysyanskaya signatures for the issuance of anonymous credentials [24]. Idemix credentials are multi-show; therefore a verifier cannot link multiple presentations of the same credential. The prover randomizes the signature of the credential for every presentation. Idemix does support inequality proofs, which allows partial information disclosure.

Hyperledger Indy incorporates IBM's Idemix protocol [59], where they combine CL-Signatures with the anonymous credential construction from Au et al. [2]. This project is used by Sovrin [68] as underlying framework.

3.5.2 *Hierarchical Signing Right Delegation*

The concept of hierarchical group signatures has been introduced by Trolin and Wikström [75]. The scheme generalizes group signatures, where the group is organized in a tree structure with group managers and signers. When a verifier checks the signature produced by a signer, it only learns that someone in the group has signed, without revealing the signer's identity or the identity of group managers that were involved. Group managers can open signatures generated from its own subtree and identify the next group manager/signer in the chain.

3.5.3 *Anonymous Proxy Signatures*

In Anonymous Proxy Signatures (APS), the root entity delegates a set of attributes to a user, where the attributes act as authorization rights. The user can assign subsets of the acquired attributes to subsequent users. This can happen consecutively. It also allows users to sign messages, along with the acquired attributes. The user is able to present the signature to verifiers, which proves anonymously that the message is signed on behalf of the root entity by a user who has obtained the attached attributes.

The concept of Anonymous Proxy Signatures (APS) has been properly introduced by Fuchsbauer et al. [43, 44]. Users create signatures on messages, and verifiers are able to verify the signatures with solely the public key of the root entity. All intermediate delegators and the final signer remain anonymous during the presentation of the signature. In case of misuse or fraud, an opening authority will be able to open the signature and revoke the anonymity of the delegation chain.

[43, 44] gives a proper generalization of group signatures and proxy signatures. The paper gives a theoretical instantiation and

formal definitions of security for the model. However, no concrete instantiation has been given.

In another paper, Fuchbauer et al. [45] do give a concrete instantiation, although it is not practical. The authors use encryption and proofs based on a generalization of techniques of Boyen-Waters Group Signatures [18]. However, the signature scheme is inefficient due to zero-knowledge proofs using bit-by-bit techniques.

Abe et al. [1] introduce automorphic signatures, which are structure-preserving signatures where the verification keys lie in the message space. The authors give an instantiation of anonymous proxy signatures. They describe how automorphic signatures could instantiate the anonymous proxy signatures, but do not give concrete details. The scheme incorporates the proof system of Groth-Sahai (Section 3.4.2).

3.5.4 *Delegatable Anonymous Credentials*

In Delegatable Anonymous Credentials (DAC), the root authority issues credentials to users with various attributes. Users can use their credential to anonymously authenticate themselves for having selective attributes. It is also possible for users to issue credentials for others. The user is free to choose attributes for the next user. Subsequent users can use their credential to anonymously authenticate itself by showing particular attributes from itself and the prior levels. This will form a chain of attributes, where different authority levels can be modeled.

For instance, a user wishes to anonymously authenticate himself on a web forum. The prover can prove that he is a user of the forum by showing the attribute chain of delegation: User $\leftarrow$ Moderator $\leftarrow$ Web Master $\leftarrow$ System Administrator, without disclosing any concrete identities. The verifier only requires the public key of the root authority to verify the credential. In most of the Delegatable Anonymous Credential (DAC) schemes, anonymity also holds between delegators during the delegation of credentials. Therefore, delegators do not know the identities of its predecessors in the delegation chain.

Chase and Lysyanskaya [25] were the first to introduce the concept of Delegatable Anonymous Credentials. Belenkiy et al. [6] formalized the concept with security requirements and gave a construction of DACs. Multiple other constructions have been proposed [26, 42] to improve upon the work of Belenkiy et al. [6], but roughly use the same techniques, such as the Groth-Sahai proof system.

Recently, Camenisch et al. [20] have proposed a practical scheme for Delegatable Anonymous Credentials. In contrast to previous

	CL06 [25]	BCC+09 [6]	Fuch11 [42]	CDD17 [20]	BB18 [12]
ZKP	Generic	Groth-Sahai	Groth-Sahai	Schnorr	Schnorr
Non-interactive Delegation	V	X	V	V	X
Non-interactive Verification	V	V	V	V	V
Randomizable	X	V	V	V	V
Inter-delegation anonymity	X	V	V	X	V
Attribute-based credentials	X	X	X	X	V
Signature type	Generic	Boneh-Boyen	Automorphic / Commuting	Structure-preserving	Dynamically Malleable Signatures
Credential size increase per delegation	Exponential	$\mathbb{G}_1^{50} \times \mathbb{G}_2^{40}$	$\mathbb{G}_1^{20} \times \mathbb{G}_2^{18}$	$\mathbb{G}_1^{2} \times \mathbb{G}_2^{2}$	-

Table 3.1: Comparison between DAC schemes.

schemes, they do not consider the privacy between the delegators and the delegatee. This is because they believe that in most cases, the credential will be delegated hierarchically and therefore the delegators already know each other. By omitting this privacy notion, expensive primitives can be avoided to make the scheme practical.

Even more recently, Blömer and Bobolz [12] introduced *Delegatable Attribute-Based Anonymous Credentials from Dynamically Malleable Signatures*. The Delegatable Attribute-Based Anonymous Credential (DAAC) scheme allows root authorities to issue delegatable credentials with fixed indexed attributes to delegators. Delegators can, in turn, decides to delegate the credential further with more restrictions, where concrete values are assigned to attributes. The attribute "`city`" can, for instance, be attached to the value "`San Francisco`", which becomes immutable for other delegatees. Furthermore, delegations can be prevented anytime by lowering the delegation flag, therefore issuing a final attribute-based credential to provers.

Delegation privacy is achieved by removing the identity of the delegator from the credential during the delegation to a delegatee. While this is efficient, the scheme does not incorporate a delegation chain and therefore does not allow any possibility to trace the preceding delegators. The credential is, in fact, a malleable signature which changes ownership when delegated.

3.5.5 *Comparison of Delegatable Anonymous Credentials schemes*

Table 3.1 gives an overview of the different DAC instantiations and the different properties. As can be noticed, the DAC schemes all differ in properties and incorporate different signature schemes as proof systems. Throughout the years, there have been major improvements in reducing the credential size per delegation.

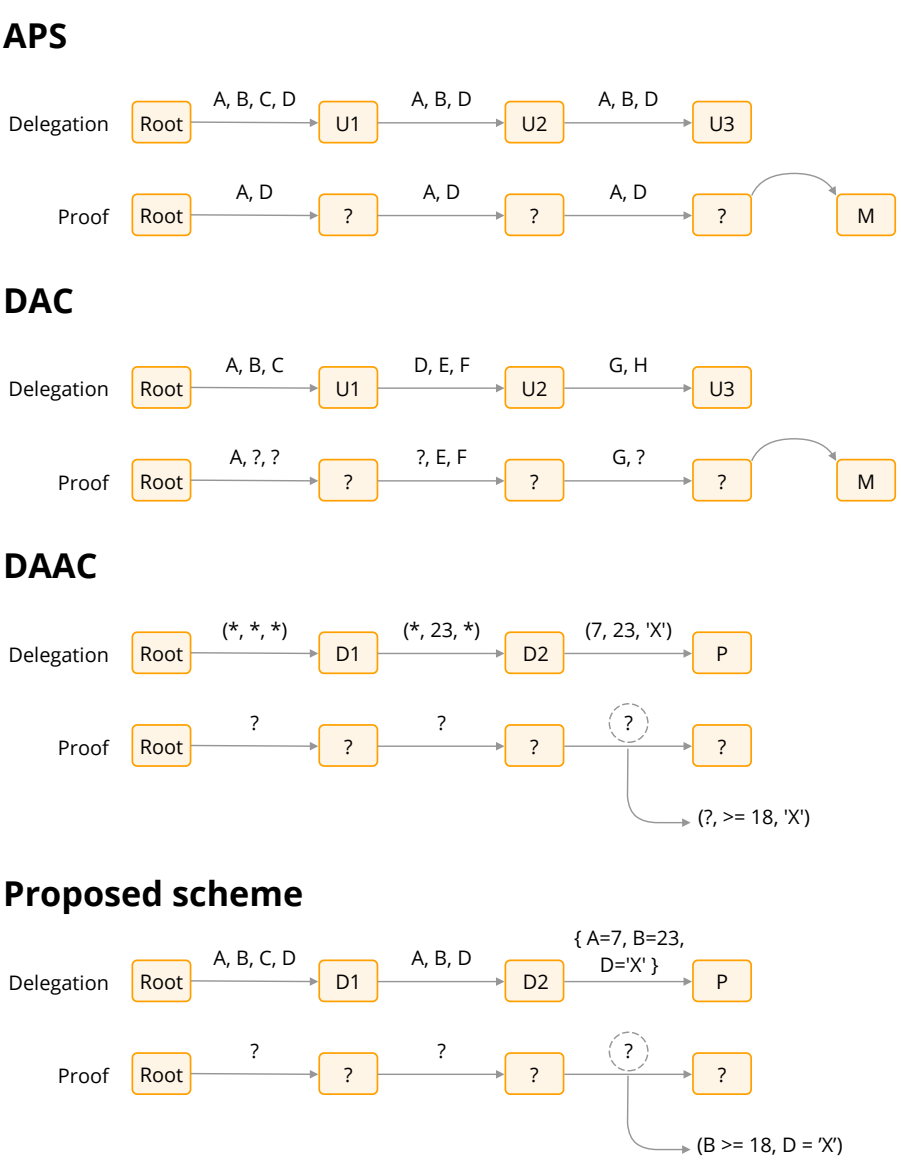

Figure 3.1: Comparison between different types of delegation schemes. For the first two scheme, U3 receive their attributes after three propagations (root → U1 → U2 → U3). The third user is able to sign a message M, along with the acquired attributes. The figure also depicts which information verifiers would learn from proofs during verification. For the last two schemes, D2 receives a credential or signing rights after two delegations. The second delegator creates a credential for the prover P. The prover is able to present predicates on attribute values hidden in commitments.

3.5.6 *Comparsion between Anonymous Proxy Signatures and Delegatable Anonymous Credentials*

Delegatable Anonymous Credentials and Anonymous Proxy Signatures have similar properties, but there are some fundamental differences between the concepts. Figure 3.1 depicts the comparison

between the delegation concepts APS, DAC and DAAC. Entities in APS cannot be an issuer and a prover simultaneously, while in users in DAC and DAAC posses the credential, which they could delegate further. In DAC, delegators are free to choose the attributes for their delegatees, while delegators in APS are only allowed to delegate a subset of their acquired attributes. In DAAC, the attributes are fixed throughout the whole delegation and cannot be changed. The verification phase in DAC could be more sophisticated, since the verifier would need complex business logic to determine if the attributes on each level of delegation occur correctly in the chain to be accepted. Also, the root entity of a DAC is not able to restrict the message signing space of its delegatees.

3.5.7 *Analysis*

By analyzing the prior art of delegatable signing rights and anonymous attribute-based credentials, we have observed the following:

APS schemes do not issue attribute-based credentials. In Anonymous Proxy Signature schemes, it is only possible to sign messages with particular acquired attributes as a user. The delegated attributes only act as authoritative rights of the user who signs the message. Therefore, it is not possible to issue (attribute-based) credentials.

DAC schemes do not issue attribute-based credentials. In Delegatable Anonymous Credential schemes, users are able to assign attributes to subsequent users in the credential. Nevertheless, DAC does not allow users to assign attribute values to specific attributes, which does not allow more flexible credentials in contrast to attribute-based credentials. Furthermore, the scheme can only be used for authentication purposes, since the prover selectively shows the attributes of himself and its predecessors. Therefore, verifiers might require complex business logic to verify the attribute chain and to decide whether to accept the credential.

DAC schemes do not offer complete privacy of intermediate delegators during verification. Delegators in Delegatable Anonymous Credentials sign attributes for its delegatees sequentially. Intermediate delegators and provers are able to mask attributes from its predecessors while maintaining the ability to prove that the signature is valid. However, attribute values are signed sequentially in DAC. Therefore, the indices of the disclosed and hidden attributes are disclosed on each delegation level to the verifier. The verifier could use this information to correlate multiple presentations of the same

credential with more accuracy. Therefore, presentations of DAC do not preserve complete privacy of the provers.

Most of the DAC and APS schemes do not incorporate practical zero-knowledge proofs systems. Various Delegatable Anonymous Credential schemes are efficient in a complexity theoretical sense but are not practical because they use generic zero-knowledge proofs or Groth-Sahai proofs (Section 3.4.2), which has a large credential size and requires many expensive pairing operations [20]. The same applies to Anonymous Proxy Signature schemes.

DAAC does not offer delegation of selective attributes. The DAAC scheme introduced by Blömer and Bobolz [12] offers malleable credentials with fixed credentials, where delegators can selectively set attribute values and make the attribute values unmalleable for delegatees. The downside of this scheme is that the attributes are already fixed in the credential by the root authority, and does not allow delegators to delegate a subset of the attributes in the credential.

Anonymous attribute-based credential schemes lack the ability of consecutive delegation of signing rights. In anonymous attribute-based credential schemes, there is a single root authority which handles the issuance and management of all anonymous attribute-based credentials. This assumption works for many practical cases. However, larger organizations which operate in a hierarchy often delegate the management of specific tasks to other trusted entities. This could also apply to cases where the root authority wishes to delegate selective signing right for the issuance of attribute-based credentials to lower levels in the hierarchy.

3.6 RESEARCH CHALLENGES

From our analysis of the prior art, we observe that various schemes have been proposed for anonymous attestation of attribute-based credentials, as well as for the delegation of signing rights or credentials. Nevertheless, current approaches either only focus on authentication, do not involve attribute-based credentials, do not offer complete privacy for provers, or lack practical efficiency.

We observe the lack of a scheme which allows consecutive delegation of signing rights, for the issuance of anonymous attribute-based credentials. Figure 3.1 depicts the missing scheme. The scheme could have interesting applications for larger organizations where the root authority wishes to delegate issuance of attribute-based credentials downwards in the hierarchy. As shown in Chapter 1, this scheme has interesting potential applications. Our previously stated research question aims to realize a protocol and a

concrete instantiation of the aforementioned scheme. In order to achieve this, there are three main challenges which need to be overcome.

3.6.1 *Adapting and Integrating Multiple Signature Schemes*

Multiple preliminary signatures schemes are required as fundamental building blocks in order to realize the protocol. The primary challenge is to identify, adapt, combine and integrate these signature schemes into a protocol. Different kinds of signatures should be interlinked to each other and should zero-knowledge proved during presentations to verifiers.

3.6.2 *Zero-Knowledge Proof*

The prover cannot directly present its credential to verifiers, since verifiers could learn the identities of the prover or the intermediate delegators, or can link multiple presentations of the same credential together.

A presentation token should be able to prove that (1) the delegation chain is valid, e.g., the issuing delegator has obtained the signing rights from its predecessors correctly, where the first entity in the chain is the root entity; (2) all intermediate delegators have the signing rights of the attributes used in the presentation token; (3) the presented predicates are valid under the attribute value in the signed commitment.

All these statements must be proven without disclosing the signatures, commitments nor the attribute values directly. Since zero-knowledge proofs often lead to large credential sizes and require many expensive pairings, it would be desired to incorporate more efficient and practical zero-knowledge proof systems.

3.6.3 *Efficiency and Practicality*

It is essential that the verification process is efficient in terms of computational time and token size, such that it would achieve practical feasibility. This implies that the presentation token should be verified within a reasonable time and should be as compact as possible. Achieving fast verification and small token size is challenging since the verifier needs to be convinced by verifying multiple zero-knowledge proofs in the presentation token.

4

SYSTEM MODEL DEFINITION

In the prior chapter, we concluded that a scheme which allows consecutive delegation of signing rights for the issuance of anonymous attribute-based credentials has not been introduced before in prior art. In this chapter, we give a formal definition of our system model, which forms the foundation for the design of our protocol.

4.1 ACTORS

In order to give a proper application setting, we define four entities in our protocol:

Root Entity The root entity is an entrusted party by verifiers and (indirectly) issues credentials to provers.

Delegator Delegators gain signing rights over specific attributes from the root entity and act on behalf of the root entity as they are issuing credentials to provers within the allowed message space, or assigning the signing rights further to delegatees.

Prover Provers obtain credentials from delegators and can use them to convince verifiers that they possess specific attributes and values issued on behalf of the root entity.

Verifier Verifiers trust the root entity in a particular context and accepts every credential that has been issued on behalf of the root entity.

Figure 4.1 depicts a schematic overview of the interaction between the entities within the protocol. The root entity assigns signing rights over a set of attributes to a delegator, which allows the delegator to assign the acquired signing rights further to other delegatees. The delegation of signing rights can happen consecutively. Delegators are able to issue credentials to provers with the acquired signing rights attributes on behalf of the root entity. Provers holding a credential can generate presentation tokens from the credential, to selectively proof predicates over the attribute values of selective attributes. The verifier inspects the proof and can be convinced that the prover holds a valid credential with the presented attributes and attribute values.

To get a better grasp of the idea behind the application setting, we provide a depiction of a more concrete application scenario in Figure 4.2. In this example, the prover is an elderly who wishes to prove to a museum that his age is at least 65 in order to receive a benefit (e.g., discount on the entrance fee). The prover can prove this fact to the

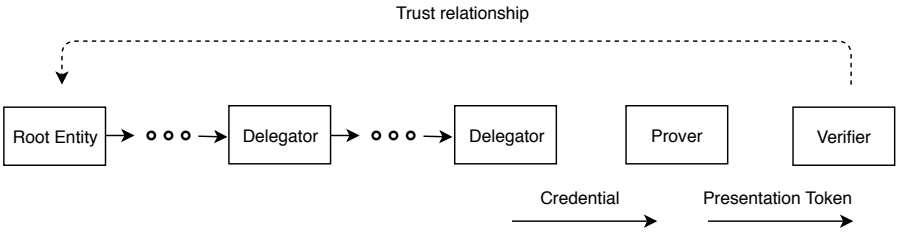

Figure 4.1: Schematic overview of the application setting.

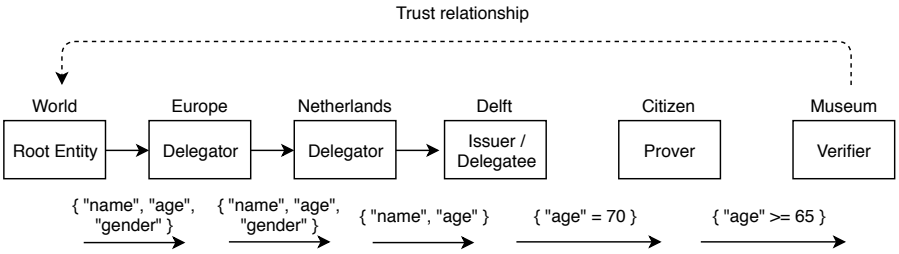

Figure 4.2: Schematic overview of a concrete application scenario.

museum by providing proof, showing that he possesses a credential from the world root entity stating that his age is above 65. Since there are many world citizens on earth, the root entity delegates the signing rights of the attribute "`age`" along with other attributes to its delegatees. After several delegations, the municipality Delft obtains the signing right over the attribute "`age`" and can issue the claim "`age = 70`" to the prover on behalf of the root entity. The prover can generate a presentation token about this claim proving the predicate "`age >= 65`" and present it to the museum.

4.2 ALGORITHMS

We define our formal definition of *delegatable signing rights for the issuance of anonymous attribute-based credentials*, along with the notation of elements which will be used in the rest of the thesis.

The root entity assigns signing rights for credential issuance to a delegator. The first delegator obtaining signing rights directly from the root becomes a level-1 delegator. When the signing rights have been delegated, the level-1 delegator does not need to communicate with the root entity anymore. Delegators can pass its acquired signing rights further to other delegatees. When a delegator on level-L delegates the signing rights to the next delegatee, the delegatee becomes a level-$(L+1)$ delegator. A level-L delegator could also issue a *credential* to a prover, containing attributes and attribute values which the prover is allowed to hold. The prover can generate a *presentation token* containing proofs about the knowledge over signatures and attribute values.

The entities in the protocol interact with each other through five main operations: Setup, Delegation, Credential Issuance, Token Generation and Verification. The definitions for these five operations in the protocol will be discussed in the following subsections.

4.2.1 *Setup*

In the setup phase, an honest party (either the root entity or a trusted entity not related nor involved the protocol) generates a set of public parameters Λ, which will be accessible for all entities participating in the protocol at all times. The root entity sets up its secret and public key (rsk, rpk). The root entity publishes its public key rpk, and will be entrusted by the verifier. A delegator on level L generates a secret and public key (dsk_L, dpk_L). The prover generates the secret and public key (psk, ppk).

4.2.2 *Delegation*

Instead of allowing delegators to generate credentials over the all possible attributes, the root entity defines a set of attributes $\alpha_i \in attr_1$ to delegate to the level-1 delegator. The delegator is thereby only allowed to issue credentials with the acquired attributes on behalf of the root entity. Delegators are able to issue credentials to provers with the acquired attributes or to delegate the signing rights further to a delegatee. The delegator on level L could restrict the signing rights of the delegatee on level $(L+1)$ by delegating a subset of the attributes $\alpha_k \in attr_{L+1}$, where $attr_{L+1} \subseteq attr_L$.

4.2.3 *Credential Issuance*

When the prover wishes to obtain a credential from an issuing delegator holding signing rights $attr_L$, the prover makes various claims he wishes to be endorsed by the delegator. For a selective set of attributes $\alpha_k \in attr_C$, the prover defines attribute values $v_k \in val$ for each attribute α_k. Then, the prover generates commitments $com_{v_k} \in com$ over the attribute values, and send $attr_C$, com and val to the issuing delegator, along with proofs which should convince the delegator that the commitments in com open to the attribute values in val.

Upon receipt of the credential request, the delegator verifies if $attr_C \subseteq attr_L$, decides whether he wishes to endorse the provided claims for the considered prover, and checks if the given commitment opening proofs are valid. When the delegator is convinced of the proofs, the delegator will issue the credential to the prover by signing the attributes and the commitments.

4.2.4 *Token Generation*

The prover is able to use the obtained credential to generate unlinkable presentation tokens to prove selective attributes and attribute values to a verifier. The prover chooses $(\alpha_k, v_k) \in (attr_P, val_P)$, where $attr_P \in attr_C$ and $val_P \in val_C$, and prepares predicate proofs for each attribute pair.

4.2.5 *Verification*

The verifier checks if the presentation token is valid using solely the root's public key rpk, i.e., proofing the prover possesses a valid credential issued on behalf of the root containing the presented attributes and predicates over the signed attribute values. The verifier should not be able to link multiple presentations of the same underlying credential, and should not be able to learn any information including the identity from the prover and the intermediate delegators.

4.3 PROPERTIES

In this section, we define the properties for our protocol, for the delegation of signing rights for the issuance of anonymous attribute-based credentials.

The main properties of the protocol are the following:

Consecutive Signing Rights Delegation The root entity should be able to delegate the signing rights over a set of credential attributes consecutively through intermediate delegators.

Selective Signing Right Delegation Intermediate delegators should be able to delegate a subset of their acquired attributes to a delegatee.

Credential Issuance A delegator may issue a credential for a prover on behalf of the root entity with the acquired attributes.

Selective Anonymous Attestation The prover should be able to create presentation tokens from the acquired credential, which allows anonymous attestation of selective attributes. The verifier should be able to determine the validity of the presentation token with solely the knowledge of the public key of the root entity rpk and the provided presentation token, without learning any information (including identities) about the prover or the intermediate delegators.

Predicate Proofs The prover should be able to create abstractions over the values hidden in the signed commitments. The verifier should be convinced that the presented predicate proofs are

valid without gaining knowledge over the actual value.

From these main requirements, the following sub-requirements are derived:

Complete Delegation Privacy The presentation token should not disclose any information about non-relevant attributes which intermediate delegators acquired during delegation.

Unlinkability The verifier should not be able to link two different presentations of the same credential together.

Proof Integrity The verifier should not be able to reuse or replay the presented presentation token anywhere else.

Non-Interactive Proof The presentation token should contain non-interactive proofs which will be sent to verifiers for inspection.

We consider that once a delegatee obtained signing rights from its delegator, the delegatee does not need to communicate with any of its predecessors to issue credentials or to delegate signing rights to others.

In various other Anonymous Proxy Signature and Delegatable Anonymous Credential schemes (Section 3.5), privacy is achieved by preserving anonymity during both the delegation and the presentation of the credential. Since the delegators are unaware of the identities of its predecessors, the credentials also do not disclose any information about other identities in the chain. Similar to the work of Camenisch et al. [20], we argue that it is unnecessary to achieve this superior privacy guarantee. In most of the real-life scenarios, the entities already know each other, especially in settings where signing rights are delegated. Inherently, there is already trust involved that the delegatee will act faithfully on behalf of the root entity, and performs issuance and delegation as expected. Therefore, there is no need to hide the identities of the delegators during delegation or issuance of credentials. We assume that delegators and provers are allowed to know the identities of its predecessors, which avoids unnecessary and expensive computation. Additionally, it allows us to incorporate Schnorr's Proof of Knowledge (Section 3.4.1) as an efficient proof system for the presentation tokens.

5

PROTOCOL DESIGN

To realize the protocol for consecutive delegation of signing rights for the issuance of anonymous attribute-based credentials, a generic construction need to be designed, in order to select or design the fundamental signature schemes. This chapter covers ideas, motivation, considerations, and decisions while designing the protocol in order to realize a generic construction.

5.1 GENERIC CONSTRUCTION

The entities in the protocol interact with each other through five main operations: the setup, the delegation of signing rights over attributes, the issuance of credentials, the generation of presentation tokens and the verification of the presentation tokens. We will discuss various concepts for our protocol design which will be applied across these algorithms.

The main idea for delegating signing rights is to let the root entity sign the public key of the first delegator, and let the delegator signs the public key of the next delegator. The consecutive delegation will result in a delegation chain of signatures, where the designated delegatee in one signature will be the signer in the next signature. The delegation signatures are represented as $\phi_{r \to D_1}, \phi_{D_1 \to D_2}, \ldots, \phi_{D_{L-1} \to D_L}$.

For each delegation, the delegator delegates a subset of their acquired attributes to the next delegator. To allow the next delegator to use selected attributes for the issuance of credentials, the delegator signs the attributes to delegate to the next delegatee. For each to-be-delegated attribute α_i, the delegator will create an attribute signature σ_{α_i}.

Instead of hiding/masking particular attributes from each delegation level, the delegator will sign each attribute individually. This allows the next delegator to selectively leave out signatures of attributes that he does not wish to pass further to other delegators. In other concrete DAC schemes [12, 20], the delegator signs the attributes for the delegatee sequentially. During delegation, issuance or presentation, the concerned entity can decide to hide non-relevant attributes from various delegation levels. However, this exposes information about the delegators and prover involved in the presentation token to the verifier. The verifier will gain information about the number of attributes for each delegation level, and the indices of the hidden and disclosed attributes for each delegation

level. This information could allow verifiers to correlate delegators and multiple presentations of the same credential.

As discussed before, the delegator on level L delegating attributes $\alpha_1, \ldots, \alpha_n \in attr_{L+1}$ to a level-$(L+1)$ delegator will generate the signatures $(\sigma_{\alpha_1,L}, \ldots, \sigma_{\alpha_n,L}, \phi_{D_L \to D_{L+1}})$. Nevertheless, these signatures are not linked to each other in any way. The signatures only convince verifiers that the same delegator has signed them, but does not tell whether the signatures occur in the same *delegation bundle*. This could allow an adversary to potentially combine signatures from other delegations to craft a delegation bundle which never has been created by a delegator. To make sure that these signatures are linked to each other, the signatures will be randomized with the same random $r \xleftarrow{R} \mathbb{Z}_q$. Since the randomized signatures can then be verified with the same randomization element $R \leftarrow g_2^r \in \mathbb{G}_2$, they are innately related to each other.

In Table 5.1, the signatures created by the root entity and three delegators are depicted for three delegated attributes. The circled groups portray the relations between the signatures and the respective R-values.

	α_1	α_2	α_3	ϕ	R
root	$\sigma_{\alpha_1,r}$	$\sigma_{\alpha_2,r}$	$\sigma_{\alpha_3,r}$	$\phi_{r \to D_1}$	R_1
D_1	$\sigma_{\alpha_1,1}$	$\sigma_{\alpha_2,1}$	$\sigma_{\alpha_3,1}$	$\phi_{D_1 \to D_2}$	R_2
D_2	$\sigma_{\alpha_1,2}$	$\sigma_{\alpha_2,2}$	$\sigma_{\alpha_3,2}$	$\phi_{D_2 \to D_3}$	R_3
D_3	$\sigma_{\alpha_1,3}$	$\sigma_{\alpha_2,3}$	$\sigma_{\alpha_3,3}$	$\phi_{D_3 \to D_4}$	R_4

Table 5.1: Signatures in a delegation bundle: Relation between signatures and R-values.

After creating the signatures and the R-values, the level-L delegator will send a delegation bundle to the delegatee on level-$(L+1)$, consisting of the group elements

$$\left(\langle \sigma_{\alpha_i,r} \rangle_{i=1}^{n}, \phi_{r \to D_1}, R_1, \left\langle \langle \sigma_{\alpha_i,j} \rangle_{i=1}^{n}, \phi_{D_j \to D_{j+1}}, R_{j+1} \right\rangle_{j=1}^{L} \right).$$

5.2 AGGREGATION OF ATTRIBUTE SIGNATURES

By observing Table 5.1, one can notice that signatures $(\sigma_{\alpha,r}, \sigma_{\alpha,1}, \ldots, \sigma_{\alpha,L})$ all sign the same attribute α. To reduce the total number of signatures in the delegation bundle, signatures could be aggregated together into a single signature by incorporating a multi-signature scheme with signature aggregation. This allows the number of attribute signatures in a presentation token to be reduced from $n * (L+1)$ to n.

Considering the relationship between attribute signatures and R-values, the aggregated attribute signatures will depend on the R-values of all delegators. Table 5.2 depicts the relations between the signatures and R-values after signature aggregation.

	α_1	α_2	α_3	R	ϕ
root				R_1	$\phi_{r\to D_1}$
D_1	σ_{α_1}	σ_{α_2}	σ_{α_3}	R_2	$\phi_{D_1\to D_2}$
D_2				R_3	$\phi_{D_2\to D_3}$
D_3				R_4	$\phi_{D_3\to D_4}$

Table 5.2: Aggregated signatures in a delegation bundle: Relation between signatures and R-values.

5.3 DELEGATION CHAIN

Delegators will generate a delegation signature ϕ on the public key of its delegatee, in order to construct the delegation chain. The public key of the delegatee will be the message in the signature. Signature schemes based on bilinear pairings require that the message and the verification key should be group elements from the opposite group in the pairing. When we consider asymmetric Type-3 bilinear pairings, it is practically impossible to transform an element from $\mathbb{G}_1$ to an element in $\mathbb{G}_2$. Therefore it is not possible to directly use the message from one signature as the verification key of another.

There are two potential ways to solve this issue in order to construct the delegation chain:

Double Key Pair The first method involves public-key pairs which are group elements from both $\mathbb{G}_1$ and $\mathbb{G}_2$. Both public keys hide the same secret key x. The public key in $\mathbb{G}_1$ will be signed by its successor, while the public key in $\mathbb{G}_2$ acts as verification key. The root entity and the delegators generate their own secret key $sk_j := x \xleftarrow{R} \mathbb{Z}_q$ and public keys $pk_j := g_1^x \in \mathbb{G}_1$ and $\tilde{pk}_j := g_2^x \in \mathbb{G}_2$. A simplified delegation chain will look like this:

$$
\begin{aligned}
e\left(\phi_{D_1\to D_2}, g_2\right) &= e\left(pk_2^{sk_1}, g_2\right) = e\left(pk_2, \tilde{pk}_1\right)\\
e\left(\phi_{D_2\to D_3}, g_2\right) &= e\left(pk_3^{sk_2}, g_2\right) = e\left(pk_3, \tilde{pk}_2\right)\\
e\left(\phi_{D_3\to D_4}, g_2\right) &= e\left(pk_4^{sk_3}, g_2\right) = e\left(pk_4, \tilde{pk}_3\right)\\
e\left(\phi_{D_4\to D_5}, g_2\right) &= e\left(pk_5^{sk_4}, g_2\right) = e\left(pk_5, \tilde{pk}_4\right).
\end{aligned}
$$

The delegation signatures form a chain, since the public key in $\mathbb{G}_1$ in one delegation signature relates to the public key in $\mathbb{G}_2$ in the next signature. The disadvantage of this approach is that the verifier also needs to verify if the public-key pairs are constructed with the same private key. Therefore, the verifier additionally need to check $e(pk_j, g_2) \stackrel{?}{=} e(g_1, \tilde{pk}_j)$ for every public-key pair in the presentation token.

Group Alternation The second method involves alternating between groups for each delegation level. Depending on the delegation level, the delegator will sign messages in either $\mathbb{G}_1$ or $\mathbb{G}_2$. The delegation process will therefore differ for odd- and even-levels delegations. This method allows the message in one delegation to become the verification key in the next delegation. The root and all delegators generate secret keys $sk_j := x_1 \xleftarrow{R} \mathbb{Z}_q$ and $\tilde{sk}_j := x_2 \xleftarrow{R} \mathbb{Z}_q$, and public keys $pk_j := g_1^{x_1} \in \mathbb{G}_1$ and $\tilde{pk}_j := g_2^{x_2} \in \mathbb{G}_2$. A simplified delegation chain will look like this:

$$
\begin{aligned}
e\left(\phi_{D_1 \to D_2}, g_2\right) &= e\left(pk_2^{sk_1}, g_2\right) = e(\boxed{pk_2}, \tilde{pk}_1) \\
e\left(g_1, \phi_{D_2 \to D_3}\right) &= e\left(g_1, \tilde{pk}_3^{sk_2}\right) = e(\boxed{pk_2}, \boxed{\tilde{pk}_3}) \\
e\left(\phi_{D_3 \to D_4}, g_2\right) &= e\left(pk_4^{sk_3}, g_2\right) = e(\boxed{pk_4}, \boxed{\tilde{pk}_3}) \\
e\left(g_1, \phi_{D_4 \to D_5}\right) &= e\left(g_1, \tilde{pk}_5^{sk_4}\right) = e(\boxed{pk_4}, \tilde{pk}_5) .
\end{aligned}
$$

The downside of group alternation is that signatures signing the same message cannot be aggregated into one single signature. Attribute signatures signed by odd-level delegators can be aggregated together, and likewise for signatures signed by even-level delegators. Therefore L attribute signatures on the same message can be reduced to 2 multi-signatures, one in $\mathbb{G}_1$ and one in $\mathbb{G}_2$. In addition, it also requires twice as many pairings during the verification of the attribute signatures in comparison to one aggregated signature.

Because this method requires alternating groups on each delegation level, it requires two hash functions to hash group elements to both $\mathbb{G}_1$ and $\mathbb{G}_2$. Only Type-3 bilinear pairings are compatible with hashing to both $\mathbb{G}_1$ and $\mathbb{G}_2$ [48].

In Table 5.3, both methods are put side by side to compare the number of verification pairing and the number of group elements in a credential. One can observe that *Double Key Pair* has a smaller credential size than *Group Alternation*. Comparing the number of verification pairings, the *Double Key Pair* method grows harder when the number of delegations grows (due to two additional pairings per

entity in the chain to verify the key pair), while the *Group Alternation* grows harder when the number of attributes grows.

In practice, we expect that the number of attributes will be larger than the number of delegation. When we consider a scenario where $L = 3$ and $n = 10$, then *Double Key Pair* will result in 35 verification pairings, while *Group Alternation* has 49 requires verification pairings.

Considering our expectation of the number of attributes versus the number of delegations, the credential size, and the additional complexity of alternating between groups, the *Double Key Pair* method is preferred as construction for the delegation chain.

	Double Key Pair	Group Alternation
Bilinear group type	Type-2 & Type-3	Type-3
Verification pairings	$5L + 2n$	$3L + 4n$
Attribute signatures	$n\ \mathbb{G}_1$	$n\ \mathbb{G}_1 + n\ \mathbb{G}_2$
Delegation signatures	$L\ \mathbb{G}_1$	$\frac{1}{2}\ L\ \mathbb{G}_1 + \frac{1}{2}\ L\ \mathbb{G}_2$
Randomization elements	$L\ \mathbb{G}_2$	$\frac{1}{2}\ L\ \mathbb{G}_1 + \frac{1}{2}\ L\ \mathbb{G}_2$
Public keys	$L\ \mathbb{G}_1 + L\ \mathbb{G}_2$	$L\ \mathbb{G}_1 + L\ \mathbb{G}_2$
Total number of elements	$2L + n\ \mathbb{G}_1$ $2L\ \mathbb{G}_2$	$2L + n\ \mathbb{G}_1$ $2L + n\ \mathbb{G}_2$

Table 5.3: Comparison between the two delegation chain methods. The total number of elements in credentials is evaluated for both methods.

5.4 RANDOMIZABILITY OF THE SIGNATURES

As introduced before, the signatures should be randomized with the same r, as they can be linked together and can be verified with the same group element R. In order to realize multi-show credentials without linkability between presentations, the signature schemes need to maintain randomizability. Therefore, it should be possible to re-randomize the signatures and the R-value in the same fashion, such that the verification of the signatures will still be correct.

There are four potential methods to randomize elements, while preserving correctness during verification using the same pairings:

Inner-pairing Exponentiation Choose random elements $X \xleftarrow{R} \mathbb{G}_1$ and $Y \xleftarrow{R} \mathbb{G}_2$. When the elements are paired together, e.g. $e(X, Y)$, elements X and Y can be randomized by choosing a random $r \xleftarrow{R} \mathbb{Z}_q$, and by computing $X' \leftarrow X^r$ and $Y' \leftarrow Y^{1/r}$. The result of the pairing will not change, since $e(X', Y') = e(X^r, Y^{1/r}) = e(X, Y)^{r \cdot (1/r)} = e(X, Y)$.

Outer-pairing Exponentiation (Opposite groups) Pick $X \in \mathbb{G}_1, Y \in \mathbb{G}_2$, where the equation $e(X, \cdot) = e(\cdot, Y)$ holds. Both elements can be randomized by choosing $r \xleftarrow{R} \mathbb{Z}_q$ and by computing $X' = X^r$ and $Y' = Y^r$. Verification of both elements still holds, because

$$\begin{aligned} e(X', \cdot) &= e(\cdot, Y') \\ e(X^r, \cdot) &= e(\cdot, Y^r) \\ e(X, \cdot)^r &= e(\cdot, Y)^r , \end{aligned}$$

which is still correct.

Outer-pairing Exponentiation (Same group) Pick $X, Y \in \mathbb{G}_1$, where the equation $e(X, \cdot) = e(Y, \cdot)$ holds. Then we can randomize both elements by choosing $r \xleftarrow{R} \mathbb{Z}_q$ and computing $X' = X^r$ and $Y' = Y^r$. Verification of both elements still holds, since

$$\begin{aligned} e(X', \cdot) &= e(Y', \cdot) \\ e(X^r, \cdot) &= e(Y^r, \cdot) \\ e(X, \cdot)^r &= e(Y, \cdot)^r , \end{aligned}$$

which is still correct. This is also applicable for two group elements in $\mathbb{G}_2$ instead of $\mathbb{G}_1$.

Multiplication Pick $a \xleftarrow{R} \mathbb{Z}_q$, $h \xleftarrow{R} \mathbb{G}_1$, $X \leftarrow h^a \in \mathbb{G}_1$ and $Y \leftarrow g_2^a$. The elements can be verified by computing $e(X, g_2) = e(h, Y)$. X and Y can be randomized by choosing a random $r \xleftarrow{R} \mathbb{Z}_q$, and computing $X' \leftarrow X \cdot h^r$ and $Y' \leftarrow Y \cdot g_2^r$. The verification of the pairings will still hold, because

$$\begin{aligned} e(X', g_2) &= e(h, Y') \\ e(X \cdot h^r, g_2) &= e(h, Y \cdot g_2^r) \\ e(h^{a+r}, g_2) &= e(h, g_2^{a+r}) , \end{aligned}$$

which is still correct.

Randomizing elements by exponentiation is simple since the randomizer only requires the knowledge of the elements X and Y. In the case of multiplication, the knowledge of the base elements used in X and Y is also required. Therefore the multiplication method is only possible when the base elements are known by the randomizer.

As discussed in Section 5.2, the aggregated signature need to be verified with the elements $R_1, \ldots, R_{L+1}$ during verification. Aggregate signatures are verified with the aggregated randomization element R, which becomes:

$$R = R_1 \cdots R_{L+1} = g_2^{r_1} \cdots g_2^{r_{L+1}} = g_2^{r_1 + \ldots + r_{L+1}} .$$

Notice here that the exponents are added together in the aggregated element when group elements are multiplied. When signatures are aggregated, the aggregated signature will become

$$\sigma \leftarrow \sigma_1 \cdots \sigma_{L+1} = X^{r_1} \cdots X^{r_{L+1}} = X^{r_1 + \ldots + r_{L+1}} ,$$

where $X \in \mathbb{G}_1$. The secret keys of the signers are left out of the signature for the purpose of demonstration and clarity. The aggregate signature can be verified by checking

$$e(\sigma, g_2) \stackrel{?}{=} e(X, R)\,. \tag{5.1}$$

The *inner-pairing exponentiation* method and the *outer-pairing exponentiation* methods all randomize elements by exponentiation. Aggregated signatures from these methods can be verified using Equation 5.1. A problem arises when a randomization element $R_j \in R_1, \ldots, R_{L+1}$ need be randomized after signature aggregation. The aggregated signature should be randomized accordingly as well, such that the signature will be still correct during verification. Considering randomization by *exponentiation*, the randomizer chooses $r_x \xleftarrow{R} \mathbb{Z}_q$. The randomization element then becomes $R'_j \leftarrow R_j^{r_x} = g_2^{r_j \cdot r_x}$. The aggregated randomization element becomes

$$R = R_1 \cdots R_j \cdots R_{L+1} = g_2^{r_1} \cdots g_2^{\mathbf{r_j} \cdot \mathbf{r_x}} \cdots g_2^{r_{L+1}} = g_2^{r_1 + \ldots + \mathbf{r_j} \cdot \mathbf{r_x} + \ldots + r_{L+1}}\,.$$

However, it is not possible to randomize the aggregated signature in such a way that the exponent of the aggregated signature corresponds to the exponent of the randomization element, without the knowledge of the individual signatures. Therefore randomization by exponentiation is not suitable in combination with signature aggregation.

Considering the *multiplication* method, the delegator randomizes a signature by computing $\sigma'_i \leftarrow \sigma_i \cdot X^{r_i}$ for a random $r_i \xleftarrow{R} \mathbb{Z}_q$. After aggregating the signatures together, σ becomes $X^{r_1 + \cdots + r_L}$. This signature can be verified by checking $e(\sigma, g_2) = e(X, R_1 \cdots R_L)$. When a randomizer wants randomize element $R_j = g_2^{r_j}$, he chooses $r_x \xleftarrow{R} \mathbb{Z}_q$, computes $R'_j \leftarrow R_j \cdot g_2^{r_x} = g_2^{r_j + r_x}$ and randomized the aggregated signature $\sigma' \leftarrow \sigma \cdot X^{r_x}$. The signature can be correctly verified by checking

$$\begin{aligned} e(\sigma', g_2) &= e(X, R_1 \cdots R_L) \\ e(m^{r_1 + \ldots + \mathbf{r_j} + \ldots + r_L} \cdot m^{\mathbf{r_x}}, g_2) &= e(m, g_2^{r_1 + \ldots + (\mathbf{r_j} + \mathbf{r_x}) + \ldots + r_L}) \\ e(m^{r_1 + \ldots + \mathbf{r_j} + \mathbf{r_x} + \ldots + r_L}, g_2) &= e(m, g_2^{r_1 + \ldots + (\mathbf{r_j} + \mathbf{r_x}) + \ldots + r_L})\,. \end{aligned}$$

This randomization method therefore supports signature aggregation.

To conclude, the signature scheme needs to support the multiplication randomization method to incorporate both aggregation and randomizability into attribute signatures.

5.5 ANONYMOUS CREDENTIALS

To issue a credential to a prover, the issuing delegator signs the public key of the prover and assigns attribute values to the corresponding

attributes. The prover will first create commitments over the attribute values of the to-be-issued attributes in the credential. The prover will send the commitments to the issuing verifier and carries out a (non-) interactive zero-knowledge proof of knowledge of the opening of the commitments.

When the issuing delegator is convinced, the delegator will sign the commitment and send the commitment signatures back to the prover. The prover will be able to prove that he has a signature over a commitment which opens to a particular value, without revealing the signature nor the commitment. This also allows proofing predicates over the committed values during the presentation to a verifier.

Table 5.4 extends Table 5.2 by adding commitment signatures ω_{α_i} for each corresponding attribute signature σ_{α_i}. The value signatures will be linked to the attribute signatures by the respective R_{α_i}-value and the R_P-value of the issuing delegator. This results in the structure of a credential which will be issued to prover P.

	α_1	α_2	α_3	R	ϕ
root				R_1	$\phi_{r \to D_1}$
D_1	σ_{α_1}	σ_{α_2}	σ_{α_3}	R_2	$\phi_{D_1 \to D_2}$
D_2				R_3	$\phi_{D_2 \to D_3}$
D_3	ω_{α_1}	ω_{α_2}	ω_{α_3}	R_P	$\phi_{D_3 \to P}$
	R_{α_1}	R_{α_2}	R_{α_3}		

Table 5.4: Signatures in a credential: Relation between signatures and R-values.

Since the commitment signature ω_{α_i} and the attribute signature σ_{α_i} are linked by R_{α_i}, the commitment signature scheme for the values must also support randomization by multiplication.

5.6 GROUP ELEMENT PROOF OF KNOWLEDGE IN PAIRINGS

The verifier would need to obtain all the group elements from the credential to be able to compute the verification pairings, in order to verify the validity of the credential from a prover. However, this is not desired, since the verifier learns the identities of the delegators and the prover from the public keys in the credential. Therefore, it is preferred for the prover to prove that he has the knowledge over the signatures and public keys, without disclosing the actual group elements from the credential to the verifier.

Pairing-based zero-knowledge proofs (e.g., Groth-Sahai) discussed in Section 3.4.2 require expensive pairing operations and are not efficient enough for practical use. To avoid more pairing operations,

we incorporate Schnorr's zero-knowledge proofs of knowledge [67]. Instead of proofing the knowledge of a secret exponent, the same technique can also be applied to proving the knowledge of a group element in a pairing. The Fiat-Shamir heuristic [40] will be used to transform the interactive proof to a non-interactive one.

Given a pairing equation $z = e(X, Y)$, where $X \in \mathbb{G}_1$ and $Y \in \mathbb{G}_2$, a prover can prove the knowledge of element X non-interactively without disclosing the actual group element. The prover computes chooses $R \xleftarrow{R} \mathbb{G}_1$, and computes $t \leftarrow e(R, Y)$, $c \leftarrow H(\cdots) \in \mathbb{Z}_q$ and $S \leftarrow R \cdot X^c$. The prover sends the non-interactive proof (t, c, S) to the verifier. The verifier checks the proof by computing $\hat{t} \leftarrow e(S, Y) \cdot z^{-c}$ and check if the Fiat-Shamir hash is equal to c.

As discussed before, delegators will generate a signature ϕ on the public key of the delegatee to form a delegation chain. In this case, the public key of the delegatee will be the message in the signature. During verification, the verifier uses the public key of the delegatee as verification key for the next signature in the delegation chain.

Notice that in the delegation signature ϕ, both the message element and the verification key element need to be zero-knowledge proved. Both proofs also need to be used to verify the adjacent delegation signatures in the chain. Since the two group elements in the same pairing cannot be both zero-knowledge proved at the same time, the delegation signature ϕ, the message, and the verification key need to be verified in separate pairings in the verification equation.

5.7 CONCLUSION / KEY TAKEAWAYS

We have discussed the generic construction of our protocol, and shown which requirements are needed to realize the protocol.

There are three signatures schemes required as primitive, in order to realize a concrete construction and instantiation of the protocol.

The first signature scheme allows delegators to sign attributes which they wish to delegate to a delegatee. Signatures signing the same message can be aggregated together into a single multi-signature to reduce the total number of signatures.

The second signature scheme enables a delegator to delegate signing rights by signing the public key of the delegatee as the message of the signature. To incorporate Schnorr's zero-knowledge proofs of knowledge, the delegation signature, the message and the verification key need to be verified in separate pairings in the verification equation.

The last commitment signature scheme allows issuers to sign commitments created by provers. The provers can convince verifiers

that they have knowledge over a commitment signature signed by the issuer, and prove predicates over the committed values.

The three signature schemes need to be randomized in the same manner to link the generated signatures together. The related signatures can then be verified with the same randomization element R. As discussed and concluded in Section 5.4, the multiplication method of randomization is the most suitable for our scenario. Therefore, all three signatures schemes need to adopt this method of randomization.

6

PRELIMINARY SIGNATURE SCHEMES

In Chapter 5, we discussed the design of the protocol and concluded with the desired properties for the three signature scheme required to create a concrete construction and instantiation of the protocol.

To the best of our knowledge, none of the existing signature schemes offers the desired properties that we need nor could they be adapted to our needs with ease. Therefore, we propose three new signature schemes, inspired by existing signature schemes. These signature schemes will form the primitives for the protocol construction and instantiation in the following chapters.

In Section 6.2, we will introduce the first signature scheme *RBLS: Randomizable BLS Multi-Signatures with Public-Key Aggregation*. The second signature scheme *RSPS-M: Randomizable Structure-Preserving Signatures by Multiplication* will be introduced in Section 6.3. The last commitment signature scheme *SRS-M: Short Randomizable Signatures by Multiplication* will be introduced in Section 6.4.

The definitions and details of the existing primitive signature schemes on which the new schemes build upon can be found in Appendix A.

6.1 RANDOMIZATION BY MULTIPLICATION

As introduced in the previous chapter, we will incorporate randomizability and linkability of signatures through multiplication of the base elements exponentiated with a random scalar. We propose Assumption 8, which can be reduced to the discrete logarithm problem (Assumption 1).

Assumption 8 (Computational Linear Problem): Let $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a Type-3 bilinear group setting, with g_1 and g_2 as generators of respectively $\mathbb{G}_1$ and $\mathbb{G}_2$. Given $(g_1^{r+s}, g_2^r, g_2^s)$ for random $r, s \in \mathbb{Z}_q$, it is hard to compute g_1^r or g_1^s in polynomial-time.

Notice that deciding whether the elements hide the same exponents is easy, since one could check

$$e(g_1^{r+s}, g_2) \stackrel{?}{=} e(g_1, g_2^r \cdot g_2^s)\,.$$

This assumption only holds for Type-3 bilinear maps, where no efficiently computable homomorphisms exist between $\mathbb{G}_1$ and $\mathbb{G}_2$. When such an efficiently computable homomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ would exist (Type-2 pairings), then one can compute $\psi(g_2^r) = g_1^r$ with ease. Assumption 8 does also not hold in symmetric groups.

Given $(g^a, g^b, g^{ra+sa}, g^{rb}, g^{sb})$ for random $a, b, r, s \in \mathbb{Z}_q$, it is easy to compute g^{ra} or g^{sa} in polynomial-time, since

$$g^{rb}/g^b \cdot g^a = g^{ra} \quad \text{and} \quad g^{sb}/g^b \cdot g^a = g^{sa}\,.$$

6.1.1 *Security Proof*

We will prove that Randomization by Multiplication does not compromise the security of existing schemes and achieves both randomization and linkability to other signatures.

Theorem 1. *Randomizability by Multiplication achieves perfect randomizability of group elements.*

Proof. Let $\tilde{h} \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$. Compute signature $\sigma \leftarrow \tilde{h}^x$ and verification element $V \leftarrow h^x$. The signature can be verified as follows:

$$e(\sigma, h) \stackrel{?}{=} e(\tilde{h}, V)$$
$$e(\tilde{h}^x, h) \stackrel{?}{=} e(\tilde{h}, h^x)\,.$$

Randomization by multiplication works as follows. The randomizer chooses a random $r \stackrel{R}{\leftarrow} \mathbb{Z}_q$, and randomize the signature by using the known base element $\tilde{h}$, exponentiate the element with r and multiplies it with the signature σ. This results in $\sigma' \leftarrow \sigma \cdot \tilde{h}^r = \tilde{h}^x \cdot \tilde{h}^r = \tilde{h}^{x+r}$. The randomizer also generates a randomization element $R \leftarrow h^r$. The randomized signature can be verified by checking

$$e(\sigma', h) \stackrel{?}{=} e(\tilde{h}, V \cdot R)$$
$$e(\tilde{h}^{x+r}, h) \stackrel{?}{=} e(\tilde{h}, h^x \cdot h^r)\,.$$

Since r is random, the randomized signature $\tilde{h}^{x+r}$ is also random, and does not correlate to the existing signature $\tilde{h}^x$. The signature can be re-randomized in the same fashion, and can be verified by either generating an additional randomization element or by randomizing the existing randomization element. □

Theorem 2. *Randomization by Multiplication allows perfect linkability among signatures from diverse schemes.*

Proof. An adversary might want to remove or alter the relationship between existing signatures and randomization elements since it would enable them to link the valid signature to other signatures. Given Assumption 8, it would not be possible for an adversary to remove or alter the randomization from signatures without the knowledge of r, $\tilde{h}^r$ or $\tilde{h}^x$. Inherently, the adversary cannot link new signatures to existing randomization elements without the

knowledge of the hidden exponent r. Therefore, the randomization scheme prevents adversaries to (1) remove the link of signatures to the given randomization element; and (2) prevent new signatures to be linked to existing randomization elements. □

6.2 RBLS: RANDOMIZABLE BLS MULTI-SIGNATURES WITH PUBLIC-KEY AGGREGATION

Boneh, Lynn, and Shacham (BLS) [15] came up with a signature scheme, which allows the aggregation of multiple signatures into a single signature. The aggregated signature can be efficiently verified with just two pairings instead of $2n$ and reduces in total signature size. The limitation of this scheme is that all messages from the individual signatures have to be distinct for the aggregated signature to be secure. This is due to the possibility of a *rogue public key attack*, where an adversary could craft aggregated signatures with known public keys and claim that others have signed the maliciously crafted signature as well. Details of the scheme and the attack can be found in Appendix Section A.1.

6.2.1 *Random Oracle Public-Key Dependency*

Boneh, Drijvers and Neven [14] came up with an improvement which prevents rogue public key attacks while having the ability to combine signature signing the same message. The *BLS Multi-Signatures with Public-Key Aggregation* (MSP) scheme incorporates the public keys of the signers as the input of a random oracle, where the output will be used to exponentiate each signature which will be aggregated. The output of the random oracle will be different for every signature, which prevents the rogue public key attack. Therefore, multiple signatures signing the same message can be aggregated together. Details of the scheme can be found in Appendix Section A.2.

Nevertheless, the public keys would also be required as input to the random oracle during verification. This is not ideal when a prover does not want to directly reveal the public keys involved in the signatures but instead wishes to provide zero-knowledge proofs about the knowledge over the signatures and the public keys. Without concrete public keys, it becomes impossible for the verifier to compute the same hash values as computed during signature aggregation.

In addition, the MPS scheme is designed for the aggregation of n signatures at the same time. Therefore, the scheme computes the exponentiation of each public key based on the all public keys in the aggregated signature. When the group composition changes, all public key exponentiations will be different. While letting the

exponentiation depend on the group composition offers better protection against forgery, it does not support sequential aggregation, where signatures are added one-by-one to the aggregated signature.

In our scheme, we incorporate a seed element $s \leftarrow \mathbb{G}_2$ and a counter value $c \leftarrow \mathbb{Z}_q$ as input for the random oracle. The seed is a publicly known element in $\mathbb{G}_2$, and the counter value increments for each signature which will be aggregated. Signatures can be either aggregated in one time:

$$\sigma \leftarrow \prod_{i=1}^{n} \sigma_i^{H(seed||i)} ,$$

or sequentially:

$$\sigma' \leftarrow \sigma \cdot \sigma_i^{H(seed||i)} .$$

This allows the verifier to compute an aggregated public key

$$apk \leftarrow \prod_{i=1}^{n} pk_i^{H(seed||i)}$$

for verification of the aggregated signature. Similar to MSP, the rogue public key attack is mitigated due to the exponentiation of different hash values to every public key.

6.2.2 *Randomizability of (Aggregated) Signatures*

The *BLS Multi-Signatures with Public-Key Aggregation* scheme does not incorporate the possibility to randomize signatures. We extend this scheme by adding randomizability by multiplication. The individual signatures σ_i are randomized with a random $r_i \xleftarrow{R} \mathbb{Z}_q$ and can be verified using the element $R_i \leftarrow g_2^{r_i} \in \mathbb{G}_2$. The randomized signature becomes $\sigma_i' \leftarrow \sigma_i \cdot H_1(\alpha)^{r_i}$. When the signatures are aggregated as before to σ, the aggregated signature can be verified with the element $R \leftarrow \prod_{i=1}^{n} R_i \in \mathbb{G}_2$ in the equation:

$$e(\sigma, g_2) = e(H_1(\alpha), apk \cdot R) .$$

An essential property in our scheme is the ability to re-randomize R_i and σ, such that the randomized aggregated signature σ' can be verified with the randomized R_i'. In our scheme, the elements will be randomized by choosing $r_x \xleftarrow{R} \mathbb{Z}_q$, and computing $R_i' \leftarrow R_i \cdot g_2^{r_x} \in \mathbb{G}_2$ and $\sigma' \leftarrow \sigma \cdot H_1(m)^{r_x} \in \mathbb{G}_1$.

6.2.3 *Instantiation*

Like the two previous *BLS-Signature* schemes, we use a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are of prime order q,

and the pairing is non-degenerate and efficiently computable. Let g_1 and g_2 be generators for $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Also we incorporate two hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1$ and $H : \{0,1\}^* \to \mathbb{Z}_q$, which are viewed as random oracles [8].

6.2.3.1 *Single Signature*

KeyGen(): Choose random $x \xleftarrow{R} \mathbb{Z}_q$ and set $h \leftarrow g_2^x \in \mathbb{G}_2$. Output $pk := h$ and $sk := x$.

Sign(sk, m, L): Choose random $r \xleftarrow{R} \mathbb{Z}_q$. Output $R = g_2^r \in \mathbb{G}_2$ and $\sigma = H_1(m)^{x \cdot H(seed||L)+r} \in \mathbb{G}_1$.

Verify(pk, m, L, σ, R): If $e(\sigma, g_2) = e(H_1(m), pk^{H(seed||L)} \cdot R)$ output 1 otherwise 0.

Randomize(m, σ, R): Choose random $r \xleftarrow{R} \mathbb{Z}_q$. Output $R' = R \cdot g_2^r \in \mathbb{G}_2$ and $\sigma' = \sigma \cdot H_1(m)^r \in \mathbb{G}_1$.

6.2.3.2 *Multi Signature*

When $\sigma_1, \ldots, \sigma_n$ sign the same message m, the signatures can be aggregated and verified with two pairings.

Aggregate$(\langle \sigma_i, R_i \rangle_{i=1}^n)$: Output: $\sigma \leftarrow \prod_{i=1}^n \sigma_i$ and $R \leftarrow \prod_{i=1}^n R_i$.

AggVerify$(\langle pk_i \rangle_{i=1}^n, m, \sigma, R)$: Compute $apk \leftarrow \prod_{i=1}^n pk_i^{H(seed||i)}$. If $e(\sigma, g_2) = e(H_1(m), apk \cdot R)$ output 1 otherwise 0.

AggRandomize(m, σ, R): Choose random $r \xleftarrow{R} \mathbb{Z}_q$. Output $R' = R \cdot g_2^r \in \mathbb{G}_2$ and $\sigma' = \sigma \cdot H_1(m)^r \in \mathbb{G}_1$.

6.2.3.3 *Batch Signature*

n multi-signatures can be aggregated together and batch verified with $(n+1)$ pairings.

BatchAggregate$(\langle \sigma_i \rangle_{i=1}^n)$: Output $\sigma \leftarrow \prod_{i=1}^n \sigma_i$.

BatchVerify$(\langle apk_i, R_i, m_i \rangle_{i=1}^n, \sigma)$:
If $e(\sigma, g_2) = e(H_1(m_1), apk_1 \cdot R_1) \cdots e(H_1(m_n), apk_n \cdot R_n)$ output 1 otherwise 0.

6.2.3.4 *Batch signature from the same signers*

When $apk_1 = \cdots = apk_n$, and $R_1 = \cdots = R_n$, then the number of pairings for verification could be even further reduced to two.

BatchVerify2$(apk, R, \langle m_i \rangle_{i=1}^n, \sigma)$:
If $e(\sigma, g_2) = e(H_1(m_1) \cdots H_1(m_n), apk \cdot R)$ output 1 otherwise 0.

BatchRandomize2$(\langle m_i \rangle_{i=1}^n, \sigma, R)$: Choose random $r \xleftarrow{R} \mathbb{Z}_q$. Output $R' = R \cdot g_2^r \in \mathbb{G}_2$ and $\sigma' = \sigma \cdot \left(\prod_{i=1}^n H_1(m_i)\right)^r \in \mathbb{G}_1$.

6.2.4 *Security Proof*

We will show that our modification to the *BLS Multi-Signatures with Public-Key Aggregation* scheme does not affect the security of the original scheme.

Theorem 3. *RBLS prevents rogue public key attacks.*

Proof. Assume an adversary wishes to execute a rogue public key attack on signatures in *RBLS*. The adversary computes

$$\begin{aligned} e(H_1(\alpha)^{x_r}, g_2)^{H(seed||1)} &= e(H_1(\alpha), pk_r^{H(seed||1)}) \\ &= e(H_1(\alpha), (g_2^{x_r})^{H(seed||1)}) \\ &= e(H_1(\alpha), (g_2^{x_1})^{H(seed||1)} \cdot (g_2^{x_r - x_1})^{H(seed||1)}) \\ &= e(H_1(\alpha), pk_1^{H(seed||1)} \cdot pk_2^{H(seed||1)}) \,. \end{aligned}$$

Although the adversary is still able to compute $pk_2 \leftarrow pk_r / pk_1$, the forged signature worthless since pk_2 will be exponentiated by the verifier with a different hash value $H(seed||2)$ instead of $H(seed||1)$. The adversary does not control the public key exponentiation during verification and therefore the rogue key attack is ineffective.

Even though the exponentiation values are the same for public keys across the same delegation level, an adversary still cannot perform signature forgery which passes verification, since the exponentiation values differ for each public key in the delegation chain. □

Theorem 4. *The addition of the Randomization by Multiplication method in RBLS does not affect the security of the underlying MSP scheme.*

Proof. RBLS randomizes signatures by using the base element $H_1(m)$, which is already known to all entities. The entity chooses a random r to establish relationships between RBLS and other signatures. As shown in Section 6.1.1, we have proven that randomizing elements using the Randomization by Multiplication method is secure under the Computational Linear Assumption (Assumption 8). Even when this assumption appears to be easy, it only affects the linkability and randomization property of the signature scheme and does not influence the security of the existing signature properties. □

6.3 RSPS-M: RANDOMIZABLE STRUCTURE-PRESERVING SIGNATURES BY MULTIPLICATION

In order to construct our protocol for the consecutive delegation of signing rights, we need a signature scheme to be randomizable in the same way as the *RBLS: Randomizable BLS Multi-Signatures with Public-Key Aggregation* scheme introduced and discussed in Section 6.2. Therefore we need to randomize the signature in such a way,

that signatures can be verified with the element $R \leftarrow g_2^r$, and be randomized with $R' \leftarrow R \cdot g_2^{r'}$. Also, the signature, the message, and the verification key elements need to occur in different pairings in the verification equation as concluded in Section 5.7.

Groth [52] introduced a structure-preserving signature scheme, which verifies the message and the verification key in different pairings during verification. However, the scheme incorporates inner-pairing randomizability, which prevent the signatures from to be aggregated as discussed in Section 5.4. Details on the structure-preserving signature scheme by Groth can be found in Appendix Section A.3.

Inspired by the construction of the structure-preserving signature scheme of Groth, we propose *RSPS-M: Randomizable Structure-Preserving Signatures by Multiplication*, where the signature is randomizable by the multiplication method, and the signature, the message, and the verification key elements are verified in different pairings during verification.

6.3.1 *Instantiation*

We use a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are of prime order q, and the pairing is non-degenerate and efficiently computable. Let g_1 and g_2 be generators for $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. We assume that the setup will be done by an honest party.

$\mathsf{Setup}()$: Choose random $y_1, y_2 \xleftarrow{R} \mathbb{G}_2$. Output: y_1, y_2.

$\mathsf{KeyGen}()$: Choose random $x \xleftarrow{R} \mathbb{Z}_q$ and set $h \leftarrow g_2^x \in \mathbb{G}_2$. Output: $pk := h$ and $sk := x$.

$\mathsf{Sign}(sk, m)$: Output: Choose random $r \xleftarrow{R} \mathbb{Z}_q$. Output $R = g_2^r \in \mathbb{G}_2$ and $\phi = y_1^{sk} \cdot (y_2 \cdot m)^r \in \mathbb{G}_1$.

$\mathsf{Verify}(pk, m, \phi, R)$: If $e(\phi, g_2) = e(y_1, pk) \cdot e(y_2 \cdot m, R)$ output 1 otherwise 0.

$\mathsf{Randomize}(m, \phi, R)$: Choose random $r \xleftarrow{R} \mathbb{Z}_q$. Output $R' = R \cdot g_2^r \in \mathbb{G}_2$ and $\phi' = \phi \cdot (y_2 \cdot m)^r \in \mathbb{G}_1$.

6.3.2 *Security Proof*

Definition 1 (EUF-CMA). *The RSPS-M scheme* (Setup, KeyGen, Sign, Verify, Randomize) *with message space* $(\mathbb{G}_1^*)$ *is* existentially unforgeable under adaptively chosen-message attacks *if for all probabilistic*

polynomial-time adversaries $\mathcal{A}$ having access to a signing oracle $\mathsf{Sign}(sk, \cdot)$*, we have*

$$\Pr\left[\begin{array}{l|l} \Lambda \leftarrow \mathsf{Setup}(); & \\ (sk, pk) \leftarrow \mathsf{KeyGen}(); & M^* \notin Q \wedge \\ (M^*, \phi^*, R^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(sk,\cdot)} & \mathsf{Verify}(pk, M^*, \phi^*, R^*) = 1 \end{array}\right] \leq \epsilon(\kappa)$$

where Q is the set of queries which $\mathcal{A}$ made to the signing oracle.

Theorem 5. *The RSPS-M scheme is resistant against* existential unforgeability under chosen message attacks *(Definition 1).*

Proof. Assume adversaries $\mathcal{A}$ have the ability send queries to the oracle $\mathcal{O}^{Sign(sk,\cdot)}$ in polynomial time, where oracle outputs valid signatures for the chosen messages. The adversaries could compute the following for each two signatures:

$$(\phi_1, R_1) \leftarrow \mathcal{O}^{Sign(sk,m_1)}$$
$$(\phi_2, R_2) \leftarrow \mathcal{O}^{Sign(sk,m_2)}$$

$$\frac{\phi_1}{\phi_2} = \frac{y_1^{sk} \cdot (y_2 \cdot m_1)^{r_1}}{y_1^{sk} \cdot (y_2 \cdot m_2)^{r_2}} = \frac{(y_2 \cdot m_1)^{r_1}}{(y_2 \cdot m_2)^{r_2}} = y_2^{r_1 - r_2} \cdot \frac{m_1^{r_1}}{m_2^{r_2}}.$$

Since the oracle used different random scalars r for each signature, the chosen messages in the signature are also random and does not give the adversaries any advantage. Assuming that r_1 and r_2 are large scalars, the adversaries are not able to compute y_1^{sk} to forge signatures.

When adversaries $\mathcal{A}$ queries the oracle for the same message m in polynomial time, then $\mathcal{A}$ would be able to compute for every two signatures:

$$(\phi_1, R_1) \leftarrow \mathcal{O}^{Sign(sk,m)}$$
$$(\phi_2, R_2) \leftarrow \mathcal{O}^{Sign(sk,m)}$$

$$\frac{\phi_1}{\phi_2} = \frac{y_1^{sk} \cdot (y_2 \cdot m)^{r_1}}{y_1^{sk} \cdot (y_2 \cdot m)^{r_2}} = \frac{(y_2 \cdot m)^{r_1}}{(y_2 \cdot m)^{r_2}} = (y_2 \cdot m)^{r_1 - r_2}.$$

$(y_2 \cdot m)^{r_1 - r_2}$ is only useful to the adversaries when they were able to obtain a signature from the oracle where the signature on m is randomized with $r = r_1 - r_2$. In addition, adversaries need to compute two pairings to decide whether $r = r_1 - r_2$:

$$e((y_2 \cdot m)^{r_1 - r_2}, g_2) \stackrel{?}{=} e((y_2 \cdot m), R_x).$$

Since r is uniformly random over $\mathbb{Z}_q$, the probability that adversaries obtain this signature in polynomial time is negligible.

We also proof that $\mathcal{A}$ are not able use the randomization property to alter the integrity of the message in the signature. When $\mathcal{A}$ wishes to perform existential forgery on a signature by randomization, then they need to compute r_2, m' and m_2 such that:

$$\begin{aligned} y_1^{sk} \cdot (y_2 \cdot m_1)^{r_1} \cdot (y_2 \cdot m_2)^{r_2} &= y_1^{sk} \cdot (y_2 \cdot m')^{r'} \\ (y_2)^{r_1} \cdot (y_2)^{r_2} \cdot (m_1)^{r_1} \cdot (m_2)^{r_2} &= (y_2)^{r'} \cdot (m')^{r'} \\ (y_2)^{r_1} \cdot (m_1)^{r_1} &= \frac{(y_2)^{r'} \cdot (m')^{r'}}{(y_2)^{r_2} \cdot (m_2)^{r_2}} . \end{aligned}$$

Since r_1, $(y_2)^{r_1}$, $(m_1)^{r_1}$ and $(y_2)^{r_1} \cdot (m_1)^{r_1}$ are unknown to the adversaries, they cannot compute r_2, m' and m_2 in polynomial time, such that the alter ranodmized signature is correct. □

6.4 SRS-M: SHORT RANDOMIZABLE SIGNATURES BY MULTIPLICATION

Most of the signature schemes for commitments use the inner-pairing or the outer-pairing methods (Section 5.4) for the randomization of their signatures [24, 64]. Therefore, we need to modify an existing signature/commitment scheme to support randomization by multiplication. For the signature scheme for commitments, we are incorporating the signature scheme by Pointcheval and Sanders [64]. More details about this signature scheme can be found in the Appendix Section A.4.

In this section, we introduce *SRS-M: Short Randomizable Signatures by Multiplication*, which allows PS-signatures to be randomized by exponentiation and multiplication in a hybrid fashion.

6.4.1 *Randomization*

The signature scheme by Pointcheval and Sanders [64] is randomizable by the outer-pairing method, where the signature and the randomization element are in the same group. The signature scheme outputs $\sigma \leftarrow (\sigma_1, \sigma_2) = (\tilde{h}^r, (\tilde{X}C)^r)$, and can be randomized by choosing a random $r \leftarrow \mathbb{Z}_q$ and computing $\sigma' \leftarrow (\sigma_1^r, \sigma_2^r)$.

In Table 5.4, we can see that commitment signature ω_{α_i} should be dependent on the randomization elements R_P and R_{α_i}. Therefore the signature scheme must also support randomization by multiplication.

In our modified scheme, we allow PS-signatures to be randomized with both exponentiation and multiplication mixed at the same time. To achieve this, we introduce two additional elements to the signature: $\pi \leftarrow (\pi_1, \pi_2)$. When the signer wants to sign commitment C with its private element $\tilde{X}$, the signer chooses $r, s \xleftarrow{R} \mathbb{Z}_q$, and computes $\sigma \leftarrow ((\tilde{h}^s)^r, ((\tilde{X}C)^s)^r)$ and $\pi \leftarrow (\tilde{h}^s, (\tilde{X}C)^s)$. Note here that $\sigma = (\pi_1^r, \pi_2^r)$. Therefore, the π elements are the base elements of σ elements.

When the signature need to be randomized by *multiplication*, the randomizer will choose $t \xleftarrow{R} \mathbb{Z}_q$ and computes:

$$\sigma_1' \leftarrow \sigma_1 \cdot \pi_1^t = \left(\tilde{h}^s\right)^r \cdot \left(\tilde{h}^s\right)^t = \left(\tilde{h}^s\right)^{r+t}$$
$$\sigma_2' \leftarrow \sigma_2 \cdot \pi_2^t = \left(\left(\tilde{X}C\right)^s\right)^r \cdot \left(\left(\tilde{X}C\right)^s\right)^t d = \left(\left(\tilde{X}C\right)^s\right)^{r+t} .$$

This results in the accumulation of the secret exponent r and t, which both resides outside of the base element. When the previously obtained randomized signature σ' needs to be randomized by *exponentiation*, the randomizer will choose $v \xleftarrow{R} \mathbb{Z}_q$ and computes:

$$\sigma_1'' \leftarrow \left(\sigma_1'\right)^v = \left(\left(\tilde{h}^s\right)^{r+t}\right)^v = \left(\tilde{h}^{sv}\right)^{r+t}$$
$$\sigma_2'' \leftarrow \left(\sigma_2'\right)^v = \left(\left(\left(\tilde{X}C\right)^s\right)^{r+t}\right)^v = \left(\left(\tilde{X}C\right)^{sv}\right)^{r+t}$$

$$\pi_1' \leftarrow \pi_1^v = \left(\tilde{h}^s\right)^v = \tilde{h}^{sv}$$
$$\pi_2' \leftarrow (\pi_2)^v = \left(\left(\tilde{X}C\right)^s\right)^v = \left(\tilde{X}C\right)^{sv} .$$

In this case, the base element accumulates the s exponent and does not affect the outer exponents of the base element.

As shown, both signatures σ_1'' and σ_2'' are still randomized with the same exponents, while the signature is now randomized with both methods. Also, π_1' and π_2' became the new base elements of σ_1'' and σ_2'' to support the next iteration of randomization by multiplication.

6.4.2 *Verifying Randomized Credentials*

The randomized PS-signatures can verified as usual, since the randomization exponents of σ_1 and σ_2 are still the same after randomization with both methods. Because the signature also incorporates randomization by multiplication, we can verify the signature with a randomization element R. Given the previous computed signature σ_1'', base element π_1' and randomization element $R \leftarrow g_2^{r+t}$, we can verify that the signature includes the secret exponents in the randomization element R:

$$e(\sigma_1'', g_2) = e(\pi_1', R) \tag{6.1}$$
$$e(\left(\tilde{h}^{sv}\right)^{r+t}, g_2) = e(\tilde{h}^{sv}, g_2^{r+t}) . \tag{6.2}$$

6.4.3 *Instantiation*

Keygen(): Choose random $\tilde{h} \xleftarrow{R} \mathbb{G}_1, h \xleftarrow{R} \mathbb{G}_2$ and $(x, y) \xleftarrow{R} \mathbb{Z}_q$. Compute $(\tilde{X}, \tilde{Y}) \leftarrow (\tilde{h}^x, \tilde{h}^y)$ and $(X, Y) \leftarrow (h^x, h^y)$, set $sk \leftarrow \tilde{X}$ and $pk \leftarrow (\tilde{h}, \tilde{Y}, h, X, Y)$.

Commit(m): Choose random $t \xleftarrow{R} \mathbb{Z}_q$, and compute $C \leftarrow \tilde{h}^t \tilde{Y}^m \in \mathbb{G}_1$. Output: (C, t).

Sign(sk, C): Choose random $r, s \xleftarrow{R} \mathbb{Z}_q$ and $\sigma' \leftarrow (\tilde{h}^{sr}, (\tilde{X}C)^{sr})$ and $\pi' \leftarrow (\tilde{h}^s, (\tilde{X}C)^s)$.

Unblind(σ', t): Parse $\sigma' = (\sigma'_1, \sigma'_2)$. Output $\sigma \leftarrow (\sigma'_1, \sigma'_2 / \sigma'^t_1)$ and $\pi \leftarrow (\pi'_1, \pi'_2 / \pi'^t_1)$.

Verify(pk, m, σ): Parse $\sigma = (\sigma_1, \sigma_2)$. If $\sigma_1 \neq 1_{\mathbb{G}_1}$ and $e(\sigma_1, X \cdot Y^m) = e(\sigma_2, h)$, output 1 otherwise 0.

Randomize(σ, π): Parse $\sigma = (\sigma_1, \sigma_2)$ and $\pi = (\pi_1, \pi_2)$. Choose random $t \xleftarrow{R} \mathbb{Z}_q$ and output $\sigma' \leftarrow (\sigma_1 \cdot \pi_1^t, \sigma_2 \cdot \pi_2^t)$

GenPoK(ω): Parse $\omega = ((\sigma_1, \sigma_2), \pi)$. Choose random $r, t \xleftarrow{R} \mathbb{Z}_q$. Compute $\sigma' \leftarrow (\sigma_1^r, (\sigma_2 \cdot \sigma_1^t)^r)$ and output (t, σ').

6.4.4 *Security Proof*

Theorem 6. *The addition of the π elements to the SRS-M scheme does not affect the security of the underlying PS-Signature scheme.*

Proof. $\pi \leftarrow (\pi_1, \pi_2)$ are in fact the base elements of the signature $\sigma \leftarrow (\sigma_1, \sigma_2)$. Because σ is in fact π exponentiated with random scalars, π itself is also a valid signature on commitment C. An adversary does not learn anything about the random exponents given σ and π, assuming the discrete logarithm problem. Therefore disclosing π does not compromise the security of the original signature scheme in any way. □

Theorem 7. *The addition of the Randomization by Multiplication method to the SRS-M scheme does not affect the security of the underlying PS-Signature scheme.*

Proof. As shown in Section 6.4.1, both randomization methods randomize the elements in $\sigma \leftarrow (\sigma_1, \sigma_2)$ in a different way, but always maintain the same random scalar in σ_1 and σ_2. The modification to the scheme does not affect the security of the signature scheme at all and operates identically to the existing protocol. □

Theorem 8. *The addition of the Randomization by Multiplication method to the SRS-M scheme enables linkability to other signatures.*

Proof. Given $\sigma \leftarrow \left(\tilde{h}^{sv}\right)^{r+t}$, $\pi \leftarrow \tilde{h}^{sv}$ and $R \leftarrow g_2^{r+t}$ derived from Equation 6.2, the verifier checks if the following holds:

$$e(\sigma_1, g_2) = e(\pi_1, R)$$
$$e(\left(\tilde{h}^{sv}\right)^{r+t}, g_2) = e(\tilde{h}^{sv}, g_2^{r+t})$$

As shown in the previous proof, σ and π do not leak the secret random exponent $r + t$ assuming the discrete logarithm problem. When an adversary wished to alter R, it also need to alter σ such that the verification equation still holds. This is only possible when the adversary randomizes σ and R according to the protocol, therefore σ and R are still linked together. The verification equation proofs that the secret random exponent $r + t$ is also hidden in the randomization element R. Since an adversary does not know the secret random exponent, it cannot link the signature to a different R values. As proven in Section 6.1.1, randomizing elements using the Randomization by Multiplication method is secure under the Computational Linear Assumption (Assumption 8). □

7

ENTRUST: CONSECUTIVE DELEGATABLE SIGNING RIGHTS FOR ANONYMOUS CREDENTIALS

In the prior chapters, we have designed the protocol for the delegation of signing rights for the issuance of attribute-based credentials, and introduced the preliminary signature schemes which are required to build such a protocol. In this chapter, we put everything together in order to realize a concrete instantiation. We propose our protocol, ENTRUST, for the consecutive delegation of signing rights for anonymous attribute-based credentials.

7.1 OVERVIEW

We will first give a high-level overview of our concrete instantiation, where we apply the preliminary signature scheme to our protocol design.

The goal of the protocol is to convince the verifiers that the presented presentation tokens are authentic and integer, without revealing the identity of the intermediate delegators and the prover. A presentation token has to prove the following:

- The delegation chain is correct, e.g $\phi_{r\to D_1}, \ldots, \phi_{D_{L-1}\to D_L}, \phi_{D_L\to P}$;
- The public key pairs are correctly constructed;
- All attributes are signed by the root and all intermediate delegators;
- The committed values correspond to the correct attributes;
- The presented predicates holds for the attribute value committed in the signed commitment.

During delegation of the signing rights of attributes, the delegator will create a signature σ_{α_i} on each attribute α_i, and a delegation signature $\phi_{D_x\to D_y}$ on the public key of the delegatee. To ensure that all these signatures are related to each other, signatures are linked together by randomizing the signatures with the same $r \xleftarrow{R} \mathbb{Z}_q$. By incorporating the same r value in each signature, all signatures can be verified the same randomization element $R \leftarrow g_2^r \in \mathbb{G}_2$. Given $(\sigma_{\alpha_1}, \ldots \sigma_{\alpha_n}, \phi_{x\to y}, R)$, an adversary cannot recover r due the assumed discrete-log problem.

The delegator which issues a credential to a prover creates a signature σ_{α_i} on each attribute α_i as before, and a commitment signature ω_{α_i} on the value v_i corresponding to α_i. The delegator will

also generate a signature $\phi_{x \to P}$ on the public key of the prover. However, this signature will be signed in a different way than delegation signatures, in order to prevent the prover to gain signing rights and issue credentials for others as a delegator. All signatures generated by the issuing delegator will be randomized with $r_P \xleftarrow{R} \mathbb{Z}_q$. Additionally, the delegator generates r_{α_i} and R_{α_i} values for each attribute, such that the attribute α_i and the corresponding value v_i can be linked together. Therefore, σ_{α_i} and ω_{α_i} are also randomized with r_{α_i}, and can both be verified with R_{α_i}.

7.2 AGGREGATED SIGNATURES IN A CREDENTIAL

As proposed in our design chapter (Chapter 5), we will use aggregated signatures for signing the attributes in order to reduce the total size of the credential. To realize this, we use the *RSPS-M* scheme introduced and discussed in Section 6.2. The delegator signs each attribute he wishes to delegate and aggregates the signature with the aggregated signature from its predecessor. This allows the total number of attribute signatures in the credential to reduce from $n * (L + 1)$ signatures to n signatures.

Table 7.1 gives an overview of how the (aggregated) signatures in a credential are interlinked together by the R-values as before.

	α_1	α_2	α_3	R	ϕ
root				R_1	$\phi_{r \to D_1}$
D_1	σ_{α_1}	σ_{α_2}	σ_{α_3}	R_2	$\phi_{D_1 \to D_2}$
D_2				R_3	$\phi_{D_2 \to D_3}$
D_3	ω_{α_1}	ω_{α_2}	ω_{α_3}	R_P	$\phi_{D_3 \to P}$
	R_{α_1}	R_{α_2}	R_{α_3}		

Table 7.1: Aggregated signatures in a credential: Relation between signatures and R-values.

Like before, the delegation signature $\phi_{D_x \to D_y}$ is linked to the R_x-value of delegator D_x. The aggregated signature σ_{α_i} will depend on the R_i-values of all delegators, the R_p value of the credential issuer and R_{α_i}. The commitment signature ω_{α_i} is linked to the R_p value of the credential issuer and the attribute-value binding element R_{α_i}.

7.3 RANDOMIZABILITY

One of the properties of the credential is randomizability. Provers randomize their credential before generating presentation token in order to make multiple presentations of the same credential

unlinkable. To randomize the credential, the prover picks $r \leftarrow \mathbb{Z}_q$ for each R-value in the credential. The prover will then use r to randomize the R-value and all related signatures in the credential which are linked to that R-value. For instance, when the prover randomizes R_2 in Table 5.2, the signatures σ_{α_1}, σ_{α_2}, σ_{α_3} and $\phi_{D_1 \to D_2}$ need to be randomized as well with the same r. During verification, the verifier will observe different R-values every presentation of the same credential and can therefore not link multiple presentations together. Nevertheless, the credential remains valid, since the credential still proofs the correctness of the signatures and the relations between the signatures by verifying signatures with the corresponding R-values.

7.4 CONSTRUCTION

We incorporate the three preliminary signature schemes introduced in Chapter 6 in order to create a concrete instantiation of our protocol.

7.4.1 *Delegation*

The *Randomizable BLS Multi-Signatures with Public-Key Aggregation* scheme proposed in Section 6.2 is used for the signing of attributes during delegation. When a delegator wishes to delegate a set of acquired attributes, the delegator signs all attributes, and the aggregates the signatures with the aggregate signature of its predecessors. Therefore the delegation bundle will always contain at most n attribute signatures. The public key of the root entity rpk will be used as the seed value for the hash function in the RBLS scheme, and the counter value L will be the delegation level of the current delegator. The first delegator from the root entity to the first delegator uses counter value $L = 0$.

The delegation chain will be signed using the *Randomizable Structure-Preserving Signatures by Multiplication* scheme proposed in Section 6.3. If a level-L delegator wishes to assign signing rights to a level-$(L+1)$ delegatee, the secret key will be $sk := dsk_L$ and the message $m := dpk_{L+1}$ will be the public key of its delegatee. The delegation chain will be constructed using the *Double Key Pair* method as discussed in Section 5.3.

On each level L, the attribute signatures and the delegation signature will be randomized with the same $r \xleftarrow{R} \mathbb{Z}_q$ and can all be verified with the same randomization element R_{L+1}. After the aggregation of the attribute signature, the aggregate signature can be verified with $R \leftarrow \prod_{j=1}^{L} R_j$.

7.4.2 *Credential Issuance*

When the delegator wishes to issue credentials to provers, delegators create the same signatures as if they would delegate signing rights. However, when the delegator signs the public key of the prover, the delegator will use a different group element y_3 from the public parameters instead of y_2. This prevents the prover from gaining signing rights. Besides these steps, attribute values need to be assigned with the to-be-issued attributes. The *Short Randomizable Signatures by Multiplication* scheme proposed in Section 6.4 will be used to sign commitments over attribute values. The prover commits the attribute values for the credential and sends them to the issuing delegator, along with a zero-knowledge proof proving that the commitment, in fact, open to the attribute values. The verifier signs the commitments and links the commitment signatures to the corresponding attribute signatures. Because both the RSPS-M and the SRS-M scheme support randomization by multiplication, the attribute-value pairs can be interlinked to the same randomization element R_{α_i}.

7.4.3 *Presentation Token Generation and Verification*

To generate a presentation token, the prover will first randomize all R-values in the credential and all signatures related to the corresponding R-values accordingly. To reduce the size of the presentation token, the aggregated signatures of the to-be-presented attributes will be aggregated to a single batch signature.

After these steps, the prover will incorporate Schnorr's zero-knowledge proof of knowledge scheme to prove the knowledge of elements in verification pairings. First, the prover generates random γ-values for each element/value which need to be zero-knowledge proved and commits these values according to the pairing in which they need to be verified. Then, a Fiat-Shamir hash will be calculated over the commitment values and other elements which will be sent to the verifier, such as the R-values and rpk. The hash output will be the challenge value for the non-interactive proof. The response values can be generated with this challenge value c.

During verification, the verifier will use the response values to compute the verification pairings. The pairings should result in the same commitment values as the computed commitment values by the prover. The verifier can compute the Fiat-Shamir hash in the same fashion, and check if the hash is the same as the challenge value of the prover.

Figure 7.1 depicts all verification equations, where the Schnorr-based proofs will prove the knowledge over the signatures, public keys in the chain, hidden values of attributes and the prover's

private key. The to-be-proved elements and values are underlined in Figure 7.1 and will be replaced with witnesses (the response value). The figure distinguishes between attribute values which are disclosed or hidden. When attribute values are hidden, the prover only proves that it has knowledge over an attribute value, but does not disclose it. D_v represents the set of attribute values which the prover wishes to disclose.

$$
\begin{aligned}
SPK\Bigg\{ & (\sigma, apk \leftarrow \prod_{j=1}^{L} dpk_j^{H(seed||j)}, \langle \tilde{dpk}_j, \phi_j \rangle_{j=1}^{L}, \phi_P, X, \langle v_i, t_i \rangle_{i \notin D_v}, psk) : \\
& e(\underline{\sigma}, g_2) = e(M, \underline{apk}) \cdot e(M, rpk^{H(seed||0)} \cdot \prod_{j=1}^{L} R_j) \cdot \prod_{i=1}^{n} e(H_1(\alpha_i), R_{\alpha_i}) \wedge \\
& e(\underline{\phi_P}, g_2) = e(g_1^{\frac{psk}{}}, R_j) \cdot e(y_3, R_P) \cdot e(y_1, \underline{dpk_L}) \wedge \\
& \bigwedge_{j=1}^{L} \Bigg(e(\underline{\tilde{dpk}_j}, g_2) = e(g_1, \underline{dpk_j}) \wedge \\
& \qquad e(\underline{\phi_j}, g_2) = e(\underline{\tilde{dpk}_j}, R_j) \cdot e(y_2, R_j) \Big[\cdot e(y_1, \underline{dpk_{j-1}}) \Big]_{j \neq 1} \Big[\cdot e(y_1, rpk) \Big]_{j=1} \Bigg) \wedge \\
& \bigwedge_{i=1}^{n} e(\pi_{v_i}, R_p \cdot R_{\alpha_i}) = e(\varphi_{v_i,1}, g_2) \wedge \\
& \quad \bigwedge_{i \in D_v} e(\varphi_{v_i,2}, h) = e(\varphi_{v_i,1}, \underline{X}) \cdot e(\varphi_{v_i,1}, Y^{v_i}) \cdot e(\varphi_{v_i,1}, h)^{\underline{t_i}} \wedge \\
& \quad \bigwedge_{i \notin D_v} e(\varphi_{v_i,2}, h) = e(\varphi_{v_i,1}, \underline{X}) \cdot e(\varphi_{v_i,1}, Y)^{\underline{v_i}} \cdot e(\varphi_{v_i,1}, h)^{\underline{t_i}} \Bigg\}
\end{aligned}
$$

Figure 7.1: Signature Proof of Knowledge (witnesses are underlined for clarity).

7.5 SIGNATURE SCHEMES

To construct the efficient consecutive delegation of signing rights scheme, we will use the *Randomizable BLS Multi-Signatures with Public-Key Aggregation* scheme proposed in Section 6.2, the *Randomizable Structure-Preserving Signatures by Multiplication* scheme proposed in Section 6.3 and the *Short Randomizable Signatures by Multiplication* scheme proposed in Section 6.4.

The three signature schemes are slightly modified for the protocol, such that multiple signatures can be randomized using the same r. Therefore the signing and randomization methods will not handle the generation of new r's, but instead, take them as input.

7.5.1 *General*

$\mathsf{Setup}(\kappa)$: Run $\Lambda \leftarrow \mathsf{BGGen}(\kappa)$. Choose random $y_1, y_2, y_3 \xleftarrow{R} \mathbb{G}_1$, $\tilde{h} \xleftarrow{R} \mathbb{G}_1$, $h \xleftarrow{R} \mathbb{G}_2$ and $y \xleftarrow{R} \mathbb{Z}_q$. Compute $\tilde{Y} \leftarrow \tilde{h}^y$ and $Y \leftarrow h^y$. Output: $\Lambda = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, y_1, y_2, y_3, \tilde{h}, h, \tilde{Y}, Y)$.

$\mathsf{KeyGen}()$: Choose random $sk \xleftarrow{R} \mathbb{Z}_q$ and set $\tilde{pk} \leftarrow g_1^{sk} \in \mathbb{G}_2$ and $pk \leftarrow g_2^{sk} \in \mathbb{G}_2$. Compute $\tilde{X} \leftarrow \tilde{h}^{sk}$, $X \leftarrow h^{sk}$, and output secret $\leftarrow (sk, \tilde{X})$ and public $\leftarrow (\tilde{pk}, pk, X)$.

$\mathsf{VerifyKeypair}(\tilde{pk}, pk)$: If $e(\tilde{pk}, g_2) = e(g_1, pk)$ output 1 otherwise 0.

$\mathsf{GenR}()$: Choose random $r \xleftarrow{R} \mathbb{Z}_q$ and set $R \leftarrow g_2^r \in \mathbb{G}_2$. Output: r and R.

$\mathsf{RandomizeR}(R)$: Choose random $r \xleftarrow{R} \mathbb{Z}_q$ and set $R' \leftarrow R \cdot g_2^r \in \mathbb{G}_2$. Output: r and R'.

7.5.2 *Randomizable BLS Signature - Single Signature*

$\mathsf{RBLS.Sign}(sk, \alpha, L, r)$: Output $\sigma = H_1(\alpha)^{sk \cdot H(seed||L)+r} \in \mathbb{G}_1$.

$\mathsf{RBLS.Verify}(pk, \alpha, L, \sigma, R)$: If $e(\sigma, g_2) = e(H_1(\alpha), pk^{H(seed||L)} \cdot R)$ output 1 otherwise 0.

$\mathsf{RBLS.Randomize}(\alpha, \sigma, r)$: Output $\sigma' = \sigma \cdot H_1(\alpha)^r \in \mathbb{G}_1$.

7.5.3 *Randomizable BLS Signature - Multi Signature*

$\mathsf{RBLS.Aggregate}(\langle \sigma_i \rangle_{i=1}^n)$: Output: $\sigma = \prod_{i=1}^n \sigma_i$.

$\mathsf{RBLS.AggVerify}(\langle pk_i \rangle_{i=1}^n, \alpha, \sigma, \langle R_i \rangle_{i=1}^n)$:
Compute $apk \leftarrow \prod_{i=1}^n pk_i^{H(seed||i)}$ and $R \leftarrow \prod_{i=1}^n R_i$. If $e(\sigma, g_2) = e(H_1(\alpha), apk \cdot R)$ output 1 otherwise 0.

$\mathsf{RBLS.AggRandomize}(\alpha, \sigma, r)$: Output $\sigma' = \sigma \cdot H_1(\alpha)^r \in \mathbb{G}_1$.

7.5.4 *Randomizable BLS Signature - Batch Signature*

$\mathsf{RBLS.BatchAggregate}(\langle \sigma_i \rangle_{i=1}^n)$: Output: $\sigma = \prod_{i=1}^n \sigma_i$.

$\mathsf{RBLS.BatchVerify}(\langle pk_i, R_i, \alpha_i \rangle_{i=1}^n, \sigma)$: Compute $apk \leftarrow \prod_{i=1}^n pk_i^{H(seed||i)}$ and $R \leftarrow \prod_{i=1}^n R_i$. If $e(\sigma, g_2) = e(H_1(\alpha_1) \cdots H_1(\alpha_n), apk \cdot R)$ output 1 otherwise 0.

$\mathsf{RBLS.BatchRandomize}(\langle \alpha_i \rangle_{i=1}^n, \sigma, r)$: Output $\sigma' = \sigma \cdot (\prod_{i=1}^n H_1(\alpha_i))^r \in \mathbb{G}_1$.

7.5.5 *Randomizable Structure-Preserving Signatures by Multiplication*

$\mathsf{RSPS.Sign}(sk, m, r)$: Output $\phi = y_1^{sk} \cdot (y_2 \cdot m)^r \in \mathbb{G}_1$.

$\mathsf{RSPS.Sign2}(sk, m, r)$: Output $\phi = y_1^{sk} \cdot (y_3 \cdot m)^r \in \mathbb{G}_1$.

RSPS.Verify(pk, m, ϕ, R): If $e(\phi, g_2) = e(y_1, pk) \cdot e(y_2 \cdot m, R)$ output 1 otherwise 0.

RSPS.Randomize(m, ϕ, r): Output $\phi' = \phi \cdot (y_2 \cdot m)^r \in \mathbb{G}_1$.

7.5.6 *Short Randomizable Signatures by Multiplication*

SRS.Commit(m): Choose random $b \xleftarrow{R} \mathbb{Z}_q$, and compute $C \leftarrow \tilde{h}^b \tilde{Y}^m \in \mathbb{G}_1$. Output: (C, b).

SRS.Sign$(\tilde{X}, C, r)$: Choose random $s \xleftarrow{R} \mathbb{Z}_q$. Compute $\varphi' \leftarrow (\tilde{h}^{sr}, (\tilde{X}C)^{sr})$, $\pi' \leftarrow (\tilde{h}^s, (\tilde{X}C)^s)$. Output: $\omega' \leftarrow (\varphi', \pi')$.

SRS.Unblind(ω', b): parse $\omega' = ((\varphi'_1, \varphi'_2), (\pi'_1, \pi'_2))$. Output $\omega \leftarrow ((\varphi'_1, \varphi'_2 / {\varphi'_1}^b), ((\pi'_1, \pi'_2 / {\pi'_1}^b))$.

SRS.Verify(X, m, φ): Parse $\varphi = (\varphi_1, \varphi_2)$. If $\varphi_1 \neq 1_{G_2}$ and $e(\varphi_1, X \cdot Y^m) = e(\varphi_2, h)$, output 1 otherwise 0.

SRS.Randomize(ω, r): Parse $\omega = ((\varphi_1, \varphi_2), (\pi_1, \pi_2))$. Output $\omega' \leftarrow ((\varphi_1 \cdot \pi_1^r, \varphi_2 \cdot \pi_2^r), (\pi_1, \pi_2))$.

SRS.GenPoK(ω): Parse $\omega = ((\varphi_1, \varphi_2), \pi)$. Choose random $r, t \xleftarrow{R} \mathbb{Z}_q$. Compute $\varphi' \leftarrow (\varphi_1^r, (\varphi_2 \cdot \varphi_1^t)^r)$ and output (t, φ').

7.6 CONCRETE INSTANTIATION

By incorporating the slightly modified signature schemes from the previous section, we give a concrete instantiation of ENTRUST.

7.6.1 *Setup*

A root entity or another trusted entity not related to the protocol runs $\Lambda \leftarrow \mathsf{Setup}(\kappa)$, and outputs $\Lambda = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, y_1, y_2, y_3, \tilde{h}, h, \tilde{Y}, Y)$. Everyone in the protocol has access to these public parameters at all times. The public key rpk of the root entity will be used as seed for the RBLS signature scheme.

7.6.2 *Initial Delegation*

The root entity wishes to delegate signing rights for particular attributes for anonymous credentials to a delegatee. It will sign the to-be-delegated attributes $\alpha_i \in attr_1$ and the public key dpk_1 of the delegatee. The delegatee will become a level-1 delegator. For the RBLS signature, the root entity starts signing attributes with the counter value $L = 0$.

Algorithm 3 The Initial Delegation Algorithm

input: $rsk, d\tilde{p}k_1, \alpha_i \in attr_1$
$r, R_1 \leftarrow \mathsf{GenR}()$
for $\alpha_i \in attr_1$ **do**
    $\sigma_{\alpha_i} \leftarrow \mathsf{RBLS.Sign}(rsk, \alpha_i, 0, r)$
end for
$\phi_1 \leftarrow \mathsf{RSPS.Sign}(rsk, d\tilde{p}k_1, r)$
output: $\langle \sigma_{\alpha_i} \rangle_{i=1}^{n}, \phi_1, R_1$

7.6.3 *Consecutive Delegation*

A delegator on level-L wishes to assign signing rights for a subset of its acquired attributes to another delegatee. The delegator will chooses the attributes $\alpha_k \in attr_{L+1}$, such that $attr_{L+1} \subseteq attr_L$. The delegator will sign the attributes in $attr_{L+1}$ and aggregate the signatures with the attribute signatures of the previous delegators. Also, the delegator signs the public key dpk_{L+1} of the delegatee. The counter value for the RBLS signature will be the current level of the delegator.

Algorithm 4 The Consecutive Delegation Algorithm

delegator input: $dsk_L, d\tilde{p}k_{L+1}, \alpha_k \in attr_{L+1}$
input previous delegator: $\langle \sigma_{\alpha_i} \rangle_{i=1}^{n}, \langle \phi_j, R_j, d\tilde{p}k_j, dpk_j \rangle_{j=1}^{L}, rpk, attr_L$
$r, R_{L+1} \leftarrow \mathsf{GenR}()$
for $\alpha_k \in attr_{L+1}$ **do**
    $\sigma'_{\alpha_k} \leftarrow \mathsf{RBLS.Sign}(dsk_L, \alpha_k, L, r)$
    $\sigma_{\alpha_k} \leftarrow \mathsf{RBLS.Aggregate}(\{\sigma_{\alpha_k}, \sigma'_{\alpha_k}\})$
end for
$\phi_{L+1} \leftarrow \mathsf{RSPS.Sign}(dsk_L, d\tilde{p}k_{L+1}, r)$
output: $\langle \sigma_{\alpha_k} \rangle_{k=1}^{|attr_{L+1}|}, \phi_{L+1}, R_{L+1}$

7.6.4 *Credential Issuance*

A prover with public key ppk wants to obtain a credential from a delegator on level-L. The prover chooses $\alpha_k \in attr_C$ and defines the attribute values $v_k \in val$ for the corresponding attributes. The prover to generate commitments over the attribute values (Algorithm 5) and send the commitments $com_{v_k} \in com$ to the verifier, along with a non-interactive zero-knowledge proof of knowledge over the opening of the commitment to the hidden attribute values. For simplicity, we use the Schnorr's Proof of Knowledge protocol [67] in combination with the Fiat-Shamir heuristic [40] to make the proof non-interactive. The

proof can also be carried out interactively or be replaced with another zero-knowledge proof of knowledge scheme.

Algorithm 5 The Commitment Algorithm (Prover)

input from the issuing delegator: $\langle v_i \rangle_{i=1}^{n}$
for $i = 1, \ldots, n$ **do**
    $(b_i, com_{v_i}) \leftarrow \mathsf{SRS.Commit}(v_i)$
    $p \xleftarrow{R} \mathbb{Z}_q$
    $P_i \leftarrow \tilde{h}^p$
    $c \leftarrow H(com_{v_i} || P_i)$
    $s_i \leftarrow p + c \cdot b_i$
end for
output: $\langle b_i, (com_{v_i}, P_i, s_i) \rangle_{i=1}^{n}$

The delegator receives $attr_C, val, \langle (com_{v_i}, P_i, s_i) \rangle_{i=1}^{n}$ from the prover, and checks whether $attr_C \subseteq attr_L$. When the delegator decides to endorse the provided claims, he checks if the non-interactive zero-knowledge proof of knowledge is correct. If the delegator is convinced that the prover knows the opening values of the provided commitments, the delegator will sign the commitments and link them to the corresponding attributes. The delegator will sign the public key of the prover with RSPS.Sign2 instead of RSPS.Sign, in order to prevent the prover to obtain signing rights and to issue credentials.

Algorithm 6 The Credential Issuance Algorithm

delegator input: $dsk_L, \tilde{X}_L$
input from prover: $\tilde{ppk}, \alpha_k \in attr_C, v_k \in val, \langle com_{v_k}, P_k, s_k \rangle_{k=1}^{|val|}$
input previous delegator: $\langle \sigma_{\alpha_i} \rangle_{i=1}^{n}, \langle \phi_j, R_j, \tilde{dpk}_j, dpk_j \rangle_{j=1}^{L}, rpk, attr_L$
check if: $attr_C \subseteq attr_L$ **else return** $\perp$
for $v_k \in val$ **do**
    $c \leftarrow H(com_{v_i} || P_i)$
    check if: $\tilde{h}^{s_k} \cdot \tilde{Y}^{v_k \cdot c} \stackrel{?}{=} (com_{v_k})^c \cdot P_k$ **else return** $\perp$
end for
$r_P, R_P \leftarrow \mathsf{GenR}()$
for $\alpha_k \in attr_C$ **do**
    $r_{\alpha_k}, R_{\alpha_k} \leftarrow \mathsf{GenR}()$
    $\sigma'_{\alpha_k} \leftarrow \mathsf{RBLS.Sign}(dsk_L, \alpha_k, L, r_P + r_{\alpha_k})$
    $\sigma_{\alpha_k} \leftarrow \mathsf{RBLS.Aggregate}(\{\sigma_{\alpha_k}, \sigma'_{\alpha_k}\})$
    $\omega_{\alpha_k} \leftarrow \mathsf{SRS.Sign}(\tilde{X}_L, com_{v_k}, r_P + r_{\alpha_k})$
end for
$\phi_P \leftarrow \mathsf{RSPS.Sign2}(dsk_L, \tilde{ppk}, r)$
output: $\langle \sigma_{\alpha_k}, \omega_{\alpha_k}, R_{\alpha_k} \rangle_{k=1}^{|attr_C|}, \phi_P, R_P$

7.6.5 *Presentation Token Generation*

When the prover wants to generate a new presentation token, the prover will randomize all R-values, and all signatures accordingly, depending on the relations to the R-values. The commitment signatures ω_i' will be unblinded by the prover with the b_i values used during commitment. Instead of including the signatures and the public keys directly in the presentation token, the prover generates zero-knowledge proofs of knowledge about the group elements instead. Figure 7.1 depicts all the verification equations which the verifier needs to run in order to be convinced. Based on the verification equations, the commitment values are generated by the prover.

As example, we demonstrate this by creating a Zero-Knowledge Proof of Knowledge over a simple pairing equation $e(A, \underline{B}) = e(\underline{C}, D)$, where $A, C \in \mathbb{G}_1$, $B, D \in \mathbb{G}_2$ and group elements B and C need to be zero-knowledge proved.

The verification equation can be rewritten as $0 = e(A, B) \cdot e(C, D)^{-1}$. For each element that need to be hidden, a random value γ will be generated. Therefore $\gamma_B, \gamma_C \xleftarrow{R} \mathbb{Z}_q$. The response values of the to-be-hidden elements will be $res_B \leftarrow g_2^{\gamma_B} B^c$ and $res_C \leftarrow g_2^{\gamma_C} C^c$. When we use these response values in the verification, it will result in:

$$
\begin{aligned}
& e(A, res_B) \cdot e(res_C, D)^{-1} \\
=& e(A, g_2^{\gamma_B} B^c) \cdot e(g_2^{\gamma_C} C^c, D)^{-1} \\
=& e(A, g_2^{\gamma_B}) \cdot e(A, B^c) \cdot e(g_2^{\gamma_C}, D)^{-1} \cdot e(C^c, D)^{-1} \\
=& e(A, g_2)^{\gamma_B} \cdot e(g_2, D)^{-\gamma_C} \cdot e(A, B)^c \cdot e(C, D)^{-c} \\
=& e(A, g_2)^{\gamma_B} \cdot e(g_2, D)^{-\gamma_C} .
\end{aligned}
$$

The hidden elements in the response value will be used to verify the verification pairings. If the prover does have the knowledge over B and C, the verification parings will cancel each other out, since they should be equal. This will result in a remainder value, which will become the commitment value. The prover will precalculate these commitments values using the generated random values γ and should be the same as the remainder value of the verification pairings of the verifier.

Our protocol supports predicates, which allows prover to minimizes information disclosure depending on the use case. The predicates in our protocol can be easily extended to support e.g., equality proofs, range proofs [17, 19, 62], set membership proofs [19], AND, OR, and NOT proofs [21], or any other predicate proofs based on Pedersen commitments. Nevertheless, only the predicates ProveValue and ProveCommitment are shown in the algorithm for demonstration purposes, where ProveValue is used to disclose the value of attributes, and ProveCommitment is used to solely proof the

Algorithm 7 The Presentation Token Generation Algorithm

prover input: $psk, \alpha_k \in attr_D, \rho_k \in predicate, \langle b_i \rangle_{i=1}^{n}$

input from signer: $\langle \sigma_{\alpha_i}, \omega'_{\alpha_i}, R_{\alpha_i} \rangle_{i=1}^{n}, \left\langle \phi_j, R_j, \tilde{dpk}_j, dpk_j \right\rangle_{j=1}^{L}, rpk, \tilde{ppk}, ppk, \phi_P, R_P, X_L, attr, val$

$r_P, R'_P \leftarrow \mathsf{RandomizeR}(R_P)$

$\phi'_P \leftarrow \mathsf{RSPS.Randomize}(\tilde{ppk}, \phi_P, r_P)$

for $R_j \in R_1, \ldots, R_L$ **do**

  $r_j, R'_j \leftarrow \mathsf{RandomizeR}(R_j)$

  $\phi'_j \leftarrow \mathsf{RSPS.Randomize}(\tilde{dpk}_j, \phi_j, r_j)$

end for

for $\alpha_k \in \alpha_1, \ldots, \alpha_{|attr_D|}$ **do**

  $r_{\alpha_k}, R'_{\alpha_k} \leftarrow \mathsf{RandomizeR}(R_{\alpha_k})$

  $\sigma'_{\alpha_k} \leftarrow \mathsf{RBLS.AggRandomize}(\alpha_k, \sigma_{\alpha_k}, \left(\sum_{j=1}^{L} r_j\right) + r_P + r_{\alpha_k})$

  $\omega_{\alpha_k} \leftarrow \mathsf{SRS.Unblind}(\omega'_{\alpha_k}, b_k)$

  $\omega^*_{\alpha_k} \leftarrow \mathsf{SRS.Randomize}(\omega_{\alpha_k}, r_P + r_{\alpha_k})$

  $(t_k, \varphi_{\alpha_k}) \leftarrow \mathsf{SRS.GenPoK}(\omega^*_{\alpha_k})$

end for

$\sigma \leftarrow \mathsf{RBLS.BatchAggregate}(\{\sigma'_{\alpha_1}, \ldots, \sigma'_{\alpha_{|attr_D|}}\})$

$\gamma_\sigma, \left\langle \gamma_{\phi_j}, \gamma_{dpk_j}, \gamma_{\tilde{dpk}_j} \right\rangle_{j=1}^{L}, \langle \gamma_{v_i}, \gamma_{t_i} \rangle_{k=1}^{|attr_D|}, \gamma_{\phi_P}, \gamma_{psk}, \gamma_X \xleftarrow{R} \mathbb{Z}_q$

$M \leftarrow \prod_{i=1}^{n} H_1(\alpha_i) \in \mathbb{G}_1$

$com_\sigma \leftarrow e(g_1^{-1}, g_2)^{\gamma_\sigma} \cdot e(M, g_2)^{\sum_{j=1}^{L} \gamma_{dpk_j} \cdot H(seed||j)}$

$com_{\phi_P} \leftarrow e(g_1^{-1}, g_2)^{\gamma_{\phi_P}} \cdot e(g_1, R_P)^{\gamma_{psk}} \cdot e(y_1, g_2)^{\gamma_{dpk_L}}$

for $j = 1, \ldots, L$ **do**

  $com_{\phi_j} \leftarrow e(g_1^{-1}, g_2)^{\gamma_{\phi_j}} \cdot e(g_1, R_j)^{\gamma_{\tilde{dpk}_j}} \left[\cdot e(y_1, g_2)^{\gamma_{dpk_{j-1}}} \right]_{j \neq 1}$

  $com_{dpk_j} \leftarrow e(g_1, g_2)^{\gamma_{\tilde{dpk}_j}} \cdot e(g_1, g_2^{-1})^{\gamma_{dpk_j}}$

end for

for $k = 1, \ldots, |attr_D|$ **do**

  $com_{v_k} \leftarrow \mathsf{Entrust.ValComm}(\rho_k)$ ▷ Algorithm 8

end for

$c \leftarrow H\left(rpk, com_\sigma, \left\langle R_j, com_{\phi_j}, com_{\tilde{dpk}_j}, com_{dpk_j} \right\rangle_{j=1}^{L}, com_{\phi_P}, \langle R_{\alpha_k}, \varphi_{\alpha_k} \rangle_{k=1}^{|attr|}\right)$

$res_\sigma \leftarrow g_1^{\gamma_\sigma} \sigma^c$

$res_{\phi_P} \leftarrow g_1^{\gamma_{\phi_P}} \phi_P^c$

$res_{psk} \leftarrow \gamma_{psk} + c \cdot psk$

$res_X \leftarrow g_2^{\gamma_X} X_L^c$

for j = 1, ..., L **do**

  $res_{\phi_j} \leftarrow g_1^{\gamma_{\phi_j}} \phi_j^c$

  $res_{dpk_j} \leftarrow g_2^{\gamma_{dpk_j}} dpk_j^c$

  $res_{\tilde{dpk}_j} \leftarrow g_1^{\gamma_{\tilde{dpk}_j}} \tilde{dpk}_j^c$

end for

for $k = 1, \ldots, |attr_D|$ **do**

  $(res_{v_k}, res_{t_k}) \leftarrow \mathsf{Entrust.ValRes}(\rho_k)$ ▷ Algorithm 8

end for

output: $c, res_\sigma, \left\langle res_{\phi_j}, res_{dpk_j}, res_{\tilde{dpk}_j}, R'_j \right\rangle_{j=1}^{L}, \langle res_{v_k}, res_{t_k}, R'_{\alpha_k}, \varphi_{\alpha_k} \rangle_{k=1}^{|attr|}, res_{\phi_P}, res_{psk}, R'_P$

knowledge over the value. For every attribute α_k the prover wishes to disclose, the prover will choose a predicate type ρ_k over the corresponding attribute value.

Algorithm 8 Predicate Commitments

procedure ENTRUST.VALCOMM(ρ_k)
  if $\rho_k ==$ ProveValue **then** ▷ Value is disclosed
    $com_{\nu_k} \leftarrow e(\varphi_{\nu_k,1}, g_2)^{\gamma_X} \cdot e(\varphi_{\nu_k,1}, h)^{\gamma_{t_k}}$
  end if
  if $\rho_k ==$ ProveCommitment **then** ▷ Value is hidden
    $com_{\nu_k} \leftarrow e(\varphi_{\nu_k,1}, g_2)^{\gamma_X} \cdot e(\varphi_{\nu_k,1}, Y)^{\gamma_{\nu_k}} \cdot e(\varphi_{\nu_k,1}, h)^{\gamma_{t_k}}$
  end if
end procedure

procedure ENTRUST.VALRES(ρ_k)
  if $\rho_k ==$ ProveCommitment **then** ▷ Value is hidden
    $res_{\nu_k} \leftarrow \gamma_{\nu_k} + \nu_k \cdot c$
    $res_{t_k} \leftarrow \gamma_{t_k} + t_k \cdot c$
    return (res_{ν_k}, res_{t_k})
  end if
  return $\emptyset$
end procedure

7.6.6 *Zero-Knowledge Verification*

As discussed in the prior subsection, the verifier will use the response values to compute the verification pairings. The remainder value of the pairings should be equal to the commitment value of the prover. When all remainder values are the same as the commitment values of the prover, i.e., the challenge hash from the prover is equal to the hash of the verifier, the verifier is convinced that the prover has the knowledge over the hidden elements depicted in Figure 7.1.

Algorithm 9 The Zero-Knowledge Verification Algorithm

verifier input: Λ, rpk

input from prover: $attr, c, res_{\sigma}, res_{\phi_P}, res_{psk}, res_X, R_P, \left\langle res_{\phi_j}, res_{dpk_j}, res_{\tilde{dpk}_j}, R_j \right\rangle_{j=1}^{L}, \langle res_{v_i}, res_{t_i}, R_{\alpha_i}, \varphi_{v_i}, \pi_{v_i} \rangle_{i=1}^{n}$

$M \leftarrow \prod_{i=1}^{n} H_1(\alpha_i) \in \mathbb{G}_1$

$R \leftarrow R_P \cdot \prod_{j=1}^{L} R_j \cdot \prod_{i=1}^{n} R_{\alpha_i} \in \mathbb{G}_2$

$res_{apk} \leftarrow \prod_{j=1}^{L} res_{dpk_j}^{H(seed||j)}$

$com_{\sigma} \leftarrow e(res_{\sigma}, g_2^{-1}) \cdot e(M, res_{apk}) \cdot e(M, rpk^{H(seed||0)} \cdot R)^c \cdot \prod_{i=1}^{n} e(H_1(\alpha_i), R_{\alpha_i})^c$

$com_{\phi_P} \leftarrow e(res_{\phi_P}, g_2^{-1}) \cdot e(g_1, R_P)^{res_{psk}} \cdot e(y_3, R_P)^c \cdot e(y_1, res_{dpk_L})$

for $j = 1, \ldots, L$ **do**

  $com_{\phi_j} \leftarrow e(res_{\phi_j}, g_2^{-1}) \cdot e(res_{\tilde{dpk}_j}, R_j) \cdot e(y_2, R_j)^c \left[\cdot e(y_1, res_{dpk_{j-1}}) \right]_{j \neq 1} \left[\cdot e(y_1, rpk)^c \right]_{j=1}$

  $com_{dpk_j} \leftarrow e(res_{\tilde{dpk}_j}, g_2) \cdot e(g_1^{-1}, res_{dpk_j})$

end for

for $i = 1, \ldots, n$ **do**

  check if: $e(\pi_{v_i}, R_p \cdot R_{\alpha_i}) \stackrel{?}{=} e(\varphi_{v_i,1}, g_2)$ **else return** $\perp$

  if $\rho_i ==$ ProveValue **then** ▷ Value is disclosed

    $com_{v_k} \leftarrow e(\varphi_{v_i,2}, h)^{-c} \cdot e(\varphi_{v_i,1}, res_X) \cdot e(\varphi_{v_i,1}, Y^{\alpha_i})^c \cdot e(\varphi_{v_i,1}, h)^{res_{t_i}}$

  end if

  if $\rho_i ==$ ProveCommitment **then** ▷ Value is hidden

    $com_{v_k} \leftarrow e(\varphi_{v_i,2}, h)^{-c} \cdot e(\varphi_{v_i,1}, res_X) \cdot e(\varphi_{v_i,1}, Y)^{res_{v_i}} \cdot e(\varphi_{v_i,1}, h)^{res_{t_i}}$

  end if

end for

$c \leftarrow H\left(rpk, com_{\sigma}, \left\langle R_j, com_{\phi_j}, com_{\tilde{dpk}_j}, com_{dpk_j} \right\rangle_{j=1}^{L}, com_{\phi_P}, \langle R_{\alpha_i}, \varphi_{\alpha_i} \rangle_{i=1}^{n}\right)$

output: $c \stackrel{?}{=} c'$

7.7 SECURITY DISCUSSION

Entities in the protocol might have different intentions with the obtained information and might want to gain more knowledge than intended or alter information such it would benefit in their favor. In this section, we will enumerate possible adversarial acts for the different entities in the protocol, and discuss how the protocol mitigates these scenarios.

7.7.1 *Root Authority*

Since the root authority is giving away its signing rights, it is in its best interest to behave honestly in the protocol. Therefore, we consider the root authority as a non-malicious entity in our scheme.

Either the root authority or a trusted entity not participating in the protocol will honestly run Setup, and output $\Lambda = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, y_1, y_2, y_3, \tilde{h}, h, \tilde{Y}, Y)$, where $g_1, y_1, y_2, y_3, \tilde{h}, \tilde{Y} \in \mathbb{G}_1$ and $g_2, h, Y \in \mathbb{G}_2$.

7.7.2 *Delegator*

A delegator receives a delegation bundle from either the root authority or a preceding delegator. A malicious delegator might want to:

- Link valid signatures from the delegation bundle to signatures created by the delegator in order to gain more signing rights;
- Alter signatures, such that the delegator gains more signing rights;
- Mix signatures from different delegation bundles to build a delegation bundle that has never been issued.

Theorem 9. *An adversary is not able to link new signatures to existing randomization elements in the delegation bundle.*

Proof. The protocol verifies the relation between signatures by incorporating the randomization elements in the verification equation. In order to pass verification, the signature needs to randomized with the same secret exponent r as in the randomization element R. Therefore, to link a signature to an existing randomization element, the adversary needs to know the secret random exponent r. However, given R and the signatures linked to R, the adversary does not gain any knowledge about the secret exponent r due to the discrete logarithm problem. This is also proven in Theorem 2. □

Theorem 10. *An adversary is not able to link valid signatures from the delegation bundle to other randomization elements.*

Proof. All signatures σ, ϕ and the randomization elements R are randomized by multiplication. To link a valid signature to a different randomization element, the adversary needs to know the secret random element r which was used to randomize the existing signature. As shown in Theorem 2 and Theorem 9, it is not possible to gain any knowledge about r with the given elements. □

Theorem 11. *A delegator cannot gain more signing rights than delegated.*

Proof. For each delegated attribute $\alpha_i \in attr_L$, the previous delegator puts a attribute signature σ_{α_i} in the delegation bundle. To gain signing rights over the attribute $\alpha_x \notin attr_L$, the malicious delegator need to forge a multi-signature, where all predecessors are involved and randomized with the secret exponent r hidden in the randomization element R.

Assume that the adversary knows the factors of $H_1(\alpha_i) = g_1^s$ and $H_1(\alpha_x) = g_1^t$. The existing valid signature will be $\sigma_{\alpha_i} \leftarrow H_1(\alpha_i)^{\sum_{j=1}^{L}(x_j \cdot H(seed||j)+r_j)} = (g_1^s)^{\sum_{j=1}^{L}(x_j \cdot H(seed||j)+r_j)}$. The adversary is still not able to factorize the exponent s from σ in polynomial time, assuming the discrete logarithm problem. □

Theorem 12. *A delegator cannot mix signatures from different delegation bundles in order to compose a valid delegation bundle which has never been issued.*

Proof. This follows from Theorem 9 and 10. In order to alter relationships between signatures and randomization elements in a delegation bundle, the delegator need to know the secret exponent r. □

7.7.3 *Prover*

Provers create commitments over attribute values they wish to be endorsed and send them to the delegator. In return, the prover receives a credential from the delegator and is able to generate presentation tokens in order to present them to verifiers. A malicious prover might want to:

- Try convincing the delegator that the provided commitments open to specific values, while the commitments, in fact, open to different values.
- Prove invalid predicates over the committed attribute values;
- Gain signing rights and issue credentials for others;
- Prove attributes or attribute values that never have been issued.

Theorem 13. *A prover cannot convince delegators that a given commitment opens to a specific value, while the commitment, in fact, open to a different value.*

Proof. In the protocol, the prover commits to attribute values using Pedersen's commitment scheme. Furthermore, the prover generates a Schnorr's zero-knowledge proof of knowledge which proves that the commitments open to committed values. The prover should not be able to fool the issuing delegator with a high probability since the challenge value depends on both the random commitment value of the proof and the commitment over the attribute value. When an adversary wishes to alter one of the commitment values in order to convince the verifier, the challenge will be different. Therefore it is computationally infeasible to compute

The proof can be made non-interactive by applying the Fiat-Shamir heuristic and is proven secure in the random oracle model. The zero-knowledge proof can also be carried out interactively if the issuing delegator does not want to rely on the random oracle assumption. Nevertheless, □

Assumption 9: A prover cannot prove invalid predicates over the committed attribute values.

The security depends on the underlying predicate scheme. ENTRUST is compatible with all predicate schemes that are compatible with Pedersen commitments.

Theorem 14. *A prover cannot gain signing rights and issue credentials for others.*

Proof. To prevent the prover to gain signing rights, the issuing delegator uses SPS.Sign2 instead of SPS.Sign to sign the public key of the prover. In the algorithm SPS.Sign2, the public element y_2 is replaced with y_3, such that this signature can only be correctly verified with y_3 and not with y_2. □

Theorem 15. *A prover cannot prove attributes or attribute values that never have been issued.*

Proof. To generate a valid attribute token with an attribute that has not been issued by the verifier, the prover need to prove that it knows a valid signature which signs this attribute. Since the prover does not have knowledge of such an attribute signature, he will not be able to generate a valid response value for this signature in order to convince the verifier. This also holds for attribute value signatures. □

7.7.4 *Verifier*

Verifiers receive presentation tokens from provers. A malicious verifier might want to:

- Link multiple presentations of the same credential;
- Gain knowledge about the identity of the prover;
- Gain knowledge about the identities of the intermediate delegators;
- Reuse the proof somewhere else.

The reuse of the presentation token can be prevented by either making the proving process interactive or including an expiration timestamp in the presentation to limit the duration of validity.

By making the proof interactive, verifiers will generate different challenges every time. The adversary cannot correctly respond to the challenges with a high probability, which makes the proofs of the presentation token invalid.

To incorporate non-interactivity, the prover could add a signed message with an expiration timestamp to the presentation token. The verifier would verify this timestamp, and only accept the proof if the signature on the expiration timestamp is valid, and if the timestamp is larger than the current timestamp. This does not prevent a replay of the proof but does limit the timebound in which an adversary can act.

Theorem 16. *A verifier cannot link two presentation tokens to the same credential.*

Proof. For every presentation token, the randomization elements and commitment signatures are randomized with random scalars. This also results in randomized response values. Since every element in the presentation token is completely random, the verifier cannot correlate two presentations to the same credential. This only holds given that the disclosed attribute values are abstract enough and do not contain identifiable information. □

Theorem 17. *A verifier does not learn any information about the identities of the prover and intermediate delegators.*

Proof. The public keys from the prover and intermediate delegators are not disclosed to the verifier, neither are the delegation and attribute signatures. Instead, response values are sent to the verifier in order to prove the knowledge of these public keys and signatures. Therefore, verifiers do not learn anything from the presentation token, while they could be convinced that the proofs are correct. □

7.7.5 *Man in the Middle Adversaries*

When the transmission of messages between the entities is compromised, an adversary could obtain delegation bundles, credentials or presentation tokens. In this section, we show that the compromised information has limited to no value to the adversary.

7.7.5.1 *Root/Delegator → Delegator*

When an adversary would intercept the delegation bundle send from the root authority or delegator to the next delegator, he would try to:

- Gain the signing rights from the delegation bundle.

Theorem 18. *An adversary intercepting delegation bundles cannot obtain signing rights for delegation or the issuance of credentials.*

Proof. The adversary cannot obtain signing rights from the delegation bundle since the previous delegator has dedicated the signing rights specifically to next delegator by signing its public key. Without the knowledge of the secret key of the designated delegator, the adversary cannot create valid delegation chains for credentials and delegation bundles. As seen in Theorem 9, the adversary also cannot generate a new delegation signature ϕ, and link it to an existing R. Therefore, he cannot issue valid credentials nor delegate signing rights to someone else. □

7.7.5.2 *Delegator ↔ Prover*

When an adversary would intercept a credential send from the delegator to a prover, he would try to:

- Use the credential to generate valid zero-knowledge presentation tokens.

Theorem 19. *An adversary intercepting credentials cannot use the credential or generate valid presentation tokens.*

Proof. Credentials are issued in two phases. In the first phase, the prover commits to attributes values and sends the commitments and zero-knowledge proofs to the verifier in order to prove that the commitment opens to the attribute values. In the second phase, the issuing delegator generates a credential and sends it to the prover.

Assume that the adversary intercepts both transmissions. The adversary does not gain any advantage knowing the commitments and zero-knowledge proof sent by the prover since all commitment are blinded by the prover. Additionally, the credential is designated to the public key of the prover. To generate a valid presentation token, the adversary needs to prove knowledge over the secret key of the designated prover. Like in Theorem 18, the adversary cannot replace the delegation signature with a newly crafted signature. Furthermore, the commitments in the credential need to be unblinded with the blinding values used during commitment. The adversary does not have knowledge about the blinding values. □

7.7.5.3 *Prover → Verifier*

When an adversary would intercept a presentation token send from the prover to a verifier, he could attempt to:

- Reuse the proof somewhere else.

We have discussed the reuse of presentation tokens in Section 7.7.4. To recap, there are two possibilities to prevent the reuse of presentation tokens. The first possibility is to let the prover add an expiration timestamp to the presentation token, limiting the duration of validity. This does not prevent the replay of proofs but does limit the timeframe in which an adversary can act. The other possibility is to carry out the zero-knowledge proof interactively. Since the verifiers will send random challenges to the prover, the adversary cannot reuse the intercepted proof, which was created for a specific challenge.

8 EVALUATION

To determine the relevance of our work, we evaluate the theoretical complexity of our protocol in terms of computational speed and storage size. Besides that, we implemented ENTRUST in a prototype to establish the practical performance of our concrete instantiation.

First, we will introduce the setting wherein the prototype will be evaluated, after which we rearrange the verification pairings of ENTRUST to ensure optimal performance. Then, we evaluate the theoretical complexity of the main algorithms in our protocol and show experimental results from our prototype. Finally, we assess our scheme in comparison to a Delegatable Anonymous Credential scheme.

8.1 THE EXPERIMENT SETTING

We realized our concrete instantiation of ENTRUST in a prototype written in the C++ programming language, which provides object-oriented convenience without compromising performance in comparison to the C programming language. We incorporated the RELIC library [34], which is a cryptographic meta-toolkit written in the C programming language. The library currently has state-of-the-art support for bilinear maps and related extension fields [36]. The last official release of RELIC is version 0.4.0 dated to Aug 19, 2014. Since then, multiple curves and hash functions have been added to the library, including the BLS12-381 curve [35]. Therefore, we decided to used the library code from the master branch for our prototype, pulled from the GitHub repository [34] on Sep 3, 2018.

As recommended personally by the library creator Diego F. Aranha, we use the BLS12-381 pairing-friendly elliptic curve for our prototype, which has been introduced by the ZCash team [80]. This curve is favored over the 256-bit Barreto-Naehrig Curve [4] in terms of performance for most protocols and also targets 128-bit security [36]. To improve performance even further, we used RELIC in combination with the GNU Multiple Precision Arithmetic Library (GMP) [41] to enhance performance on precision arithmetics on small and huge operands. As hash function we used SHA-256, which offers 128-bit security against collision attacks [1].

All experiments are run with the Google Benchmark Library [51], which is a microbenchmark library for the C++ programming

1 Keylength - NIST Recommendations 2016 https://www.keylength.com/en/4/

language. By executing a benchmark, the library dynamically determines the number of iterations to run by running the benchmark a few times and measuring the time taken. This ensures that the ultimate result will be statistically stable. Additionally, all benchmarks are run 100 times to ensure the negation of any potential outliers in the results.

We ran the prototype on commodity hardware to prove the practicality of our protocol. All experiments are run on a MacBook Pro with a 2.8 GHz Intel i7-7700HQ CPU.

8.2 RELIC LIBRARY BENCHMARKS

Since the performance of the experiments depends on the used hardware and the curve implementation, this chapter gives an overview of the performance of single operations in RELIC in our experimental setting. Furthermore, we give an overview of the storage size of the elements of the BLS12-381 curve.

8.2.1 *Execution Times of a Single Operation*

In RELIC, we use the "`x64-pbc-128-b12`" preset [2], which incorporates the BLS12-381 pairing friendly eliptic curve, and uses the optimal ate pairing algorithm for pairings.

Table 8.1 gives an overview of the individual execution times of the most expensive operations in our experimental setting on the BLS12-381 curve in RELIC.

	Time (μs)
$\mathbb{G}_1$ Exponentiation	148
$\mathbb{G}_2$ Exponentiation	283
$\mathbb{G}_T$ Exponentiation	452
1 Single Pairing	1456
2-Miller Loop Pairings	1862
n-Miller Loop Pairings	$\sim 1028 + 428n$
Hash string to element in $\mathbb{G}_1$	243

Table 8.1: Execution time of the most expensive operations on the BLS12-381 curve from the RELIC library in our experimental setting.

The execution time of all other operations used in our protocol are considered negligible since they are executed in less than 5 μs, which requires significantly less execution time than the operations listed in Table 8.1.

2 RELIC's "`x64-pbc-128-b12`" Preset https://github.com/relic-toolkit/relic/blob/master/preset/x64-pbc-128-b12.sh

As can be noticed, the pairings of elements are the most expensive operations. However, products of pairings can be computed more efficiently by computing the pairings in the same Miller loop and share a single final exponentiation [10]. This allows three pairings in a product to be computed in less time than it would take to compute two distinct pairings. Figure 8.1 shows the time to compute the product of k pairings, where the k-Miller Loop and final shared exponentiation will be compared to individual pairings.

We also observe that exponentiations in $\mathbb{G}_1$ require less computation time than in $\mathbb{G}_2$ and $\mathbb{G}_T$. Therefore during implementation, it is preferred to exponentiate in $\mathbb{G}_1$ whenever possible.

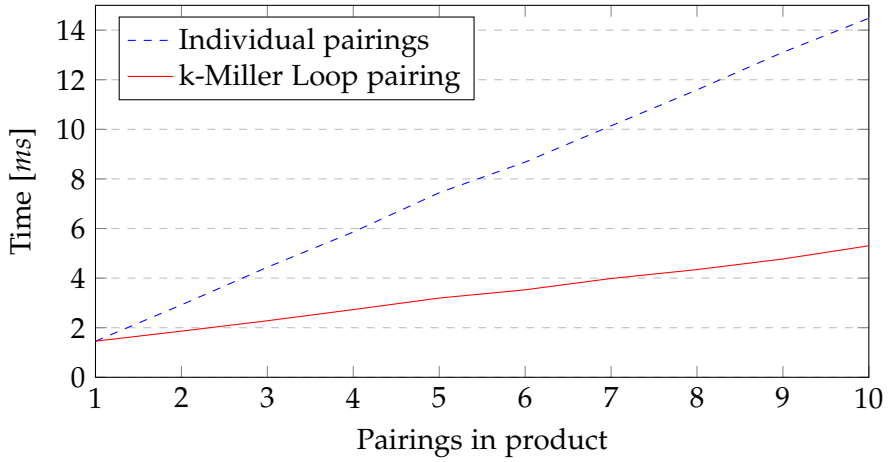

Figure 8.1: Individual Pairings versus k-Miller Loop + Final Exponentiation.

8.2.2 *Storage Size of a Single Group Element*

Table 8.2 shows the storage size of each element of the BLS12-381 curve in RELIC. The RELIC library offers the ability to compress group elements for storage. The size for both uncompressed and compressed elements is shown in the table. We denote BN as Big Number, which are scalars in $\mathbb{Z}_q$. Group elements in $\mathbb{G}_2$ takes roughly twice the storage size in comparison to group elements in $\mathbb{G}_1$.

	Storage Size (bytes)	
	Uncompressed	Compressed
$\mathbb{G}_1$ Element	97	49
$\mathbb{G}_2$ Element	193	97
$\mathbb{G}_T$ Element	576	384
Big Number BN	32	32

Table 8.2: Size of elements of the BLS12-381 curve from the RELIC library.

8.3 OPTIMALIZATION OF VERIFICATION PAIRINGS

We rearrange the verification pairings in Algorithm 9 to get the most optimal computation time during verification in the prototype of ENTRUST. First, we observe from Table 8.1 that exponentiations in $\mathbb{G}_1$ are faster than in $\mathbb{G}_2$ and $\mathbb{G}_T$. Therefore, exponentiations of pairings are moved to the group elements in $\mathbb{G}_1$ whether possible. Second, products of pairings can be computed more efficiently by incorporating the Miller Loop, whereafter the product will be exponentiated with a shared final exponentiation. Therefore, products of pairings are computed together using the same Miller Loop whenever possible. We denote fexp for this operation on a product of pairings.

Algorithm 10 shows a revised version of Algorithm 9, where all pairing equations for the verifier are optimized for efficient computation of the commitment values. Small benchmarks showed an improvement of roughly 23-35% in computation time relative to the non-optimized algorithm for implementation.

Algorithm 10 The Zero-Knowledge Verification Algorithm 9 Revised - Optimized for Implementation

$com_\sigma \leftarrow \mathsf{fexp}\left(e(M, rpk^{H(seed||0)} \cdot R) \cdot \prod_{i=1}^{n} e(H_1(\alpha_i), R_{\alpha_i})\right)^{c} \cdot e(M, res_{apk}) \cdot e(res_\sigma, g_2)^{-1}$

$com_{\phi_P} \leftarrow \mathsf{fexp}\left(e(g_1^{res_{psk}}, R_P) \cdot e(y_3^c, R_P) \cdot e(y_1, res_{dpk_L})\right) \cdot e(res_{\phi_P}, g_2)^{-1}$

for $j = 1, \ldots, L$ **do**

  $com_{\phi_j} \leftarrow \mathsf{fexp}\left(e(res_{\tilde{dpk}_j}, R_j) \cdot e(y_2^c, R_j)\left[\cdot e(y_1, res_{dpk_{j-1}})\right]_{j\neq 1}\left[\cdot e(y_1^c, rpk)\right]_{j=1}\right) \cdot e(res_{\phi_j}, g_2)^{-1}$

  $com_{dpk_j} \leftarrow e(res_{\tilde{dpk}_j}, g_2) \cdot e(g_1, res_{dpk_j})^{-1}$

end for

for $i = 1, \ldots, n$ **do**

  check if: $e(\pi_{v_i}, R_p \cdot R_{\alpha_i}) \stackrel{?}{=} e(\varphi_{v_i,1}, g_2)$ **else return** $\perp$

  if $\rho_i ==$ ProveValue **then** ▷ Value is disclosed

    $com_{v_k} \leftarrow \mathsf{fexp}\left(e(\varphi_{v_i,1}, res_X) \cdot e(\varphi_{v_i,1}^{\alpha_i \cdot c}, Y) \cdot e(\varphi_{v_i,1}^{res_{t_i}}, h)\right) \cdot e(\varphi_{v_i,2}^c, h)^{-1}$

  end if

  if $\rho_i ==$ ProveCommitment **then** ▷ Value is hidden

    $com_{v_k} \leftarrow \mathsf{fexp}\left(e(\varphi_{v_i,1}, res_X) \cdot e(\varphi_{v_i,1}^{res_{v_i}}, Y) \cdot e(\varphi_{v_i,1}^{res_{t_i}}, h)\right) \cdot e(\varphi_{v_i,2}^c, h)^{-1}$

  end if

end for

8.4 COMPLEXITY

This section provides a theoretical performance analysis of ENTRUST. First, we discuss the computational complexity for all entities and main algorithms, where after we evaluate the storage sizes and communication costs of delegation bundles, credentials and presentation tokens.

8.4.1 *Computational Complexity*

We have evaluated all main algorithms of ENTRUST in terms of theoretical complexity and present them for each entity in Tables 8.3 and 8.4. The complexity is determined based on the operations which have a significant impact on the execution time. The expensive operations mentioned Section 8.2.1 are listed in the upper part of the tables. Besides that, we also give the complexity of less significant operations in the lower part of the tables. Note that in Table 8.4, the optimized algorithm for verification is evaluated as introduced in Algorithm 10.

As can be observed, the delegation, credential issuance and commitment algorithms depend on the number of attributes in terms of computational complexity. The complexity of the token generation algorithm is large and depend on both the level of delegation and the number of attributes since the prover needs to randomize all signatures and randomization elements, whereafter it needs to generate non-interactive zero-knowledge proofs for all signatures and public keys. The complexity of the verification algorithm is also $O(n + L)$, as it has to verify the validity of L delegation signatures, n attribute, and n commitment signatures.

	Root Authority	Delegator	
	Delegation	Delegation	Cred. Issuance
Operation type	Complexity		
$\mathbb{G}_1$ Exponentiation	$n+2$	$n+2$	$8n+2$
$\mathbb{G}_2$ Exponentiation	1	1	$n+1$
Hash to $\mathbb{G}_1$	n	n	n
Hash to BN	1	1	$n+1$
Random $BN \in \mathbb{Z}_q$	1	1	$2n+1$
$\mathbb{G}_1$ Multiplication	2	$n+2$	$4n+2$
BN Multiplication	1	1	$n+1$
BN Addition	n	n	$n+1$
Overall Complexity	$O(n)$	$O(n)$	$O(n)$

Table 8.3: Complexity of all operations for the Root Authority and a Delegator.

	Prover		Verifier
	Commitment	Token Generation	Verification
Operation type	Complexity		
$\mathbb{G}_1$ Exponentiation	$3n$	$5L+8n+5$	$L+3n+2$
$\mathbb{G}_2$ Exponentiation	$-$	$3L+n+3$	$L+1$
$\mathbb{G}_T$ Exponentiation	$-$	$5L+2n+4$	1
1 Single Pairing	$-$	$5L+2n+4$	$3L+3n+3$
3-Miller Loop Pairings	$-$	$-$	$L+n+1$
$(n+1)$-Miller Loop Pairings	$-$	$-$	1
Hash to $\mathbb{G}_1$	$-$	n	n
Hash to BN	n	$L+1$	$L+2$
Random $BN \in \mathbb{Z}_q$	$2n$	$4L+3n+5$	$-$
$\mathbb{G}_1$ Multiplication	n	$4L+8n+2$	$2n-1$
$\mathbb{G}_2$ Multiplication	$-$	$2L+n+2$	$2L+2n$
$\mathbb{G}_T$ Multiplication	$-$	$3L+n+2$	$2L+n+3$
$\mathbb{G}_T$ Inversion	$-$	$2L+2$	$2L+n+2$
BN Multiplication	n	$L+2n+1$	n
BN Addition	n	$2L+2n+1$	$-$
Overall Complexity	$O(n)$	$O(n+L)$	$O(n+L)$

Table 8.4: Complexity of all operations for a Prover and a Verifier.

8.4.2 *Complexity of Storage and Communication*

Table 8.5 provides an overview of the storage sizes of delegation bundles, credentials and presentation tokens in our protocol. The storage size complexity of the three data bundles is $O(n + L)$. This is accomplished by aggregating attribute signatures during the delegation phase.

	Complexity	Elements
Delegation bundle size on level L	$O(n + L)$	$(2L + n)\ \mathbb{G}_1 + (2L + 1)\ \mathbb{G}_2$
Credential size at the prover	$O(n + L)$	$(2L + 5n + 2)\ \mathbb{G}_1 +$ $(2L + n + 4)\ \mathbb{G}_2$
Presentation token size send to verifier	$O(n + L)$	$(2L + 3n + 2)\ \mathbb{G}_1 +$ $(2L + n + 2)\ \mathbb{G}_2 +$ $(2n + 2)\ \mathbb{Z}_q$

Table 8.5: Complexity of storage size of the data bundles in ENTRUST.

8.5 RUN-TIME ANALYSIS

We conducted experiments on our prototype in order to determine the practical performance of our protocol. We evaluate the practical performance of generating and verifying presentation tokens, as well as the storage size of presentation tokens in practice.

8.5.1 *Presentation Token Generation Times*

Figure 8.2 shows the computational time for provers to generate presentation tokens, where 1,2,3,4 and 5 intermediate delegators were involved in the delegation process. The shown performance only reflects token generation on a laptop. Although provers in real-life scenarios will likely not use a device as powerful as a laptop to generate presentation tokens, it still gives an impression of the performance could achieve on devices with fewer resources.

The token generation time grows linearly to the number of attributes since the aggregated attribute signatures need be randomized before combining them into a single batch signature.

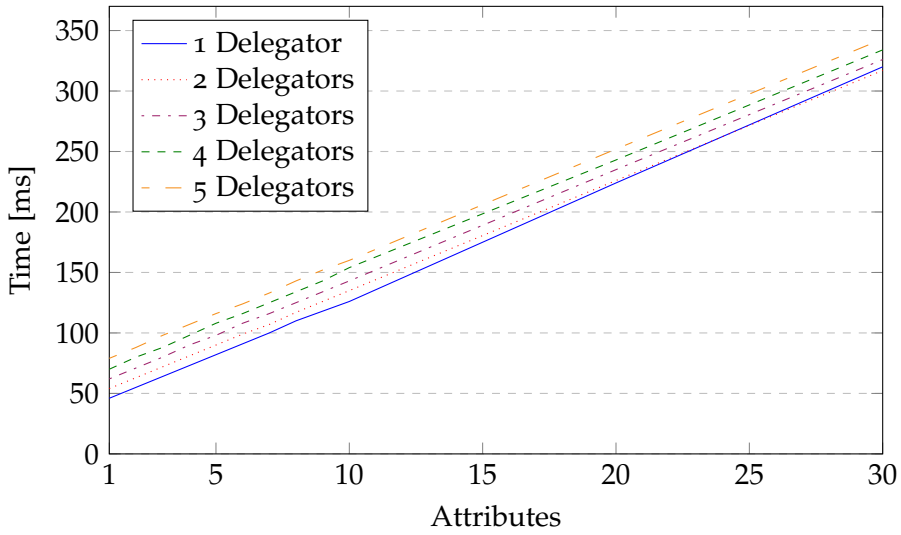

Figure 8.2: Computational time of generating a presentation token.

8.5.2 *Presentation Token Verification Times*

Figure 8.3 depicts the performance of the verification of a presentation token. The verification time grows linearly to the number of attributes. Even though aggregated signatures are combined into a single batch signature, the presentation token is not verified in constant time relative to n, since there are still n commitment signatures in the presentation token which need to be verified.

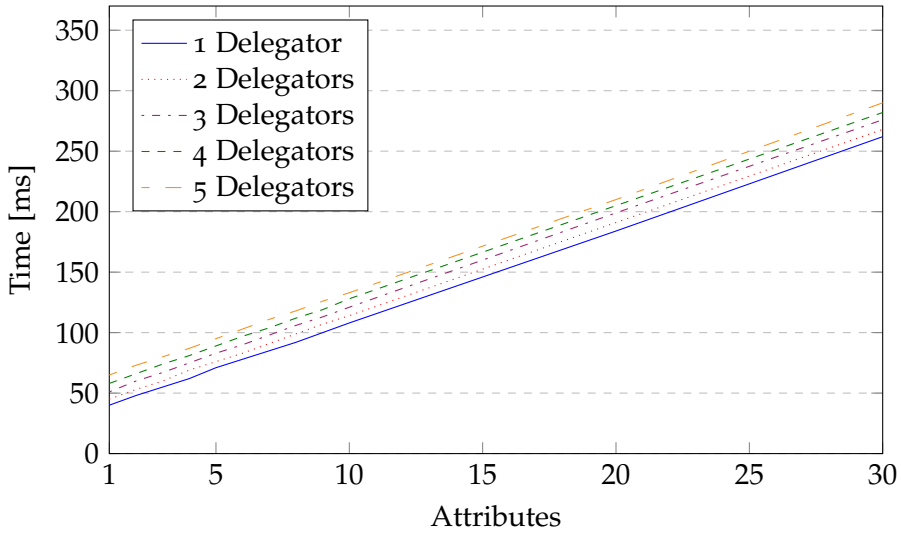

Figure 8.3: Computational time of verifying a presentation token.

8.5.3 *Presentation Token Size*

Figure 8.4 depicts the storage size in practice of a presentation token, where all group elements are stored in the compressed format. As expected, we see that the presentation token size grows linearly to the number of attributes, since each additional attribute requires an extra commitment signature and randomization elements.

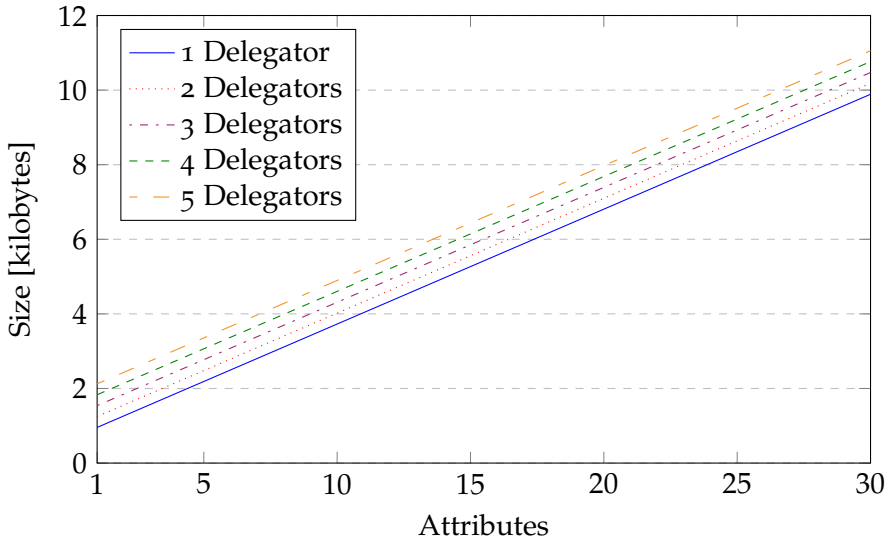

Figure 8.4: The storage size of a presentation token.

8.5.4 *Analysis*

When we assume that an average presentation token contains 10 attributes and is issued by 3 intermediate delegators, the token can be generated in 143 ms, verified in 121 ms, and requires 4312 bytes of storage space. It is noteworthy that the verification speed of the verification token lies within the challenging time bound of 300 ms for access control systems in public transportation.

Presentation tokens containing 30 attributes issued by 3 intermediate delegators are able to be verified in 276 ms. Although the computational speed of verification is acceptable during practical use of presentation tokens, it still might be challenging for devices with restricted resources to generate presentation tokens, since the size of the presentation token grows to 10472 bytes, linearly to the number of attributes. Nevertheless, we expect in practice that credentials could contain many attributes, but that most of the practical verification cases require a few predicates to be proven during the presentation. In use cases where many predicates need to be proven, we expect that strict verification times to ensure optimal human-computer interaction are often not required.

8.6 COMPARISON TO DAC

To the best of our knowledge, we are the first to present a protocol which allows the delegation of signing rights for the issuance of attribute-based credentials. Therefore, there are no concrete schemes which we can directly compare performance against. However, we could divide ENTRUST into two parts: the elements and equations which are responsible for signing right delegation, and elements and equations which are responsible for attribute-based credentials.

For the attribute-based credentials, we use the Short Randomizable Signature scheme of Pointcheval and Sanders [64]. Although we made an adjustment in the signature scheme to include randomization by multiplication, it does not affect the performance of the existing scheme whatsoever.

For the part relevant for the signing right delegation, we compare our scheme to Delegatable Anonymous Credentials, where users can delegate their credentials to other users. Our delegation scheme is actually more similar to Anonymous Proxy Signatures. However, most of the scheme does not give a concrete instantiation or use the Groth-Sahai proof system, which is not as efficient as using Schnorr's zero-knowledge proof of knowledge. DAC is similar to APS, expect that APS delegates subsets of the acquired attributes, while DAC allows different attributes on each delegation level.

Most of the DAC schemes also incorporates the Groth-Sahai proof system. Therefore, we will compare our delegation scheme with the recent work on Delegatable Anonymous Credential from Camenisch et al. [20], as their scheme use the same proof system like ours. Since DAC considers different attributes for each delegation level, it is not an entirely fair comparison. However, it still highlights the effect of signature aggregation on the efficiency in ENTRUST.

8.6.1 *Computational complexity*

Table 8.6 compares the verification complexity for a verifier in the DAC scheme of Camenisch et al. [20] and in ENTRUST. In this comparison, we consider that all delegators delegate the same n attributes and all delegated attributes are disclosed during the presentation. For ENTRUST, solely the correctness of the delegation chain and the attributes will be checked during verification.

	CDD17 [20]	ENTRUST
	Attribute + Delegation All attributes disclosed	Attribute + Delegation
Operation type	Complexity	
$\mathbb{G}_1$ Exponentiation	–	$L+3$
$\mathbb{G}_2$ Exponentiation	–	$L+1$
$\mathbb{G}_T$ Exponentiation	$(n+1)(L+1)+3$	1
1 Single Pairing	$n+1$	$3L+4$
2-Miller Loop Pairings	$(n+1)(L-1)+2$	–
3-Miller Loop Pairings	$L-2$ if $L >= 2$	$L+1$
$(n+1)$-Miller Loop Pairings	–	–
Hash to $\mathbb{G}_1$	–	n
Overall complexity	$O(n*L)$	$O(n+L)$

Table 8.6: Verification speed comparison between CDD17 [20] and ENTRUST.

As noticed from the comparison in Table 8.6, the computational complexity of verification for CDD17 is $O(n*L)$, while ENTRUST achieves $O(n+L)$. This is because CDD17 needs to check the signatures of every single attribute for each level, whereas ENTRUST is able to solely verify the single batch signature.

To demonstrate the effect of the difference in complexity to both protocols, we show in Figure 8.5 the theoretical computation time to verify presentation tokens in both CDD17 and ENTRUST, using the operation times from Table 8.1.

We noticed that CDD17 did not optimize its performance by moving exponentiations to $\mathbb{G}_1$, but instead exponentiates more expensively in $\mathbb{G}_T$. To give a more fair comparison, we consider the computation time of exponentiating to $\mathbb{G}_T$ for the CDD17 scheme the same as exponentiating to $\mathbb{G}_1$.

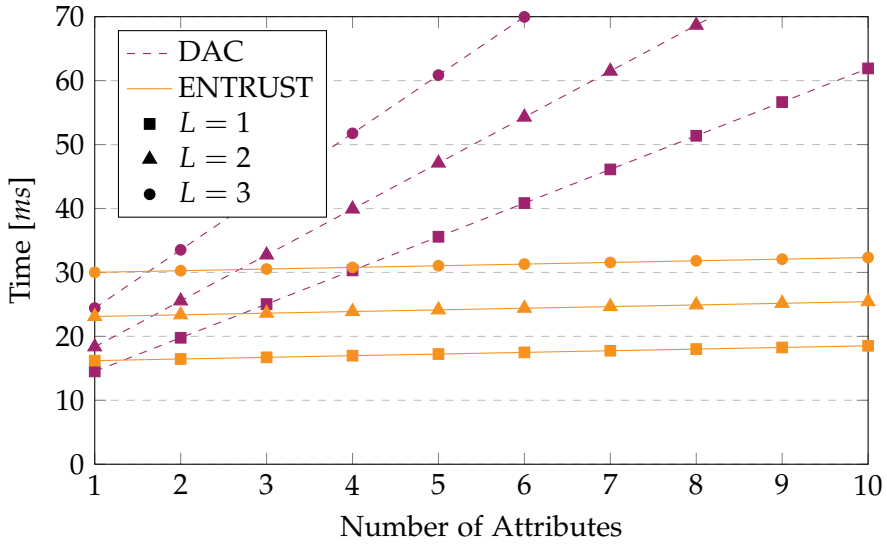

Figure 8.5: Comparison in verification time between CDD17 [20] and ENTRUST.

In Figure 8.5, we observe that DAC outperforms ENTRUST in verification speed for the first attributes. This is due to the difference in delegation technique. DAC incorporates *Group Alternation*, while ENTRUST uses *Double Key Pair* as delegation method (Section 5.3). Due to the *Double Key Pair* method, ENTRUST requires 2 extra pairing to check if each public keys pair hides the same secret exponent.

Nevertheless, the verification time of DAC grows sub-quadratically, while ENTRUST grows linearly to the number of attributes. When the verifier would precompute the hashed elements of the expected attributes, then the verification time would stay constant to the number of attributes. For 3 intermediate delegators, ENTRUST requires 258 μs for the verification of each additional attribute in the presentation token, while CDD17 requires 9108 μs.

8.6.2 *Presentation Token Size*

We also compare the size of the presentation tokens of both schemes, where only elements required for the verification of attributes and delegations are considered. Since users in CDD17 are able to directly generate presentation tokens, we consider a user on level-L equivalent to a prover in ENTRUST, which got its credential from delegator on level-$(L-1)$. Table 8.7 shows our comparison, where we only take into account the storage size within a presentation token required for the verification of attributes and delegations.

We observe from the comparison that both presentation tokens are equivalent in size when there are no attributes present. Nevertheless, the presentation token size of CDD17 grows sub-quadratically when

	CDD17 [20]	ENTRUST
	Without attributes	
Every odd level adds	$3\,\mathbb{G}_1 + 1\,\mathbb{G}_2$	$2\,\mathbb{G}_1 + 2\,\mathbb{G}_2$
Every even level adds	$1\,\mathbb{G}_1 + 3\,\mathbb{G}_2$	$2\,\mathbb{G}_1 + 2\,\mathbb{G}_2$
Total on an even level L	$2L\,\mathbb{G}_1 + 2(L-1)\,\mathbb{G}_2 + 2\,\mathbb{Z}_q$	$2L\,\mathbb{G}_1 + 2(L-1)\,\mathbb{G}_2 + 2\,\mathbb{Z}_q$
	With attributes	
Every odd-level attribute adds	$1\,\mathbb{G}_1$	-
Every even-level attribute adds	$1\,\mathbb{G}_2$	-
Total on an even level L	$(2L + \frac{1}{2}Ln)\,\mathbb{G}_1 + (2(L-1) + \frac{1}{2}Ln)\,\mathbb{G}_2 + 2\,\mathbb{Z}_q$	$(2L+1)\,\mathbb{G}_1 + (2(L-1))\,\mathbb{G}_2 + 2\,\mathbb{Z}_q$
Overall complexity	$O(n * L)$	$O(L)$

Table 8.7: Comparison in presentation token size between DAC and ENTRUST.

attributes are taken into account, while the presentation token size in ENTRUST is constant and is not dependent on the number of attributes. This is achieved by combining the aggregated attribute signatures into a single batch signature.

8.7 CONCLUSION

From the theoretical complexity of ENTRUST, we saw that the aggregation of attribute signatures allows both the computation and size complexity to be in the order of $O(n + L)$ instead of $O(n * L)$. When we compare the part responsible for the verification of attributes and delegation chain in ENTRUST to DAC, we clearly see the efficiency of the delegation scheme of ENTRUST. Due to the aggregation of attribute signatures into a single batch signature, the computational time to verify attribute signatures and the total attribute signatures size reduces from $O(n * L)$ to $O(1)$.

From the run-time experiments, we saw that presentation tokens containing 10 attributes and issued by 3 intermediate delegators could be generated in 143 ms, verified in 121 ms, and require 4312 bytes of storage space. The verification speed, in this case, lies within the challenging time bound of 300 ms for access control systems in public transportation. Although more attributes can be added to the presentation token while staying under the 300 ms boundary, we saw that the presentation token size grows linearly to the number of attributes. This is due to our choice to commit each value

individually in order to maintain complete delegator chain privacy, which results into n commitment signatures. Researchers have already proposed multiple commitment schemes where multiple values can be committed to a single commitment. The downside of these schemes is that values are committed sequentially and are bounded to an index value in the commitment. During selective disclosure, it would disclose information about the specific indices of the specific attributes. It would be interesting to apply non-sequential commitment schemes to ENTRUST in order to reduce the total commitment signature size.

9 DISCUSSION AND FUTURE WORK

The presence of the peer-based trust model within Self-Sovereign Identity motivates to apply the notion of consecutive trust for claim endorsement. Authorities are able to declare trust consecutively towards smaller entrusted entities, which are able to endorse claims for provers on behalf on the authority. This conception is interesting for various applications and settings, especially for large-scale organizations, which wish to delegate rights for credential issuance to lower-level entities in the hierarchy.

Current approaches supporting consecutive delegation of rights are only suitable for authentication and do not support credential issuance, while anonymous attribute-based credential schemes all rely on a single central entity for the issuance and management of credentials. Therefore, we attempt to offer a solution for consecutive delegation of signing rights for the issuance of anonymous attribute-based credentials. The main research question examined in this work was as follows:

> *How can we consecutively delegate signing rights for the issuance of attribute-based credentials, and construct practical, verifiable, and anonymous attribute-based proofs with delegation chain anonymity?*

In this chapter, we review the stated research question and discuss how the proposed protocol achieves the research goal. Furthermore, we discuss the limitations of our scheme and identify possible future improvements.

9.1 DISCUSSION

At the end of the introductory chapter, we have identified five sub-questions to our main research question. We are discussing and answering these questions in this section.

How can signing rights for attribute-based credentials be delegated in a more efficient manner opposed to existing schemes?

As seen in the complexity section of the evaluation chapter (Section 8.4), the delegation of signing rights is realized efficiently. The speed of delegating signing rights solely dependent on the number of attributes which the delegator wishes to delegate. Other schemes achieve this as well since delegators solely add their own signatures to the delegation bundle and do not perform any

computation on signatures from predecessors. However, delegators in ENTRUST aggregate their attribute signatures with the attribute signatures from its predecessors, which allows the storage size complexity of a delegation bundle in ENTRUST to be reduced to $O(n + L)$ instead of $O(n * L)$ in other schemes.

How can delegators issue valid attribute-based credentials on behalf of the root authority?

ENTRUST achieves this by incorporating delegation signatures, where delegators sign the public key of the delegatee to whom they wish to delegate selective signing rights. The delegation signatures will form a delegation chain, where the owner of the signed public key in one delegation signature becomes the signer in the next delegation signature. The verifier accepts the credential when the delegation signatures form a correct delegation chain.

Delegation chains alone are not sufficient to realize attribute-based credentials. Therefore, we introduce three signature schemes: RBLS, RSPS-M, and SRS-M, in order to realize attribute signatures, delegation signatures, and commitment signatures. To create these signature schemes, we modified three existing signature scheme to support randomization by multiplication. This type of randomization enables multiple of these different signatures to be linked together and allows flexible randomization even when signatures are aggregated together.

How can provers transform such a credential into proofs for anonymous attestation?

In our protocol, provers do not provide the signatures and the public keys from the credential to the verifier. Instead, provers generate Schnorr's zero-knowledge proofs of knowledge for each signature and public key. The presentation token containing all these proofs should convince the verifier that (1) the delegation chain in the credential is valid; (2) all intermediate delegators have the signing rights of the attributes used in the presentation token; (3) the presented predicates are valid under the attribute value in the signed commitment.

How can verifiers check the proofs without the knowledge of the identity of the intermediate delegators and the prover?

As discussed in the previous paragraph, the prover generates a presentation token with Schnorr's zero-knowledge proofs of knowledge for each signature and public key. While the verifier does not know the actual signatures and public keys, the verifier is still able to compute adapted verification pairings, which are compatible with the zero-knowledge proofs. During the creation of the zero-knowledge proofs, each concealed signature and public key

will result in a response element or response value. Since these response elements and values will be used multiple times in the verification pairings in a specific way to prove relations between group elements, the verifier can be sure that the credential is correct with a high probability.

Can we improve the efficiency of proof verification in terms of computational speed and storage size?

Since the verification phase should be as efficient as possible, we have employed multiple optimizations to reduce the computational time of verification pairings. Small benchmarks have shown an improvement of roughly 23-35% opposed to our non-optimized algorithm.

In our experiments, we have shown that ENTRUST can verify presentation tokens with 30 attributes under 300 ms considering 5 intermediate delegators. It is noteworthy that the verification time of these presentation tokens lies within the challenging time bound of 300 ms required for access control in public transportation systems [33, 54]. Therefore, the scheme achieves a certain degree of practical feasibility. The addition of each extra delegator to the credential solely adds $\sim$ 8 ms to the verification time.

A limitation of ENTRUST is that the presentation token size grows linearly depending on the number of attributes. Although we attempt to remove the linear dependency by aggregating the aggregated attributes signatures into a single batch signature in the presentation token, there are still commitment signatures present for every committed attribute value. It is already possible to commit multiple commitment values into a single commitment, which allow the presentation token to contain a single commitment signature. However, this does not comply with our complete privacy requirement, since the values are committed sequentially, which could allow verifiers to correlate two presentations of the same credential more accurately. Future work should determine whether multiple values can be committed to a single commitment without compromising the complete privacy of the prover.

9.2 FUTURE WORK

The proposed protocol forms a building block towards more flexible and advanced digital identities. Although this work attempts to address numerous properties required for the realization of anonymous credentials created by delegatable signing rights, there are several additional properties which are desired for anonymous credentials. We introduce several of these properties and state the opportunities to improve the limitations of our protocol.

Delegation chain auditability In case of unusual behavior or suspicion of fraud with anonymous credentials, verifiers should be able to hand over the presented presentation token to an inspector party, which will be to revoke the anonymity of the prover and intermediate delegators. This allows the inspector to check if the credential has been correctly issued and who is responsible for potential misbehavior.

Credential revocation The validity of delegation bundles and anonymous credentials should be able to be revoked by the root authority or issuing entities when necessary. Ideally, neither the prover nor the verifier should interact with the root authority before or during verification. The main challenge is to efficiently achieve the revocation of credentials since communication with the root authority is unavoidable to fetch the latest revocation information.

There are multiple approaches to achieve revocability of credentials. The verifier could beforehand periodically fetch lists or accumulators of either valid or revoked credentials, and check during verification if some element in the presentation token occurs in the lists or accumulators. Besides the overhead of continuous fetching of the verifier, it could be possible this way for the verifier to correlate multiple presentations of the same credentials, since there is some sort of identifier present in the presentation token.

The other approach is to let provers convince verifiers that their credential has not been revoked in the most recent list or accumulator, and therefore have a valid credential. The downside of this approach is that provers need to contact the root authority every time they wish to generate a presentation token.

Incorporating non-sequential multi-message commitments In ENTRUST, the prover creates commitments over every single attribute value which will be issued by the issuing delegator. This results in n commitment signatures in the credential. As already mentioned in our discussion of the last sub-question of the main research question, it would be more efficient to commit multiple values into a single commitment, as the credential will result in just one commitment signature. Researchers have already proposed multiple commitment schemes where multiple values can be committed into one commitment. The downside of these schemes is that values are committed sequentially and are bounded to an index value in the commitment. During selective disclosure, this would disclose information about the specific indices of the attributes. Because ENTRUST preserves privacy

whenever possible, we decided to use single-message commitments. It would be interesting to apply non-sequential multi-message commitments to ENTRUST, such that only one commitment signature is needed and complete privacy will be preserved.

Aggregation of the delegation signatures In this work, the attribute signatures are aggregated in order to reduce the credential size. However, it would also be interesting to see whether the delegation signature could also be aggregated, while maintaining the same security as the current scheme.

Immutable attribute values during delegation Blömer and Bobolz [12] introduced *Delegatable Attribute-Based Anonymous Credentials from Dynamically Malleable Signatures*, where selective attribute values are assigned and made immutable in the credential during delegation. It would be interesting to see whether this could apply to the signing rights in ENTRUST since this allows delegators to restrict the subsequent delegatees to issue restricted attribute values for particular attributes.

9.3 CONCLUSION

The primary goal of this thesis was to realize a concrete and practical instantiation for delegatable signing rights for the issuance of anonymous attribute-based credentials. Previous work already attempts to achieve anonymity in a consecutive delegation setting but lacks practical efficiency due to expensive zero-knowledge proofs. In addition, many schemes do not involve attribute-based credentials, and many of them only focus on authentication. This thesis presents a protocol with incorporates both anonymous attestation with attribute-based credentials and the delegation of signing rights for the issuance of these credentials.

Since none of the existing signatures schemes met our requirements derived from our protocol design, we introduced three signature schemes based on existing work. By incorporating these preliminary signature schemes as fundamental building blocks, we realize the protocol for delegatable signing rights for the issuance of anonymous attribute-based credentials. This thesis provides a concrete instantiation of the protocol and shows practical feasibility for the verification of presentation tokens. In most use cases, presentation tokens can be verified within 300 ms, which complies to challenging time bound of access control in public transportation systems [33, 54].

The proposed scheme is not limited to the application on large-scale organizations for the hierarchical issuance of credentials.

The scheme can already be applied in a Self-Sovereign Identity setting where relevant authorities declare trust consecutively towards trustees, and individuals are able to obtain endorsements on claims from the trustees on behalf of the authority. Due to the nature of the peer-to-peer economy of trust in Self-Sovereign Identity, it is likely that the scheme can be applied for other diverse scenarios in Self-Sovereign Identity as well. Nevertheless, the presented protocol forms a building block towards more flexible and advanced digital identities.

BIBLIOGRAPHY

[1] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. "Structure-Preserving Signatures and Commitments to Group Elements." In: *CRYPTO*. Vol. 6223. Lecture Notes in Computer Science. Springer, 2010, pp. 209–236.

[2] Man Ho Au, Willy Susilo, and Yi Mu. "Constant-Size Dynamic *k*-TAA." In: *SCN*. Vol. 4116. Lecture Notes in Computer Science. Springer, 2006, pp. 111–125.

[3] Australian Government. *Australian Community Attitudes to Privacy Survey 2017*. https://www.oaic.gov.au/resources/engage-with-us/community-attitudes/acaps-2017/acaps-infographic.pdf. [Online; accessed 17-Sep-2018].

[4] Paulo S. L. M. Barreto and Michael Naehrig. "Pairing-Friendly Elliptic Curves of Prime Order." In: *Selected Areas in Cryptography*. Vol. 3897. Lecture Notes in Computer Science. Springer, 2005, pp. 319–331.

[5] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. "P-signatures and Noninteractive Anonymous Credentials." In: *TCC*. Vol. 4948. Lecture Notes in Computer Science. Springer, 2008, pp. 356–374.

[6] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. "Randomizable Proofs and Delegatable Anonymous Credentials." In: *CRYPTO*. Vol. 5677. Lecture Notes in Computer Science. Springer, 2009, pp. 108–125.

[7] Mihir Bellare and Phillip Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols." In: *ACM Conference on Computer and Communications Security*. ACM, 1993, pp. 62–73.

[8] Mihir Bellare and Phillip Rogaway. "The Exact Security of Digital Signatures - How to Sign with RSA and Rabin." In: *EUROCRYPT*. Vol. 1070. Lecture Notes in Computer Science. Springer, 1996, pp. 399–416.

[9] David Bernhard, Georg Fuchsbauer, Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. "Anonymous attestation with user-controlled linkability." In: *Int. J. Inf. Sec.* 12.3 (2013), pp. 219–249.

[10] Jean-Luc Beuchat, Jorge Enrique González-Díaz, Shigeo Mitsunari, Eiji Okamoto, Francisco Rodríguez-Henríquez, and Tadanori Teruya. "High-Speed Software Implementation of the Optimal Ate Pairing over Barreto-Naehrig Curves." In: *Pairing*. Vol. 6487. Lecture Notes in Computer Science. Springer, 2010, pp. 21–39.

[11] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. "Get Shorty via Group Signatures without Encryption." In: *SCN*. Vol. 6280. Lecture Notes in Computer Science. Springer, 2010, pp. 381–398.

[12] Johannes Blömer and Jan Bobolz. "Delegatable Attribute-Based Anonymous Credentials from Dynamically Malleable Signatures." In: *ACNS*. Vol. 10892. Lecture Notes in Computer Science. Springer, 2018, pp. 221–239.

[13] Dan Boneh, Xavier Boyen, and Hovav Shacham. "Short Group Signatures." In: *CRYPTO*. Vol. 3152. Lecture Notes in Computer Science. Springer, 2004, pp. 41–55.

[14] Dan Boneh, Manu Drijvers, and Gregory Neven. "Compact Multi-Signatures for Smaller Blockchains." In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 483.

[15] Dan Boneh, Ben Lynn, and Hovav Shacham. "Short Signatures from the Weil Pairing." In: *ASIACRYPT*. Vol. 2248. Lecture Notes in Computer Science. Springer, 2001, pp. 514–532.

[16] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps." In: *EUROCRYPT*. Vol. 2656. Lecture Notes in Computer Science. Springer, 2003, pp. 416–432.

[17] Fabrice Boudot. "Efficient Proofs that a Committed Number Lies in an Interval." In: *EUROCRYPT*. Vol. 1807. Lecture Notes in Computer Science. Springer, 2000, pp. 431–444.

[18] Xavier Boyen and Brent Waters. "Full-Domain Subgroup Hiding and Constant-Size Group Signatures." In: *Public Key Cryptography*. Vol. 4450. Lecture Notes in Computer Science. Springer, 2007, pp. 1–15.

[19] Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. "Efficient Protocols for Set Membership and Range Proofs." In: *ASIACRYPT*. Vol. 5350. Lecture Notes in Computer Science. Springer, 2008, pp. 234–252.

[20] Jan Camenisch, Manu Drijvers, and Maria Dubovitskaya. "Practical UC-Secure Delegatable Credentials with Attributes and Their Application to Blockchain." In: *CCS*. ACM, 2017, pp. 683–699.

[21] Jan Camenisch and Thomas Groß. "Efficient Attributes for Anonymous Credentials." In: *ACM Trans. Inf. Syst. Secur.* 15.1 (2012), 4:1–4:30.

[22] Jan Camenisch and Els Van Herreweghen. "Design and implementation of the *idemix* anonymous credential system." In: *ACM Conference on Computer and Communications Security*. ACM, 2002, pp. 21–30.

[23] Jan Camenisch and Anna Lysyanskaya. "An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation." In: *EUROCRYPT*. Vol. 2045. Lecture Notes in Computer Science. Springer, 2001, pp. 93–118.

[24] Jan Camenisch and Anna Lysyanskaya. "Signature Schemes and Anonymous Credentials from Bilinear Maps." In: *CRYPTO*. Vol. 3152. Lecture Notes in Computer Science. Springer, 2004, pp. 56–72.

[25] Melissa Chase and Anna Lysyanskaya. "On Signatures of Knowledge." In: *CRYPTO*. Vol. 4117. Lecture Notes in Computer Science. Springer, 2006, pp. 78–96.

[26] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. "Malleable Signatures: Complex Unary Transformations and Delegatable Anonymous Credentials." In: *IACR Cryptology ePrint Archive* 2013 (2013), p. 179.

[27] Sanjit Chatterjee and Alfred Menezes. "On cryptographic protocols employing asymmetric pairings - The role of Ψ revisited." In: *Discrete Applied Mathematics* 159.13 (2011), pp. 1311–1322.

[28] Liqun Chen, Dan Page, and Nigel P. Smart. "On the Design and Implementation of an Efficient DAA Scheme." In: *CARDIS*. Vol. 6035. Lecture Notes in Computer Science. Springer, 2010, pp. 223–237.

[29] Christopher Allen. *The Path to Self-Sovereign Identity*. http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html. [Online; accessed 17-May-2018]. 2016.

[30] Dan Gisolfi - IBM. *Self-sovereign identity: Our recent activity as a Sovrin Steward*. https://www.ibm.com/blogs/blockchain/2018/05/self-sovereign-identity-our-recent-activity-as-a-sovrin-steward/. [Online; accessed 24-Sep-2018].

[31] Darkreading. *Digital Identity Makes Headway Around the World*. https://www.darkreading.com/endpoint/authentication/digital-identity-makes-headway-around-the-world/a/d-id/1331576. [Online; accessed 24-Sep-2018].

[32] Hans Delfs and Helmut Knebl. *Introduction to Cryptography: Principles and Applications*. Information Security and Cryptography. Springer, 2002.

[33] Nicolas Desmoulins, Roch Lescuyer, Olivier Sanders, and Jacques Traoré. "Direct Anonymous Attestations with Dependent Basename Opening." In: *CANS*. Vol. 8813. Lecture Notes in Computer Science. Springer, 2014, pp. 206–221.

[34] Diego F. Aranha and Conrado P. L. Gouvêa. *RELIC Cryptographic Meta-Toolkit - Master branch*. https://github.com/relic-toolkit/relic. [Online; accessed 03-Sep-2018].

[35] Diego F. Aranha. *Git Commit - "Add curve B12_P381 generated by ZCash team."* https://github.com/relic-toolkit/relic/commit/c78652d850f28870113f1f522a4bfd095d5e2579. [Online; accessed 17-Sep-2018].

[36] Diego F. Aranha. *Pairings are not dead, just resting - ECC 2017*. https://ecc2017.cs.ru.nl/slides/ecc2017-aranha.pdf. [Online; accessed 17-Sep-2018].

[37] ECRYPT II. *D.MAYA.6 - Final Report on Main Computational Assumptions in Cryptography*. http://www.ecrypt.eu.org/ecrypt2/documents/D.MAYA.6.pdf. [Online; accessed 13-Oct-2018].

[38] European Union. *General Data Protection Regulation (GDPR)*. https://www.eugdpr.org. [Online; accessed 22-May-2018]. 2016.

[39] Uriel Feige, Amos Fiat, and Adi Shamir. "Zero-Knowledge Proofs of Identity." In: *J. Cryptology* 1.2 (1988), pp. 77–94.

[40] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems." In: *CRYPTO*. Vol. 263. Lecture Notes in Computer Science. Springer, 1986, pp. 186–194.

[41] Free Software Foundation. *The GNU Multiple Precision Arithmetic Library*. https://gmplib.org/. [Online; accessed 17-Sep-2018].

[42] Georg Fuchsbauer. "Commuting Signatures and Verifiable Encryption." In: *EUROCRYPT*. Vol. 6632. Lecture Notes in Computer Science. Springer, 2011, pp. 224–245.

[43] Georg Fuchsbauer and David Pointcheval. "Anonymous Proxy Signatures." In: *SCN*. Vol. 5229. Lecture Notes in Computer Science. Springer, 2008, pp. 201–217.

[44] Georg Fuchsbauer and David Pointcheval. "Anonymous Consecutive Delegation of Signing Rights: Unifying Group and Proxy Signatures." In: *Formal to Practical Security*. Vol. 5458. Lecture Notes in Computer Science. Springer, 2009, pp. 95–115.

[45] Georg Fuchsbauer and David Pointcheval. "Proofs on Encrypted Values in Bilinear Groups and an Application to Anonymity of Signatures." In: *Pairing*. Vol. 5671. Lecture Notes in Computer Science. Springer, 2009, pp. 132–149.

[46] José María de Fuentes, Lorena González-Manzano, Agusti Solanas, and Fatbardh Veseli. "Attribute-Based Credentials for Privacy-Aware Smart Health Services in IoT-Based Smart Cities." In: *IEEE Computer* 51.7 (2018), pp. 44–53.

[47] GSM Association (GSMA). *Digital Identity: What to Expect in 2018*. https://www.gsma.com/identity/digital-identity-expect-2018. [Online; accessed 22-May-2018]. 2018.

[48] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. "Pairings for cryptographers." In: *Discrete Applied Mathematics* 156.16 (2008), pp. 3113–3121.

[49] Gemalto. *ePassport solutions: for seamless electronic passport migrations*. https://www.gemalto.com/govt/travel. [Online; accessed 24-Sep-2018].

[50] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract)." In: *STOC*. ACM, 1985, pp. 291–304.

[51] Google. *Google Benchmark Library*. https://github.com/google/benchmark. [Online; accessed 17-Sep-2018].

[52] Jens Groth. "Efficient Fully Structure-Preserving Signatures for Large Messages." In: *ASIACRYPT (1)*. Vol. 9452. Lecture Notes in Computer Science. Springer, 2015, pp. 239–259.

[53] Jens Groth and Amit Sahai. "Efficient Non-interactive Proof Systems for Bilinear Groups." In: *EUROCRYPT*. Vol. 4965. Lecture Notes in Computer Science. Springer, 2008, pp. 415–432.

[54] Gesine Hinterwälder, Christian T. Zenger, Foteini Baldimtsi, Anna Lysyanskaya, Christof Paar, and Wayne P. Burleson. "Efficient E-Cash in Practice: NFC-Based Payments for Public Transportation Systems." In: *Privacy Enhancing Technologies*. Vol. 7981. Lecture Notes in Computer Science. Springer, 2013, pp. 40–59.

[55] Hyperledger. *Hyperledger Indy*. https://www.hyperledger.org/projects/hyperledger-indy. [Online; accessed 22-May-2018].

[56] IBM. *Identity Mixer*. https://www.zurich.ibm.com/identity_mixer/. [Online; accessed 22-May-2018].

[57] ISO/IEC. *ISO/IEC 24760-1: Information technology – Security techniques – A framework for identity management – Part 1: Terminology and concepts*. https://www.iso.org/standard/57914.html. [Online; accessed 24-Sep-2018].

[58] Neal Koblitz and Alfred J. Menezes. "The random oracle model: a twenty-year retrospective." In: *Des. Codes Cryptography* 77.2-3 (2015), pp. 587–610.

[59] Maria Dubovitskaya - IBM. *Jira Ticket - FABJ-327 - Idemix - Java SDK MVP*. https://jira.hyperledger.org/browse/FABJ-327. [Online; accessed 24-Sep-2018].

[60] Microsoft. *U-Prove*. https://www.microsoft.com/en-us/research/project/u-prove/. [Online; accessed 22-May-2018].

[61] Alexander Mühle, Andreas Grüner, Tatiana Gayvoronskaya, and Christoph Meinel. "A Survey on Essential Components of a Self-Sovereign Identity." In: *CoRR* abs/1807.06346 (2018).

[62] Kun Peng and Feng Bao. "An Efficient Range Proof Scheme." In: *SocialCom/PASSAT*. IEEE Computer Society, 2010, pp. 826–833.

[63] Phil Zimmermann. *PGP User's Guide, Volume I: Essential Topics*. https://web.pa.msu.edu/reference/pgpdoc1.html. [Online; accessed 24-Sep-2018].

[64] David Pointcheval and Olivier Sanders. "Short Randomizable Signatures." In: *CT-RSA*. Vol. 9610. Lecture Notes in Computer Science. Springer, 2016, pp. 111–126.

[65] Johan Pouwelse, André de Kok, Joost Fleuren, Peter Hoogendoorn, Raynor Vliegendhart, and Martijn de Vos. "Laws for Creating Trust in the Blockchain Age." In: *European Property Law Journal* 6.3 (2017), pp. 321–356.

[66] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems." In: *Commun. ACM* 21.2 (1978), pp. 120–126.

[67] Claus-Peter Schnorr. "Efficient Identification and Signatures for Smart Cards." In: *CRYPTO*. Vol. 435. Lecture Notes in Computer Science. Springer, 1989, pp. 239–252.

[68] Sovrin Foundation. *Sovrin Website*. https://sovrin.org. [Online; accessed 22-May-2018]. 2018.

[69] Sovrin Foundation. *Sovrin™: A Protocol and Token for Self Sovereign Identity and Decentralized Trust - White Paper*. https://sovrin.org/wp-content/uploads/2018/03/Sovrin-Protocol-and-Token-White-Paper.pdf. [Online; accessed 22-May-2018]. 2018.

[70] Quinten Stokkink and Johan Pouwelse. "Deployment of a Blockchain-Based Self-Sovereign Identity." In: *CoRR* abs/1806.01926 (2018).

[71] TRUSTe/NCSA. *2016 Consumer Privacy Infographic – US Edition*. https://www.trustarc.com/resources/privacy-research/ncsa-consumer-privacy-index-us/. [Online; accessed 17-Sep-2018].

[72] The Guardian. *Cambridge Analytica Data Scandal News Overview*. https://www.theguardian.com/uk-news/cambridge-analytica. [Online; accessed 17-Sep-2018].

[73] Tommy Koens and Stijn Meijer. *Matching Identity Management Solutions to Self-Sovereign Identity Principles*. https://www.slideshare.net/TommyKoens/matching-identity-management-solutions-to-selfsovereign-identity-principles/1. [Online; accessed 24-Sep-2018].

[74] Trend Micro. *2012 Linkedin Breach had 117 Million Emails and Passwords Stolen, Not 6.5M*. https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/2012-linkedin-breach-117-million-emails-and-passwords-stolen-not-6-5m. [Online; accessed 17-Sep-2018].

[75] Mårten Trolin and Douglas Wikström. "Hierarchical Group Signatures." In: *ICALP*. Vol. 3580. Lecture Notes in Computer Science. Springer, 2005, pp. 446–458.

[76] World Economic Forum. *Digital Identity – Why It Matters and Why It's Important We Get It Right*. https://www.weforum.org/press/2018/01/digital-identity-why-it-matters-and-why-it-s-important-we-get-it-right/. [Online; accessed 22-May-2018]. 2018.

[77] World Wide Web Consortium (W3C). *Verifiable Claims Data Model and Representations - Working Draft*. https://www.w3.org/TR/verifiable-claims-data-model. [Online; accessed 9-May-2018]. 2017.

[78] World Wide Web Consortium (W3C). *Verifiable Credentials Data Model 1.0 - Editors Draft: Privacy Spectrum*. https://w3c.github.io/vc-data-model/diagrams/privacy-spectrum.svg. [Online; accessed 22-May-2018]. 2018.

[79] World Wide Web Consortium (W3C). *Verifiable Credentials Data Model 1.0 - Editors Draft*. https://w3c.github.io/vc-data-model/. [Online; accessed 22-May-2018]. 2018.

[80] ZCash. *BLS12-381: New zk-SNARK Elliptic Curve Construction*. https://blog.z.cash/new-snark-curve/. [Online; accessed 17-Sep-2018].

A

PRIMITIVE EXISTING SIGNATURE SCHEMES

A.1 BLS MULTI-SIGNATURES

Boneh et al. [16] designed a signature scheme, where n signatures on n distinct message from n distinct signers can be combined into a single short signature. The scheme reduces the total signature size and the number of pairing during the verification phase. The signature scheme relies on hash functions, which are viewed as random oracles [8].

The BLS Signature scheme is as follows:

$\mathsf{KeyGen}()$: Choose random $x \xleftarrow{R} \mathbb{Z}_q$ and set $h \leftarrow g_2^x \in \mathbb{G}_2$. Output: $pk := h$ and $sk := x$.

$\mathsf{Sign}(sk, m)$: Output: $\sigma = H_1(m)^x \in \mathbb{G}_1$.

$\mathsf{Verify}(pk, m, \sigma)$: If $e(\sigma, g_2) = e(H_1(m), pk)$ output 1 otherwise 0.

A.1.1 *Signature Aggregation*

BLS Signatures can be aggregated together into a single signature. Given (pk_i, m_i, σ_i), where $i = 1, \ldots, n$, $m_1, \ldots, m_n$ are all distinct messages and $\sigma_1, \ldots, \sigma_n \in \mathbb{G}1$, the signatures can be combined into one short signature:

$$\sigma \leftarrow \sigma_1 \cdots \sigma_n \in \mathbb{G}_1 \,.$$

The aggregated signature can be verified by checking:

$$e(\sigma, g_2) \stackrel{?}{=} e\left(H_1(m_1), pk_1\right) \cdots e\left(H_1(m_n), pk_n\right) \,.$$

A disadvantage of the BLS Multi-Signature scheme is the requirement where all messages have to be distinct. When all messages $m_1, \ldots, m_n$ would be the same, the verification can be reduced to computing just two pairings:

$$e(\sigma, g_2) \stackrel{?}{=} e\left(H_1(m), pk_1 \cdots pk_n\right) \,.$$

However, this leads to the need to multiply public-keys together. Multiplying public-keys is susceptible to a rogue public-key attack, which will be discussed next.

A.1.2 *Rogue Public-Key Attack*

In the rogue public-key attack, an adversary chooses a random secret $x_2 \xleftarrow{R} \mathbb{Z}_q$, and computes $pk_2 \leftarrow g_2^{x_2} \in \mathbb{G}_2$. Given a known public-key pk_1, the adversary can compute a rogue public-key $pk_R \leftarrow pk_2/pk_1$. Instead of publishing pk_2, the adversary will publish pk_R as its public-key.

The adversary computes $\sigma = H(m)^{x_2}$. The adversary is now able to convince others that both pk_1 and pk_R have signed a message m, without the knowledge of secret key of pk_1. The verification of the "aggregate" signature holds, because:

$$\begin{aligned} e(H(m), pk_1 pk_R) &= e(H(m), g_2^{x_1} g_2^{x_2 - x_1}) \\ &= e(H(m), g_2^{x_2}) \\ &= e(H(m)^{x_2}, g_2) \\ &= e(\sigma, g_2)\,. \end{aligned}$$

Therefore, this attack can be prevented by enforcing the distinctness of the messages, since the secret exponents are kept separate by the different base elements. This attack can also be prevented by requiring a proof of knowledge, that proofs that the public-key holder knows the secret exponent of the published public-key. However, this method requires an unnecessary overhead of zero-knowledge proofs and computation.

A.2 BLS MULTI-SIGNATURES WITH PUBLIC-KEY AGGREGATION

Boneh et al. [14] proposed a solution which allows multiple signers to sign the same message and aggregate the signatures together while preventing the rogue key attack. Every signature will be exponentiated with a hash value containing all public-keys of the signers before aggregation. During verification, the verifier can verify the signature by computing a aggregated public-key $apk \leftarrow \prod_{i=1}^{n} pk_i^{H(pk_i, \{pk_1, \ldots, pk_n\})}$. Because all public-keys are first exponentiated with a random hash value, the public-keys cannot be combined anymore as in the rogue key attack. The MPS signature scheme is as follows:

$\mathsf{KeyGen}()$: Choose random $x \xleftarrow{R} \mathbb{Z}_q$ and set $h \leftarrow g_2^x \in \mathbb{G}_2$. Output: $pk := h$ and $sk := x$.

$\mathsf{Sign}(sk, m)$: Output: $\sigma = H_1(m)^x \in \mathbb{G}_1$.

$\mathsf{Aggregate}(\langle pk_i, \sigma_i \rangle_{i=1}^n)$: Output: $\sigma \leftarrow \prod_{i=1}^{n} \sigma_i^{H(pk_i, \{pk_1, \ldots, pk_n\})}$

$\mathsf{Verify}(\langle pk_i \rangle_{i=1}^n, m, \sigma)$: Compute $apk \leftarrow \prod_{i=1}^{n} pk_i^{H_1(pk_i, \{pk_1, \ldots, pk_n\})}$. If $e(\sigma, g_2) = e(H_1(m), apk)$ output 1 otherwise 0.

A.2.1 *Batch Verification*

Given (apk_i, m_i, σ_i), where $i = 1, \ldots, n$, $m_1, \ldots, m_n$ are all distinct messages and $\sigma_1, \ldots, \sigma_n \in \mathbb{G}_1$ are aggregated signatures, the signatures can be combined into a single batch signature:

$$\sigma \leftarrow \sigma_1 \cdots \sigma_n \in \mathbb{G}_1 .$$

The aggregated batch signature can be batch verified by checking:

$$e(\sigma, g_2) \stackrel{?}{=} e\left(H_1(m_1), apk_1\right) \cdots e\left(H_1(m_n), apk_n\right) .$$

Notice here that if all aggregated public-keys $apk_1 = \cdots = apk_n$ are the same, then the verification can be reduced to two pairings:

$$e(\sigma, g_2) \stackrel{?}{=} e\left(H_1(m_1) \cdots H_1(m_n), apk\right) .$$

A.3 STRUCTURE-PRESERVING SIGNATURES

Structure-preserving signatures [1] are pairing-based signatures where the verification keys, messages, and signatures all consist solely of group elements. This allows the verification to be done by computing and comparing pairings and multiplications. In fully structure-preserving signature schemes, the secret key also only consists of groups elements.

Groth [52] has defined a structure-preserving signature scheme and a fully structure-preserving signature scheme for large messages that allows the signing of $N = n \cdot m$ group elements with $(n + 2)$ signatures. The protocol can either create randomizable or strongly existentially unforgeable signatures.

Camenisch et al. [20] simplifies the structure-preserving signature scheme of Groth to sign n group elements resulting in $(n + 2)$ signatures, and uses the randomizability property. The simplified scheme is as follows:

$\mathsf{Setup}()$: Choose random $y_i \stackrel{R}{\leftarrow} \mathbb{G}_2$ for $i = 1, \ldots, n$. Output: $y_1, \ldots, y_n$.

$\mathsf{Gen}()$: Choose random $v \stackrel{R}{\leftarrow} \mathbb{Z}_q$ and set $V \leftarrow g_2^x \in \mathbb{G}_2$. Output: $pk := V$ and $sk := v$.

$\mathsf{Sign}(sk, \vec{m})$: Choose random $r \stackrel{R}{\leftarrow} \mathbb{Z}_q^*$ and set $R \leftarrow g_1^r$, $S \leftarrow (y_1 \cdot g_2^v)^{1/r}$, and $T_i \leftarrow (y_i^v \cdot m_i)^{1/r}$. Output: $\sigma = (R, S, T_1, \ldots, T_n)$.

$\mathsf{Verify}(vk, \sigma, \vec{m})$: For $\sigma = (R, S, T_1, \ldots, T_n)$, if $e(R, S) = e(g_1, y_1)\ e(V, g_2)$ and $e(R, T_i) = e(V, y_i)\ e(g_1, m_i)$ for $i = 1, \ldots, n$, then output 1 otherwise 0.

$\mathsf{Rand}(\sigma)$: For $\sigma = (R, S, T_1, \ldots, T_n)$, choose a random $r' \stackrel{R}{\leftarrow} \mathbb{Z}_q^*$. Let $R' \leftarrow R^{r'}$, $S' \leftarrow S^{1/r'}$ and $T_i' \leftarrow T_i^{1/r'}$. Output $\sigma' = (R', S', T_1', \ldots, T_n')$.

A.4 SHORT RANDOMIZABLE SIGNATURES

Pointcheval and Sanders [64] came up with a structure-preserving signature scheme as a more efficient alternative to CL-signatures [24]. It allows a user to get a signature on a secret value, and prove the secret value multiple times without linkability of the provided proofs. Realizing this scenario with conventional signature schemes require the prover to commit the signature, and carry out a zero-knowledge proof of knowledge about the commitment opening to a signature on the secret value. With PS-signatures, the prover can randomize the signatures and send those to the verifier to prove that the signature is valid on the secret value while remaining unlinkable. Unlinkability relies on the DDH assumption in $\mathbb{G}_1$ (Assumption 3), which lead to the requirement to use asymmetric pairings.

$\mathsf{Keygen}()$: Choose random $h \xleftarrow{R} \mathbb{G}_1, \tilde{h} \xleftarrow{R} \mathbb{G}_2$ and $(x, y) \xleftarrow{R} \mathbb{Z}_q$. Compute $(X, Y) \leftarrow (h^x, h^y)$ and $(\tilde{X}, \tilde{Y}) \leftarrow (\tilde{h}^x, \tilde{h}^y)$, set $sk \leftarrow X$ and $pk \leftarrow (h, Y, \tilde{h}, \tilde{X}, \tilde{Y})$.

$\mathsf{Commit}(m)$: Choose random $t \xleftarrow{R} \mathbb{Z}_q$, and compute $C \leftarrow h^t Y^m$. Output: (C, t).

$\mathsf{Sign}(sk, C)$: Choose random $u \xleftarrow{R} \mathbb{Z}_q$, and outputs $\sigma' \leftarrow (h^u, (XC)^u)$.

$\mathsf{Unblind}(\sigma', t)$: $\sigma' = (\sigma'_1, \sigma'_2)$. Output $\sigma \leftarrow (\sigma'_1, \sigma'_2 / \sigma'^t_1)$.

$\mathsf{Verify}(pk, \sigma, m)$: Parse $\sigma = (\sigma_1, \sigma_2)$. If $\sigma_1 \neq 1_{\mathbb{G}_1}$ and $e(\sigma_1, \tilde{X} \cdot \tilde{Y}^m) = e(\sigma_2, \tilde{h})$, output 1 otherwise 0.

$\mathsf{Randomize}(\sigma)$: Parse $\sigma = (\sigma_1, \sigma_2)$. Choose random $t \xleftarrow{R} \mathbb{Z}_q$ and output: $\sigma' \leftarrow (\sigma_1^t, \sigma_2^t)$

$\mathsf{GenPoK}(\sigma)$: Parse $\sigma = (\sigma_1, \sigma_2)$. Choose random $r, t \xleftarrow{R} \mathbb{Z}_q$. Compute $\sigma' \leftarrow (\sigma_1^r, (\sigma_2 \cdot \sigma_1^t)^r)$ and output (t, σ').