Forward-looking consistency in Attribute-Based Credentials

A privacy-preserving way to determine the revocation status of credentials after presentation

by

C. van Bruggen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday Nov 24, 2020 at 14:00.

Student number:	4234863	
Project duration:	July 1, 2019 – Nov 24, 2020	
Thesis committee:	Dr. Zekeriya Erkin,	TU Delft, chair
	Dr. Huijuan Wang,	TU Delft
	Dr. Ir. Oskar van Deventer,	TNO

Abstract

Authentication mechanisms play an important role in everyday digital interactions and allow users to prove their identity to others. Privacy-preserving Attribute-Based Credential Systems (PABCSs) allow users to authenticate by presenting their credential attributes, while multiple presentations remain unlinkable and untraceable. A revocation mechanism allows the credential issuer to revoke a credential, for example when the attributes of a user change. Verifiers can in turn determine the revocation status of the presented credential.

This thesis considers situations in which a verifier needs to determine the revocation status of a credential after the presentation. This stronger revocation level, also described as forward-looking consistency, has not been researched before in PABCSs. To address this gap, we present the PABC-FLC design, a PABCS with forward-looking consistency. Our design allows users to remain offline after presentation, without compromising on the privacy properties. To avoid an impossibility result, we introduce a new participant called the Non-Revocation Prover (NRP). The NRP facilitates a verifier in determining the revocation status of a credential, even though neither the verifier nor the issuer have to trust the NRP.

We show that our PABC-FLC design has a comparable experimental runtime to a credential system without revocation mechanism. This experiment shows our design has a runtime overhead of +20 ms (+32%) during presentation and +20 ms (+34%) during verification. Concluding that our PABC-FLC design is a feasible PABCS with forward-looking consistency, while remaining unlinkable and untraceable.

Preface

Working on this thesis turned out to be a long road. I count myself fortunate that I did not have to walk it alone, but was helped by many others along the way. That is why I want to take this oppurtunity to thank those that contributed to this thesis.

First and foremost, I want to thank my supervisors Zeki, Oskar and Rieks for sticking with me throughout this process. The past year has been quite eventful, but your support, personal advice and motivating words kept me going to finish my thesis. I also want to thank Oskar and Rieks for all the insightful discussions we have had and for giving me the opportunity to work at TNO. I want to thank Zeki for his critical, but always constructive, feedback and for creating an environment where students can learn to do research.

I am grateful for the countless people that I met while working on this thesis: Bart, Ramon, Roy, Sanne, Michiel and Jan Gerrit at TNO and Bartosz, Dieuwke, Felix, Jehan, Maurits, Sandra, Sharif, Seu, Tim the TU Delft. The lunchtime walks and coffee breaks I had with all of you made this process a lot more enjoyable. A special thanks to Chiel, Seu and Bartosz for their feedback on my thesis. I also want to thank Huijuan for taking the time to review my work as a committee member.

Finally, I want to thank my family and friends for their support, as I completed this thesis and my studies.

C. van Bruggen
Delft, November 2020

Contents

1 Introduction

1.1. Anonymous Credentials

The publication of David Chaums article “Security without identification" [23] can be seen as the beginning of the field of *Anonymous Credentials*. A credential is a collection of attributes that describe a user. Issuing organizations can *issue* credentials to the credential users. The users can *present* their credentials to verifying organizations as a form of authentication. If both the issuer and verifier refer to the user using pseudonyms, instead of a single unique identifier, we call these *anonymous credentials*. To give an example, consider a citizen who obtains a date-of-birth credential from the government. This credential can be presented to a shop owner when buying alcohol, to prove that the buyer is of age. Because both the government and shop owner use a different pseudonym to refer to the user, they are unable to correlate the activity of the user across their organizations. At the same time these credential provide security, a credential can only be obtained from a credential issuer and can only be presented by the user that obtained it. Therefore, these credentials provide security without identification. Although we refer to these credentials as *anonymous credentials*, we will continue with the term Attribute-Based Credential (ABC). We consider this a more accurate description as the attributes revealed during a credential presentation, like a date of birth, *can* be used to uniquely identify a user. We describe the schemes and protocols that provide this functionality as a Privacy-preserving Attribute-Based Credential System (PABCS) [19]. Multiple systems that adhere to the privacy properties of a PABCS have been described in literature [8, 13, 20, 2].

Credential attributes are trusted by a verifier if the credential is signed by an organization that is believed to issue trustworthy attributes. When the issuer finds out that the encoded attributes are no longer trustworthy or if it wants to prevent a user from using a credential, the issuer can *revoke* the credential. If a revocation mechanism is used, the verifier can determine the revocation status of a credential to check if a credential is revoked. Complementary to a revocation mechanism, expiration dates allow the issuer to indicate that a credential should not be used by a verifier after a specific time. When both are combined, a revocation mechanism is used to revoke a credential before its expiration date.

In this work, we aim to extend PABCSs to provide *forward-looking consistency* [47]. This strong consistency level means that the revocation status of a credential is checked even after a user made its request at a verifier. As an example, imagine a student requesting a monthly discount on a music subscription. The discount is automatically granted if the student is currently enrolled at a univerisity. As the university issues 'proof of enrolment' credentials, these can prove to the music company that the student is enrolled at a university. However, as the music company applies a monthly discount, it wants to be sure the student does not unenroll halfway during the academic year. If the student unenrolls, the university would revoke the 'proof of enrolment' credential and invalidates the requirement for the discount. This requires the verifier to determine the revocation status of the credential even after the student requested the discount, meaning *forward-looking consistency* is required in this example. A visualization of *forward-looking consistency* is given in Figure 1.1. In the illustration, C_1, C_2, C_3 are three credentials with start time t_{start} and end time t_{end}. When the student requests a discount, indicated by the first yellow line, it presents a 'proof of enrolment' credential. It is required that the request is made between the start and end times of all presented credentials. Green dots on the timeline represent revocation status checks, with $t_{r,max}$ indicating the last occurence. Every month when the music company needs to send an invoice, it needs to decide if the requirement for the discount is still met. These decision moments are indicated by the next two yellow lines. Based on the latest revocation status check, which occurs after the request, the music company decides if the discount should still be applied.

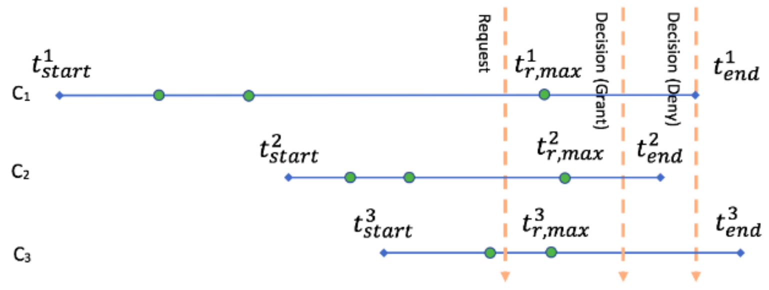

Figure 1.1: Forward-looking consistency illustrated by Shakarami and Sandhu [47]

1.2. Motivation

Existing PABCSs lack forward-looking consistency, which limits their usage in situations where this is required. This is an unfortunate situation, because PABCSs provide substantial security and privacy benefits over uniquely identifying authentication methods and self-attested attributes.

The privacy properties of a PABCS primarily benefit the users. First, it is always transparant to users what information is shared during authentication. Second, the information that is shared is reduced to a minimum by selectively disclosing only the attributes required for the authentication. Third, no other identifying information is shared other than the revealed attributes. This data minimization also benefits the verifier, as it no longer needs to process and store highly identifying and personal data about users. Furthermore, as the attributes are signed by a trusted credential issuer, users can no longer provide incorrect information about themselves.

An example situation that requires forward-looking consistency, is the Postfilter service [43]. In the Netherlands, the Postfilter service allows citizens to add their postal address to a filterlist and opt out of addressed commercial mail. Currently, this registration needs to be renewed every 5 years. One problem of the current process is that there is no reliable way to verify if the citizen actually lives at the submitted postal address, or if this changes any time during the 5-year registration period. By using a PABCS, it becomes possible to verify that the postal address belongs to the person requesting the registration. This can be done by showing a credential from the municipality that one lives at the address. While this solves part of the problem, Postfilter also needs to know when the municipality revokes the presented credential. But before Attribute-Based Credential (ABC)s can be used in this situation, the credential system requires forward-looking consistency.

1.3. Research question

How to design a privacy-preserving revocation mechanism for determining the revocation status after presentation in Privacy-preserving Attribute-Based Credential systems?

Existing PABCSs lack forward-looking consistency, which prevents them from being used in situations where this is strong notion of consistency is required. The first research question we want to answer, asks what revocation mechanisms currently exist and may be used to provide forward-looking consistency.

RQ1 How can a revocation mechanism be used to determine the revocation status of an earlier presented credential?

Revocation mechanisms by their nature rely on the usage of a unique identifier. As several credential systems allow users to present their credential multiple times in an unlinkable way, also referred to as *multi-show credentials*, this can conflict with a revocation mechanism. To avoid compromising on the functionality or privacy, we need to research how a revocation mechanism remains compatible with these types of credential systems. This is captured in the second research question:

RQ2 How do revocation mechanisms remain privacy-preserving in the context of multi-show credentials?

Another privacy property of a PABCS is that the credential issuer is unable to determine the issuance moment of a credential from a credential presentation. This is called untraceability and should hold even if the issuer colludes with a verifier. Like credentials, a revocation mechanism should also be untraceable. This is formulated in the third research question:

RQ3 How does a revocation mechanism remain untraceable when the issuer and verifier collude?

Revocation mechanisms are often regarded as an extension to a basic credential system. Although this allows them to be studied independently, this does not guarantee that they can be (easily) integrated into any type of credential system. Therefore, we take this compatability into consideration and formulate the following research question:

RQ4 How can a revocation mechanism that provides forward-looking consistency be used with an existing credential system?

Finally, we want to see if a revocation mechanism with forward-looking consistency can be used in practice. To see if this is possible, we want to measure the additional runtime cost compared to a regular credential system. This is formulated in the fifth research question:

RQ5 How does the runtime of a PABCS with forward-looking consistency compare against a regular credential system?

1.4. Contributions

In this work we research, for the first time, the consistency level of revocation mechanisms in a Privacy-preserving Attribute-Based Credential System (PABCS). We find that existing mechanisms do not address forward-looking consistency, which prevents the usage of ABCs in situations where this is required.

To address the need for forward-looking consistency in these situations, we propose the PABC-FLC credential system. This credential system provides forward-looking consistency and allows credential verifiers to determine the revocation status of a credential as newer versions of the Revocation Information (RI) are published. The PABC-FLC system provides *multi-show unlinkable* and *untraceable* credentials. Users can remain offline after credential issuance.

Finally, we measured the runtime performance of a proof-of-concept implementation of the PABC-FLC system. These measurements show that credential presentations and verifications thereof take 20 ms (+34%) longer compared to a credential system without revocation mechanism. Computing credential status updates take 85 ms and verifying them 55 ms. These results show that our design is a reasonable alternative for a credential system that does not provide forward-looking consistency.

1.5. Outline

The structure of the rest of this work is as follows. In Chapter 2, we give the preliminaries of ABCs and other cryptographic building blocks of our design. Chapter 3 describes related revocation mechanisms used both inside the field of ABCs, and outside. Chapter 4 contains the description of the proposed PABC-FLC system, a credential system with forward-looking consistency. In Chapter 5, a proof-of-concept of the PABC-FLC system is compared to a regular credential system on the runtime performance. Chapter 6 contains a discussion on the benefits and limitations of this work, along with directions for future research.

2 Preliminaries

2.1. Introduction

This chapter describes the building blocks of the proposed PABC-FLC system (see Chapter 4). These consist of Signature Proof-of-Knowledge (SPK) protocols (Section 2.2), a credential system (*idemix*, Section 2.7), an integer commitment scheme (*Damgård-Fujisaki* commitments, Section 2.3) and a revocation mechanism (*Braavos*, Section 2.5). We give a more elaborate description of PABCSs in Section 2.6 and since the *idemix* credential system uses Camenisch-Lysyanskaya signatures, this scheme is described in Section 2.4.

We use the following notations. When a value r is taken uniformly from a set S, we write $r \leftarrow_R S$. A generator g, which generates all elements in the set $\mathbb{G}$, is written as $\langle g \rangle = \mathbb{G}$. The order of a group G is written as $\#G$, and the same notation is used for the order of an element. The set of all integers in the range $[0, 2^\ell - 1]$ is written as $\{0,1\}^\ell$. The set of quadratic residues mod n, is written as $QR_n = \{r \in \mathbb{Z}_n^* | \exists a \in \mathbb{Z}_n^* \text{ s.t. } r = a^2 \bmod n\}$. A protocol between interactive Turing machines (P, V) is written as $\langle P(w), V \rangle(x)$ with w a private input to P and x a common input to both. Any (possibly malicious) machine is referenced as P^* and has the same input and output as P defined for the protocol. When checking equality to determine if the result should be $true$ or $false$, we write $1 \stackrel{?}{=} 1$. If a protocol or algorithm uses **Verify** and the result of the expression is $false$, it returns $\perp$ (abort). The function $\mathcal{H}_G \colon \{0,1\}^* \rightarrow G$ is a cryptographic hash function with a codomain G.

2.2. Proofs of knowledge

Before introducing proofs of knowledge, we introduce (zero-knowledge) interactive proofs [33]. Put simply, an interactive proof allows a prover to convince a verifier that a statement is true. Let L be a language and (P, V) be two interactive Turing machines. Both machines are given string x as input and at the end of their interaction, the powerful prover P convinces the Probabilistic Polynomial-Time (PPT) verifier V that $x \in L$. We require both *completeness* and *soundness*. *Completeness* states that V accepts, with overwhelming probability, if $x \in L$. *Soundness* states that V accepts only with negligible probability, for any prover P^*, if $x \notin L$. If V accepts, it 'learns' that $x \in L$. When V *only* learns that $x \in L$, this is referred to as *zero-knowledge*. In other words, zero-knowledge means that everything V learns, can also be learned from assuming that $x \in L$, without ever interacting with P. Like soundness should hold for all possible provers P^*, zero-knowledge should hold for all possible verifiers V^*.

It turns out that “all languages in NP have Zero-Knowledge proof systems” [32]. However, their usefullness is not always given. For example, all languages in the bounded-error probabilistic polynomial time (BPP) class are easily recognized by the verifier itself, without the need to communicate with a prover. Zero-knowledge proofs do turn out to be useful, specifically in situations where either the prover is more powerful than the verifier (as in the previous description) or when it is supplied with an input the verifier does not have access to.

In the rest of this work we focus on the latter case, where a Probabilistic Polynomial-Time (PPT) prover is given a *witness* w for a relation $(x, w) \in L_R$. (Where the language L_R describes a set of NP-relations R.) Furthermore, we want the prover to know a witness w, instead of only convincing the verifier that x is in the language. ‘Knowing’ is defined by the existence of an (expected PPT) *knowledge extractor* that outputs a witness w, such that $(x, w) \in L_R$ for any x with a witness in L_R. The knowledge extractor is given blackbox access to a prover that convinces the verifier with a probability greater than the knowledge error κ. The knowledge extractor must work for any, possibly malicious, prover P^*. A protocol that satisfies completeness and knowledge soundness, i.e. the existence of a knowledge extractor, is also called a “proof of knowledge” with knowledge error κ. If the soundness of the protocol relies on a computational assumption, e.g. hardness of the RSA assumption (see Section A.1.1), the protocol is also referred to as a *computationally convincing* proof of knowledge or *argument* of knowledge.

Σ-protocols [25] are protocol constructions that yield efficient arguments of knowledge. A Σ-protocol consists of three messages between a prover and verifier, where the verifier sends a public challenge (public-coin) as the second message. A definition, adapted to also suit statistical and computational zero-knowledge, is given in Definition 1.

Definition 1. Σ-protocol [26] A three-move protocol $\langle P, V\rangle$ is said to be a Σ-protocol for relation R if P sends a message a, V sends a random challenge e and P responds with a message z and the protocol further satisfies

- (*Completeness*) If P, V follow the protocol on input x and P has a private input w where $(x, w) \in R$, the verifier always accepts
- (*Special Soundness*) From any x and any pair of accepting conversations on input x, $(a, e, z), (a, e', z')$, where $e \neq e'$, one can efficiently compute w such that $(x, w) \in R$
- (*Special Honest-Verifier Zero-Knowledge*) There exists a polynomial-time simulator M, which on input x and a random e, outputs an accepting conversation of the form (a, e, z), with a probability distribution indistinguishable from conversations between the honest P, V on input x.

Σ-protocols can be seen as a generalization of the Schnorr identification protocol [46]. The Schnorr identitication protocol proves knowledge of a discrete log of y, i.e. knowledge of w in $y = g^w$, in a group of prime order q. It uses a subgroup of $\mathbb{Z}_p$ with p prime and q dividing

$p-1$ to ensure the discrete log problem is assumed to be hard. Both prover and verifier are given generator $\langle g \rangle = \mathbb{G}_q$, q, p, and common input y. Only the prover is given a witness w. The prover starts by taking a random element $r \leftarrow_R \mathbb{Z}_q^*$ and sending the *commitment* $t \leftarrow g^r \bmod p$ to the verifier. The verifier then sends *challenge* $c \leftarrow_R \{0,1\}^{\ell_c}$ to the prover. The prover computes *response* $s \leftarrow r + cw \bmod p$ and sends it to the verifier. The verifier then checks if $g^s \equiv tv^c \pmod p$ holds. If the relation holds it accepts, otherwise it aborts. This example is given in Protocol 1.

Protocol 1 Schnorr identification protocol, $PK\{(w)\colon y \equiv g^w \pmod p\}$

Prover	**Public**	**Verifier**
w	g, q, p, y	
$r \leftarrow_R \mathbb{Z}_q^*$		
$t \leftarrow g^r \bmod p$		
	$\xrightarrow{t}$	
		$c \leftarrow_R \{0,1\}^{\ell_c}$
	$\xleftarrow{c}$	
$s \leftarrow r + cw \bmod q$		
	$\xrightarrow{s}$	
		Verify $g^s \overset{?}{\equiv} ty^c \pmod p$

We make use of the fact that Σ-protocols can be combined, with the $\bigwedge$ (AND) operator, by using the same challenge for the two protocols. Additionally, it is possible to prove equality of two witness values in the same group. By using the same randomizer r for both relations, a single response s is received by the verifier and proves that the witness values are equal.

To simplify the description of a proof of knowledge, we use the notation introduced by Camenisch and Stadler [17]. As an example, consider the prime-order group $\mathbb{G}_q$, with generators $\langle g_1 \rangle = \langle g_2 \rangle = \mathbb{G}_q$, in which the discrete log problem is assumed to be hard. The following notation denotes a proof-of-knowledge protocol of the discrete logarithm of a_1 and a_2 regarding two different generators: $PK\{(x)\colon a_1 \equiv g_1^x \bigwedge a_2 \equiv g_2^x\}$. In this example, x is the secret discrete log that the prover claims to know. In contrast to the original notation, we do not write the secret variables in Greek letters but only specify these between the parenthesis.

As several building blocks of our credential system use special RSA modulus groups, a proof of knowledge must also work in these hidden-order groups. In contrast to prime-order groups, the knowledge extractor is unable to compute a witness since it does not know the order of the group. In fact, Bangerter shows that the minimum knowledge error of Σ_ψ-protocols for these hidden-order groups (using non-special homomorphisms) is $\frac{1}{2}$ [4]. This result shows that the only way to reduce the knowledge error is by running the protocol multiple times sequentially, which works for all proof-of-knowledge protocols [31]. This means the proof-of-knowledge protocols using a hidden-order group, like the special RSA modulus groups, require multiple rounds of communication between prover and verifier to achieve a negligible knowledge error.

Fortunately, it *is* possible to construct (single round) proofs of knowledge with negligible knowledge error, provided that the prover is given an *auxiliary input*. In this setting, it is assumed that the auxiliary input comes from a specific distribution and can be used by the knowledge extractor to compute a witness it otherwise would not be able to compute. The resulting protocol however, is no longer a proof-of-knowledge per the standard definition, due to the specific distribution requirement on the auxiliary input. This means that when analyzing the security of the protocol, the generation of this auxiliary input must be taken into consideration.

As an example, we describe the Schnorr protocol in a composite (special RSA) modulus group. Given a group $\mathbb{Z}_n^*$ with $n = pq$, the composite of two safe primes $p = 2p'+1, q = 2q'+1$. Let $\langle g \rangle = QR_n$ be a generator of the quadratic residue group. Soundness of the protocol requires that the prover is not aware of the group order [30]. The order is $p'q'$, and knowledge of it allows one to factor the modulus n, which is assumed to be hard under the Strong RSA assumption (see Section A.1.2). When the prover is not aware of the order of the group, it can still generate a randomizer in the group by taking $r \leftarrow_R \{0,1\}^{\ell_n+\ell_c+\ell_\varnothing}$. The security parameter $\ell_\varnothing$ determines the statistical zero-knowledge and ℓ_c determines the length of the challenge. Since the prover is also unable to reduce the response s, this can be used to prove that the witness w lies in $w \in \{0,1\}^{\ell_w}$. To prove this, a prover chooses $r \leftarrow_R \{0,1\}^{\ell_w+\ell_c+\ell_\varnothing}$ and the verifier checks that $s \in \{0,1\}^{\ell_n+\ell_c+\ell_\varnothing+1}$. The prover can only compute this signature with a high probability if w falls within the required range [15]. In contrast to the protocol in a prime-order group, this protocol is statistical honest-verifier zero-knowledge if $\ell_\varnothing$ is taken large enough (e.g. 80). The protocol for $PK\{(x)\colon v \equiv g^x \pmod{n}\}$ is given in Protocol 2. Proving equality of two values in two *different* (special RSA) groups is possible if the challenge is less than the order of both quadratic residue groups [15].

Protocol 2 Schnorr identification protocol in a composite group, $PK\{(w)\colon y \equiv g^w \pmod{n}\}$

Prover	**Public**	**Verifier**
w	$g, n, y, \ell_n, \ell_c, \ell_\varnothing$	
$r \leftarrow_R \{0,1\}^{\ell_n+\ell_c+\ell_\varnothing}$		
$t \leftarrow g^r \bmod n$		
	$\xrightarrow{\quad t \quad}$	
		$c \leftarrow_R \{0,1\}^{\ell_c}$
	$\xleftarrow{\quad c \quad}$	
$s \leftarrow r + cw$		
	$\xrightarrow{\quad s \quad}$	
		Verify $t \overset{?}{\equiv} y^{-c} g^s \pmod{n}$

These three-move public-coin interactive proof protocols can also be used as a signature scheme, by applying the Fiat-Shamir transformation [29]. Using Camenisch-Stadler notation, this Signature Proof-of-Knowledge (SPK) written as $SPK\{(x)\colon a_1 \equiv g_1^x \wedge a_2 \equiv g_2^x\}(m)$, with x

the prover input, m the message that is signed and all other values as public input [17]. When used as a signature scheme, the challenge is no longer provided by the verifier. Instead, the prover queries a random oracle for the challenge. In practice, we use a cryptographic hash function $\mathscr{H}_{\{0,1\}^{\ell_c}}$. The input to the oracle contains all public knowledge, a description of the relation(s) being proven, the t-commitments and a message m. Including all public parameters in the input to the hash function is required to prevent attacks on the soundness of the protocol [7]. As the prover takes the role of the verifier, it preserves the zero-knowledge property of the protocol if this was proven in the (special) honest-verifier setting. The resulting signature scheme is secure against adaptive chosen message attacks [42]. An algorithm computing $SPK\{(w)\colon y \equiv g^w \pmod{n}\}$ is given in Algorithm 3, with a verification algorithm in Algorithm 4.

Algorithm 3 Sign $SPK\{(w) : y \equiv g^w \pmod{n}\}(m)$

```
1: function SPK($w, g, n, y, \ell_n, \ell_c, \ell_\varnothing, m$)
2:     $r \leftarrow_R \{0,1\}^{\ell_n+\ell_c+\ell_\varnothing}$
3:     $t \leftarrow g^r \bmod n$
4:     $c \leftarrow \mathscr{H}_{\{0,1\}^{\ell_c}}(\mathrm{desc}(y \equiv g^w \pmod{n}), n, g, y, t, m)$
5:     $s \leftarrow r + cw$
6:     return $(c, s)$
7: end function
```

Algorithm 4 Verify $SPK\{(w)\colon y \equiv g^w \pmod{n}\}(m)$

```
1: function SPK($g, n, y, \ell_n, \ell_c, \ell_\varnothing, m, (c, s)$)
2:     $t' \leftarrow y^{-c} g^s \bmod n$
3:     $c' \leftarrow \mathscr{H}_{\{0,1\}^{\ell_c}}(\mathrm{desc}(y \equiv g^w \pmod{n}), n, g, y, t', m)$
4:     return Verify $c \overset{?}{=} c'$
5: end function
```

2.3. Damgård-Fujisaki integer commitment scheme

A commitment scheme is used by a *committer* to commit to a value, creating a commitment. At the same time, the commitment hides the committed value from the *receiver* of the commitment. If the committer wants to open the commitment, it will only be able to do so to the originally committed value. Commitment schemes therefore have two properties: *hiding* and *binding*.

The Damgård-Fujisaki integer commitment scheme [27] is a revised version of the earlier Fujisaki-Okamoto commitment scheme [30]. The scheme is statistically hiding and computationally binding under the integer factorization assumption. We describe the scheme and show how one can prove knowledge of a commitment opening. The statistical zero-knowledge argument of knowledge over a commitment opening is sound under the Strong RSA assumption (see Section A.1.2) [27]. Later work showed that soundness also holds under the (normal) RSA assumption (see Section A.1.1) [24].

Remark. Although the authors of [24] mention that their result extends to all "discrete-logarithm relation sets", the rest of this work will refer to the original analysis that assumes the Strong RSA assumption. We do so because the PABC-FLC design, presented in Chapter 4.3, requires this stronger assumption anyway. Showing that the soundness of the different SPK protocols we use hold under the (weaker) RSA assumption would therefore not improve the security of the overall design.

Setup Given a security parameter 1^κ, the receiver generates the public parameters that the committer will use. Let $n = pq$ be a special RSA modulus, with p, q safe primes that have a bit length of κ. Let $\langle h \rangle = QR_n$ be a generator of quadratic residue group and generate element $g \leftarrow_R \langle h \rangle$ such that the discrete log of g regarding h is unknown to the committer. An upper-bound on the order of QR_n, written as 2^B, is assumed to be efficiently computable (i.e. $\frac{n}{4}$). Output (n, g, h).

Commit The committer chooses a random integer $r \leftarrow_R [0, 2^{B+\kappa}]$. To commit to integer x, compute commitment $c \leftarrow g^x h^r \mod n$. The committer stores x, r and sends commitment c to the receiver.

Open To open a commitment c, the committer sends the stored x, r to the receiver. The receiver verifies that $c \overset{?}{\equiv} \pm g^x h^r \pmod{n}$.

Argument of the commitment opening If a committer wants to prove that it knows the committed integer and associated opening of a commitment, it can use a proof-of-knowledge protocol. This computationally convincing protocol is described as $PK\{(x, r) \colon c \equiv \pm g^x h^r \pmod{n}\}$ and is sound under the Strong RSA assumption [27].

2.4. Camenisch-Lysyanskaya signature scheme

The Camenisch-Lysyanskaya signature scheme [12] is used to create signatures that have efficient proof of knowledge protocols. These protocols allow a prover to prove knowledge of a signature and message without revealing either of them. The signature scheme is unforgeable under the Strong RSA assumption (see Section A.1.2) and described as follows.

KeyGen Given a security parameter 1^κ, generate an RSA modulus $n = pq$ with safe primes $p = 2p' + 1, q = 2q' + 1$. The modulus n should have a length 2 times the security parameter κ. Then randomly select quadratic residues $a, b, c \leftarrow_R QR_n$. Return public key $pk \leftarrow (n, a, b, c)$ and secret key $sk \leftarrow (p, q)$.

Sign For a message $m \in \{0,1\}^{\ell_m}$, generate random prime $e \leftarrow_R [2^{\ell_e - 1}, 2^{\ell_e} - 1]$ with $\ell_e \geq \ell_m + 2$. Generate random integer $v \leftarrow_R [2^{\ell_v - 1}, 2^{\ell_v} - 1]$ with $\ell_v = \ell_m + \ell_n + \ell_\varnothing$ and $\ell_\varnothing$ a statistical security parameter. Compute A such that $A^e \equiv a^m b^v c \pmod{n}$. Return signature (A, e, v).

Verify Given a signature (A, e, v) over message $m \in \{0,1\}^{\ell_m}$, verify that $A^e \overset{?}{\equiv} a^m b^v c \pmod{n}$. Also verify that $2^{\ell_e - 1} < e < 2^{\ell_e}$. If none of the verifications abort, return $true$.

2.5. Cryptographic accumulators

Cryptographic accumulators represent a set of elements in a short, constant size value. Each element comes with a witness that proves its (non-)membership in the set. Over time different accumulators have been proposed, varying in properties and hardness assumptions [14, 41, 11, 40]. Dynamic accumulators allow elements to be added or removed from the set of accumulated elements. After each addition or removal a new accumulator value is published. Each new accumulator value requires the witness of an element to be updated. One might wonder if it is possible to batch these witness updates, but unfortunately this is not possible and the number of updates is (at least) linear to the number of additions or removals [9]. One accumulator scheme that achieves this lower bound is the Braavos accumulator design. This scheme requires witness updates only on removals, i.e. revocations [3]. The Braavos design provides an efficient Signature Proof-of-Knowledge (SPK) of set membership, allowing a prover to keep both the accumulated element and associated witness secret.

We will give a short description of the Braavos design and associated CL-RSA-B accumulator [3]. The Braavos design shows how two different accumulators, one additive accumulator and one dynamic accumulator that lacks soundness, can be combined to provide the best properties of both. Soundness means that an adversary is unable to prove membership of an element in the accumulator, if that element is not contained in the accumulator. The adversary can add and remove any element it wants to. Non-adaptive soundness means an adversary can only add or remove elements from a specific set of elements. The adversary commits to this set before the setup. By combining an adaptively sound additive accumulator, called ACC_A, and non-adaptively sound dynamic accumulator, called ACC_{NA}, the resulting accumulator is both dynamic and adaptively sound. The ACC_A accumulator can be constructed using a signature scheme, like the Camenisch-Lysyanskaya signature scheme (see Section 2.4). The ACC_{NA} accumulator, called CL-RSA-B, is based on the earlier Camenisch-Lysyanskaya accumulator design [14]. The security of both accumulators relies on the Strong RSA assumption (see Section A.1.2). The Braavos design improves on earlier accumulators by reducing the number of witness updates. Witness updates are no longer required on additions (i.e. credential issuance) but only on removals (i.e. credential revocation). The CL-RSA-B accumulator is described in terms of the following algorithms: setup, adding, removing, proving membership and witness updating. The domain of accumulated values is the set of all odd primes in $[A, B]$ with $2 < A, B < A^2$.

Setup Given the security parameter 1^{ℓ_n}, choose safe primes $p \leftarrow_R 2p' + 1, q \leftarrow_R 2q' + 1$ of bit length $\ell_n/2$ where p', q' are also prime numbers. Compute $n \leftarrow pq$. The (initial) accumulator value is taken randomly from the set of quadratic residues $v \leftarrow QR_n$. In addition to the accumulator, two bases for the commitment scheme are generated $g, h \leftarrow_R QR_n$. Return the public key $pk \leftarrow (n, g, h)$, secret key $sk \leftarrow (p', q')$ and accumulator value v.

Adding Select a random prime $e \leftarrow_R [A, B]$ that will be added to the accumulator. Compute the witness $w \leftarrow v^{e^{-1} \pmod{p'q'}} \bmod n$. Note that the accumulator value does not change and the witness for other elements does need to be updated. Return the accumulated value e and associated witness w.

Proving Let e be the accumulated value, w the witness and v the accumulator value. A proof of membership shows that $v \equiv w^e \pmod{n}$. The proof of membership protocol is given in Protocol 5, but for more details on the SPK we refer to Appendix A of the original Camenisch-Lysyanskaya accumulator [14]. At the start of the protocol the verifier sends nonce n_1 to the prover. The prover computes integer commitments C_e, C_w, C_r to e, w and the commitment opening of w, r_2. Then it uses the specified SPK to sign n_1 and sends the signature to the verifier. The verifier checks the SPK and accepts if the signature is valid. As mentioned before, the CL-RSA-B accumulator should be combined with another accumulator. This means that the prover also needs to prove that the committed value e is part of another accumulator, for example by showing that it is an attribute value of a signed credential (see Section 2.7).

Removing First, verify that e, the element that will be removed, comes from the accumulation domain (i.e. is an odd prime in $[A, B]$). The new accumulator value can be computed as $v' \leftarrow v^{e^{-1} \pmod{p'q'}} \bmod n$. Return update message (v', e).

Updating witness Given an update message (v, y), the witness w for an element e can be updated if the element is not the one that is revoked $e \neq y$. First, compute the Bezout coefficients b, c in $bw + cy = 1$. Since both w, y are primes, these values exist. The updated witness is computed as $w' \leftarrow w^c v^b$.

2.6. Privacy-preserving Attribute-Based Credential Systems

A Privacy-preserving Attribute-Based Credential System (PABCS) consists of three roles: a credential issuer, a credential user and a credential verifier. The issuer can issue new credentials to users, usually based on another credential or a pre-existing relation. These credentials contain a bundle of attributes that are associated with the identity of the user. An example credential may be a passport containing a name, date of birth and nationality. Users can obtain credentials from different issuers and present them to credential verifiers. During this credential presentation, the user reveals a subset of the attributes in the credential to the verifier. If the presented credential was issued by an issuer the verifier trusts, the verifier can use the revealed attributes in the decision it has to make. For example to grant or deny access based on the date of birth of a user. Verifiers are free to choose the issuers they trust and can even decide to trust an issuer only for a specific set of attributes. As an example, a verifier may trust a university to state that someone is a student, but not that someone has a valid driving licence.

When we consider the revocation of credentials, we introduce a new role: the Revocation Authority (RA). The Revocation Authority (RA) has the ability to globally revoke a previously issued credential. The RA cooperates with the issuer during credential issuance and together

Protocol 5 Proof of membership $v \equiv w^e \pmod{n}$ [14]

Prover	**Public**	**Verifier**
	$g, h, n, B, \ell_\emptyset, \ell_c$	
w, e	v	
		$n_1 \leftarrow_R \{0,1\}^{\ell_\emptyset}$
	$\xleftarrow{n_1}$	
$r_1, r_2, r_3 \leftarrow_R \mathbb{Z}_{n/4}$ $C_e \leftarrow g^e h^{r_1} \bmod n$ $C_w \leftarrow w h^{r_2} \bmod n$ $C_r \leftarrow g^{r2} h^{r3} \bmod n$		
$\sigma \leftarrow SPK\{(e, u, r_1, r_2, r_3):$ $C_e \equiv g^e h^{r_1} \pmod{n}$ $\bigwedge C_r \equiv g^{r_2} h^{r_3} \pmod{n}$ $\bigwedge v \equiv C_w^e h^{-er_2} \pmod{n}$ $\bigwedge 1 \equiv C_r^e h^{-er_3} g^{-er_2} \pmod{n}$ $\bigwedge e \in [-B2^{\ell_\emptyset+\ell_c+1}, B2^{\ell_\emptyset+\ell_c+1}]\}(n_1)$		
	$\xrightarrow{\sigma}$	
		Verify σ

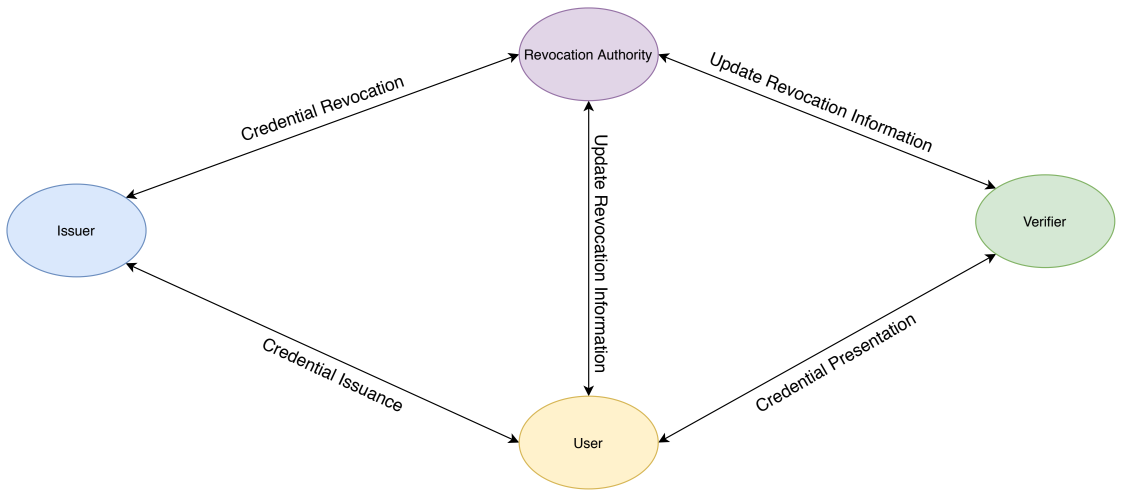

Figure 2.1: Relations between the issuer, user, verifier and revocation authority

they encode a special attribute, a Revocation Handle (RH), in the new credential. If a revocation mechanism publishes information about the set of (un)revoked credentials, we refer to this as the Revocation Information (RI). When the issuer wants to revoke a credential, it notifies the RA. The RA then updates the Revocation Information (RI) by adding or removing the revocation handle from the RI. Based on the most recent version of the RI, users and/or verifiers can determine if a credential is revoked. The issuer, user, verifier and revocation authority roles are illustrated in Figure 2.1. Although we describe the issuer and RA as two seperate roles, this work only considers the situation in which the issuer revokes a credential, and does so for all verifiers. This means the issuer and RA are the same participant, which is assumed to be the case in the rest of this work, if not otherwise specified.

As a clarification for readers more familiar with Self-Sovereign Identity (SSI) systems, what we describe as the credential user is similar to the concept of a credential holder. Although the holder has a credential, this does not mean it is also the subject of that credential. The credential subject can be another entity, while the holder only has access to the credential. For example a pet owner can hold a credential about their pet. This allows the holder to reveal attributes not just about itself, but also about their pet. In this work, we stick to the credential user terminology and only consider credentials that describe the user. Do note that a single entity can change its role at any moment. For example, a driver acts as a credential user when presenting a driving licence during a traffic stop, but at the same time acts as a credential verifier to verify that the other person is a police officer.

2.7. Idemix

One construction of a PABCS is the Identity Mixer design, also known as *idemix* [13, 18]. *Idemix* is built on the Camenisch-Lysyanskaya signature scheme (see Section 2.4). Using the Signature Proof-of-Knowledge (SPK) algorithms defined for this signature scheme, users can prove knowledge of a credential issued to them. This provides both *multi-show unlinkabil-*

ity and *issuance-presentation untraceability*. In addition, it allows blind issuance of (user) attributes and selective attribute disclosure during credential presentation.

Blind issuance of attributes, combined with the ability to keep attribute values hidden during presentation, allows a user to encode a *user secret* in the credential. This user secret is never revealing to any other participant, such as the issuer or verifier. By proving knowledge of the *user secret* attribute during a credential presentation, instead of revealing the attribute, this proves that the credential was issued to the same user that presents it. That is, if we assume users do not share their user secret with others. This is also referred to as the *non-transferability* of credentials. Furthermore, if a user proves that the user secret in one credential equals that of another credential, this proves that both credentials were issued to the same user. This property is known as *consistency of credentials*.

We will give a brief description of the *idemix* credential system, although we leave out several aspects not related to this work such as pseudonyms, range proofs or more efficient methods to encode credentials. For a description of those, we refer to the *idemix* specification [49].

System setup The system setup algorithm outputs the security parameters, which are used by all other algorithms and protocols. Attribute values are in the set $\{0,1\}^{\ell_m}$, as in the signature scheme (see Section 2.4). The $\ell_n, \ell_e, \ell_v, \ell_\emptyset$ parameters are used as in the signature scheme, with the addition of the ℓ'_e parameter that determines the upper limit of the interval of e values. The parameter ℓ_c denotes the maximum length of challenges in the SPKs. The requirements on the lengths of these parameters are slightly different from the signature scheme, for these we refer to table 2 of the *idemix* specification [49].

Issuer setup The issuer first generates a special RSA modulus by computing $N = pq$ with $p = 2p'+1, q = 2q'+1$ and p, q, p', q' all prime. As N should have a length of ℓ_n, p and q should both have a length of $\ell_n/2$. The issuer continues by generating quadratic residue $S \leftarrow_R QR_N$ and $Z, R_1, \ldots, R_i \leftarrow_R \langle S \rangle$, with i the number of credential attributes. The set of attributes is described as $Att = \{1, \ldots, i\}$. The secret key of the issuer is (p', q'), while the public key is returned as $(N, S, Z, R_1, \ldots, R_i, Att)$.

Credential issuance Using the credential issuance protocol, users can obtain a new credential from the issuer. Known attributes $Att_k \subset Att$ are known to both user and issuer, while the rest of the attributes $Att_k^{\complement} = Att \setminus Att_k$ are only known to the user. The *user secret* attribute (m_1) is always part of the set of user attributes, $1 \notin Att_k$. We assume the user and issuer agree on the known attribute values $\{m_k\}_{k \in Att_k}$ beforehand. The issuance protocol is specified in Protocol 6.

Credential presentation Given is a user with credential (A, e, v), which it will present to a verifier. Beforehand the verifier and user agree on the attributes that will be revealed. Revealed attribute values are in the set Att_r and the values of these attributes, $\{m_i\}_{i \in Att_r}$, are sent to the verifier. Other attribute values, $\{m_i\}_{i \notin Att_r}$, are kept hidden. The protocol starts

Protocol 6 Credential issuance protocol of idemix

User	**Public**	**Issuer**
	$N, S, Z, \{R_i\}_{i\in Att}, Att, Att_k \subset Att$	
$\{m_h\}_{h\notin Att_k}$	$\{m_k\}_{k\in Att_k}$	p', q'
		$n_1 \leftarrow_R \{0,1\}^{\ell_\varnothing}$
	$\xleftarrow{n_1}$	
$v' \leftarrow_R \{0,1\}^{\ell_n+\ell_\varnothing}$		
$U \leftarrow S^{v'} \prod_{h\notin Att_k} R_j^{m_j} \bmod N$		
$\sigma_P \leftarrow SPK\{(\{m_h\}_{h\in Att_h}, v'):$		
$\quad U \equiv S^{v'} \prod_{h\notin Att_k} R_h^{m_h} \pmod N$		
$\quad \bigwedge\{m_h \in \{0,1\}^{\ell_m+\ell_\varnothing+\ell_c+1}\}_{h\notin Att_k}$		
$\}(n_1)$		
$n_2 \leftarrow_R \{0,1\}^{\ell_\varnothing}$		
	$\xrightarrow{n_2,\sigma_P,U}$	
		Verify σ_P
		prime $e \leftarrow_R [2^{\ell_e-1}, 2^{\ell_e-1}+2^{\ell'_e-1}]$
		$v'' \leftarrow_R \{0,1\}^{\ell_v-1}$
		$B \leftarrow \frac{Z}{US^{v''}\prod_{k\in Att_k} R_i^{m_k}} \bmod N$
		$A \leftarrow B^{e^{-1} \bmod p'q'} \bmod N$
		$\sigma_I \leftarrow SPK\{(e^{-1}): A \equiv B^{e^{-1}} \pmod N\}(n_2)$
	$\xleftarrow{\sigma_I,(A,e,v'')}$	
$v \leftarrow v' + v''$		
Verify σ_I		
Verify $Z \overset{?}{\equiv} A^e S^v \prod_{i\in Att} R_i^{m_i} \pmod N$		

with the verifier sending a nonce to the user, to prevent replay attacks. The user can randomize the credential and prove knowledge of the signature over both the revealed and hidden attribute values. A description of the protocol is given in Protocol 7.

Protocol 7 Credential presentation protocol of idemix

User	**Public**	**Verifier**
(A, e, v)	$N, S, Z, \{R_i\}_{i \in Att}, Att$	
$\{m_i\}_{i \notin Att_r}$	$\{m_i\}_{i \in Att_r}, Att_r \subset Att$	
		$n_1 \leftarrow_R \{0,1\}^{\ell_\emptyset}$
	$\xleftarrow{n_1}$	
$r \leftarrow_R \{0,1\}^{\ell_n + \ell_\emptyset}$ $A' \leftarrow AS^{-r} \bmod N$ $v' \leftarrow v - er$		
$\sigma \leftarrow SPK\{(e, \{m_i\}_{i \notin Att_r}):$ $\frac{Z}{\prod_{i \in Att_r} R_i^{m_i}} \equiv A'^e S'^v \prod_{i \notin Att_r} R_i^{m_i} \pmod N$ $\bigwedge \{m_i \leftarrow \{0,1\}^{\ell_m + \ell_\emptyset + \ell_c + 1}\}_{i \notin Att_r}$ $\bigwedge e - 2^{\ell_e - 1} \in \{0,1\}^{\ell'_e + \ell_\emptyset + \ell_c + 1}$ $\}(n_1)$		
	$\xrightarrow{\sigma}$	
		Verify σ

3 Related work

3.1. Introduction

This chapter describes existing revocation mechanisms relevant to providing forward-looking consistency in a PABCS. First, two revocation mechanisms from outside the field of PABCSs are described, specifically Certificate Revocation Lists (CRLs) and Online Certificate Status Protocol (OCSP). Both are used as a revocation mechanism in the Web PKI. The other three mechanisms were specifically designed for PABCSs. These include Verifiable Encryption, N-Times unlinkable proofs and cryptographic accumulators. The Braavos [3] cryptographic accumulator, described in Section 2.5, will not be repeated here, but is taken into account in the discussion section. Likewise, the *idemix* credential system and Signature Proof-of-Knowledge (SPK) protocols are described in the preliminaries. This chapter ends with a comparison of the described revocation mechanisms in terms of the properties they provide, including forward-looking consistency, unlinkability and untraceability.

3.2. Web PKI revocation mechanisms

The Web PKI is used to authenticate website certificates, often recognized as a green padlock in the web browser. These certificates authenticate the domain name and are issued by trusted Certificate Authorities (CAs). Note that in contrast to ABCs, these certificates do not provide any privacy properties like selective disclosure or multi-show unlinkability.

One of the first revocation mechanisms, standardization began in 1996, is the Certificate Revocation List (CRL) [36, 37]. This mechanism requires CAs to periodically sign and publish a list of all revoked certificates. The certificates themselves are referenced by a unique serial number. These serial numbers are added to the certificate, together with a location where the latest CRL can be found. Web browsers and other verifiers retrieve the CRL on a regular basis to ensure they have the latest version. The revocation status of a certificate can be determined by searching the list for its serial number. As the size of the list is linear to the number of revoked certificates, this requires web browsers to store a large amout of data. Another downside is that if a certificate references an unknown CRL, one that was not downloaded before, it stalls the revocation check of the certificate until the list is downloaded.

Another revocation mechanism used in the Web PKI is using Online Certificate Status Protocol (OCSP) requests [38]. This mechanism works by online querying the CA for the revocation status of a certificate. Compared to CRLs, web browsers and other verifiers are no longer required to store the revocation information locally. However, the downside is that the CA is must be available at all times. From a privacy perspective, this also allows the CA to keep track of which certificates were queried by which verifier. To alleviate these problems, it is possible for the presenter of the certificate, e.g. a website, to “staple” the Online Certificate Status Protocol (OCSP) response to its certificate presentation. As the verifier no longer makes the OCSP query, this removes the privacy problem. Furthermore, the responses can be cached for a small period of time to reduce both the number of requests made to the CA and impact of availability issues.

3.3. Verifiable Encryption

Verifiable Encryption (VE) also involves querying another party for information about a credential. VE was first introduced as an anonymity revocation mechanism for ABCs [16]. It lets a user encrypt a credential attribute, and prove that it did so correctly. For example, if a verifier wants the ability to take legal action against a user that violated some policy, it can ask the user to encrypt its name under the public key of a third party auditor. As only the third party auditor is able to decrypt the information, the verifier can only learn the users name in cooporation with the auditor. VE can also be used as a revocation mechanism, by encrypting a Revocation Handle (RH) under the public key of the RA. As with OCSP, this requires the RA to be available at all times and allows it to monitor credential usage. For these reasons, VE is useful as an anonymity revocation mechanism, but less so as a credential revocation mechanism [16].

3.4. N-times unlinkable proofs

Applications like smart cards and electronic identity cards require a revocation mechanism that works without any internet connectivity of a user. The “N-times unlinkable proofs” scheme is designed for these situations and only requires verifiers to keep track of the RI. Each credential can generate a limited number of N unlinkable proofs (pseudonyms) within a predefined time period (epoch) [10]. One of these pseudonyms is generated during the credential presentation and given to the credential verifier. The RA distributes a list of revoked pseudonyms to all verifiers, who can lookup the received pseudonym in the list. However, there is a trade-off, as N pseudonyms can be generated per credential, the size of the RI increases linearly with N. The major component of the scheme is the way users prove that the pseudonym is generated correctly, meaning generated from the RH in the credential.

We give a short description of the n-times unlinkable proofs scheme as described in [10]. Users use a Revocation Handle (RH) and signed “randomizers” to generate pseudonyms. Since both the RH and randomizers are generated by the RA, the RA can also generate all pseudonyms in case the RH needs to be revoked. The RI acts as a denylist and is updated each epoch.

Setup First select parameters (j, k) such that $n = k^j$ and with n the number of unlinkable pseudonyms that can be created. Then select a group $\langle g \rangle = \mathbb{G}_p$ of prime-order p and an order related to the security parameter 1^κ. Select random elements $a_1, \ldots, a_j \leftarrow_R \mathbb{Z}_p$ and compute $g_i \leftarrow g^{a_i}$ for all a_i. Let Σ be an existentially unforgeable signing scheme that allows for efficiently generating a proof of knowledge over the signature (see Section 2.4 for such a scheme). The algorithm $\Sigma.KeyGen$ generates a signing and verification key, while the algorithms $\Sigma.Sign$ and $\Sigma.Ver$ respectively place and verify a signature on a message. As the randomizers need to be signed, generate a key pair $(sk, pk) \leftarrow \Sigma.KeyGen(1^\kappa)$. Select the set of randomizers $\{e_i \leftarrow_R \mathbb{Z}_p\}_{i=1,\ldots,k}$ and sign them $\{\sigma_i \leftarrow \Sigma.Sign(sk, e_i)\}_{i=1,\ldots,k}$. Publish the system parameters $(g, p, k, j, (g_1, \ldots, g_j), (a_1, \ldots, a_j), \Sigma, \{(e_1, \sigma_1), (e_k, \sigma_k)\})$. Store the secret key sk and publish the public key pk.

Issuance The RA generates a RH $w \leftarrow_R \mathbb{Z}_p$ and stores it in the list of issued handles. The RH w is given to the issuer and put in a credential. Since w is put in a credential, the credential signature acts as signature over w. As a result, there is no need for the RA to also sign the RH as it did with the randomizers.

Proving Per epoch, the user can generate $n = k^j$ different pseudonyms. To make sure this limit is not exceeded, which would result in reuse and linkability of pseudonyms, it keeps an integer counter ctr to indicate the number of pseudonyms that have been generated already during the current epoch. Counter ctr is also used as a k-ary number $(ctr_0, \ldots, ctr_{j-1})$. The $epoch$ value should be known to both parties and have an associated RI published. The user and verifier can for example agree to use the epoch of the latest RI RL_{epoch} that the verifier has access to. Here we take it that w comes from a (signed) credential and is contained in the commitment c. The $Com.Ver$ algorithm proves that commitment c opens to w using opening o. It is assumed that both parties know the commitment key and that $Com.Ver$ has a relation that can be proven with a SPK (see Section 2.3 for such a commitment scheme). To prevent replay attacks, the user signs a nonce n_1. The length of the nonce $(\ell_\emptyset)$ should be related to the security parameter 1^κ. Both proving and verifying the generation of a pseudonym are specified in Protocol 8.

Verifying During verification, the verifier checks the proof π and uses pseudonym C to see if it exists in the RI of that epoch (RL_{epoch}). If the pseudonym does not appear in the RI, it means the credential is not revoked. The authors mention that by using a hash function with short output length, the size of the RI remains acceptable.

Revoking The RA keeps a separate list of all revoked RHs to which a RH is added when it gets revoked. When a RH gets revoked, the RA publishes all possible pseudonyms in the RI for the next and all following epochs. It does so by computing $\mathcal{H}(C_i)$ for all pseudonyms $i = 0, \ldots, k^j - 1$ during the next epoch. It then publishes the RI, a sorted list of all revoked pseudonyms, for the next epoch. This repeats for each epoch as long as the credential is revoked, for example until it expires.

Protocol 8 Proving and verifying a pseudonym in N-times unlinkable proofs

Prover	**Public**	**Verifier**
	$g, p, k, j, \Sigma, pk, \ell_\varnothing$	
	$(g_1, \ldots, g_j), (a_1, \ldots, a_j)$	
	$\{(e_1, \sigma_1), \ldots, (e_k, \sigma_k)\}$	
ctr, w, o, c	$epoch$	RL_{epoch}
		$n_1 \leftarrow_R \{0,1\}^{\ell_\varnothing}$
	$\xleftarrow{n_1}$	
$(ctr_0, \ldots, ctr_{j-1}) \leftarrow ctr$		
$i \leftarrow \sum_{\ell=0}^{j-1} a_i e_{ctr_\ell}$		
$C \leftarrow g^{1/(w+i+\mathscr{H}(epoch))} \bmod p$		
$\sigma \leftarrow SPK\{(w, i, e_{ctr_1}, \ldots, \{e_i\}, \{\sigma_i\}, o):$		
$\quad gC^{-\mathscr{H}(epoch)} \equiv C^w C^i \pmod p$		
$\quad \bigwedge g^{-i} \prod_{\ell=1}^{j} g_\ell^{e_{ctr_\ell}} \equiv 1 \pmod p$		
$\quad \bigwedge_{\ell=1}^{j} \Sigma.Ver(pk, \sigma_{ctr_\ell}, e_{ctr_\ell}) \equiv 1$		
$\quad \bigwedge Com.Ver(c, w, o) \equiv 1$		
$\}(n_1)$		
$ctr \leftarrow ctr + 1$		
	$\xrightarrow{\sigma, C, c}$	
		Verify $\mathscr{H}(C) \notin RL_{epoch}$
		Verify σ

This construction does not require the user to keep track of updates in the RI and still maintains backwards unlinkability, even when the credential is revoked. Computing the pseudonym and associated proofs are efficient enough to run on smart cards with limited computation power [10]. However, since anyone with knowledge of to the RH can generate all pseudonyms (for all epochs), this means the pseudonyms are traceable. In other words, since the credential issuer has access to the RH, it is able to link any pseudonym to a specific issuance moment. The generated pseudonyms themselves are unlinkable due to the Dodis-Yampolskiy Verifiable Random Function construction [28]. This ensures that without knowledge of the RH, credentials can be presented multiple times without linkability between those presentations.

3.5. Delegatable cryptographic accumulator

When researching delegatable credentials, Acar and Nguyen designed a delegatable revocation mechanism [1]. Delegatable credentials allow a user to provide their credentials to a delegatee who is then able to use the credentials on their behalf. Delegation should be unlinkable in that two different delegated credentials cannot be linked to the same ancestor credential. Regarding revocation, this means that the RH is not revealed during delegation, just as it is not during presentation. The cryptographic accumulator scheme is build on the homomorphic properties of Groth-Sahai proofs [34] and provides this unlinkability. The combination of accumulator and homomorphic proofs allows the delegatee to also update the non-membership proof when the RI changes, providing a new non-revocation proof that is required when presenting the credential.

We will give a description of the Accumulator with Delegatable Non-Membership Proofs (ADNMP) scheme, but refer to the original paper for the instantiation using Groth-Sahai proofs [1]. All algorithms mentioned run in PPT and implicitly take the system parameters $Para$ as input. The first algorithms are standard in non-membership accumulators, the $UpdateVal$ and $UpdateNMWit$ algorithms make it a dynamic accumulator and delegation uses the $Dele, Rede, Vali$ and $CompNMProof$ algorithms.

Setup is given a security parameter 1^κ and returns the system parameters and auxiliary information $(Para, Aux)$.

Accu is given the set elements to accumulate $AcSet$. It returns the accumulator value $AcVal$.

CompNMWit takes the set of accumulated values $AcSet$ and the accumulator value $AcVal$. It is also given an accumulated element Ele for which it computes the non-membership witness $NMWit$.

ProveNM can produce a proof of non-membership of element Ele and associated witness $NMWit$ for the accumulator value $AcVal$. As this is an accumulator, it does not need access to the entire set of accumulated elements.

VerifyNM takes the proof of non-membership and accumulator value $AcVal$ and returns a boolean value indicating if the proof is valid.

UpdateVal computes the new accumulator value $AcVal'$ when element Ele is added. It does not take $AcSet$ as input to compute the new accumulator value, but does require the old accumulator value $AcVal$.

UpdateNMWit allows the computation of a new $NMWit'$ when the element Ele is added to the set of accumulator values, given also the current $NMWit$ and new $AcVal$.

Dele is given an element Ele and returns the delegating key De.

Rede takes as input a key De and produces another delegating key De'.

Vali is given a delegating key De and returns a boolean value indicating if the delegation is valid.

CompNMProof takes a delegating key De, accumulator set $AcSet$ and accumulator value $AcVal$. It returns a non-membership proof that the element Ele, associated with delegation key De, is not accumulated in the accumulator value $AcVal$.

The authors describe how the ADNMP scheme can be used to provide a revocation mechanism for delegatable credentials. The scheme provides delegatability by allowing users to prove the ancestor delegators are not included in the accumulator. It also provides redelegability by delegating the non-membership proof to others. The non-membership proofs by users and delegators are indistinguishable from each other and unlinkability prevents users from linking delegations to the same delegator. There is a trade-off however, between delegability and anonymity. Any user who delegates a credential can collude with the RA to determine if a non-membership proof was generated by their element, i.e. break anonymity (see Section 4.2.7).

3.6. Discussion

Our goal is to find a revocation mechanism that provides forward-looking consistency in a PABCS. At the same time, we require that such a mechanism does not compromise the privacy properties of Attribute-Based Credentials. First, we compare the described mechanisms on the properties they already provide. We then discuss the missing properties to see if these can be added, for example by changing how the revocation mechanism works.

We compare the discussed revocation mechanisms on the different properties they provide. The key property we need is that a credential verifier can check the revocation status of a credential in newer versions of the RI, which is referred to as *forward-looking consistency* (FLC). As a PABCS provides both *multi-show unlinkable* and *untraceable*, we also compare the revocation mechanisms on these properties. Concerning the RI, we compare the revocation mechanisms on the size of the RI and frequency of new versions. Both are described in terms of the number of revoked handles / revocations (R), epochs (E), unlinkable proofs (U), constant (1) or not applicable (?). Note that for both accumulator, Braavos and ADNMP, the user needs to download each version of the RI. While the size is constant, the total size that users need to download is not. The last property we consider is the required connectivity of the participants. In the table we write I if the issuer needs to be online during verification. If the user needs to download the RI or update a witness, we write U. If the verifier needs to download the RI, we write V. We exclude the connectivity requirement of the issuer if it needs to publish the RI and note that if the connectivity of the user can replace that of the verifier. Any information the verifier needs, can be passed by the user if the information is signed by the issuer.

Mechanism	FLC	Unlinkable	Untraceable	RI (size/freq)	Online
CRL [37]	✓	✗	✗	R / R	V
OCSP [38]	✓	✗	✗	✗	I, V
VE [16]	✓	✓	✗	✗	I, V
N-times [10]	✗	✓	✗	$R \times U$ / E	V
Braavos [3]	✗	✓	✓	1 / R	U, V
ADNMP [1]	✓	✓	✗	1 / R	U, V

Table 3.1: Comparison of the described revocation mechanisms.

The Certificate Revocation List (CRL) and OCSP mechanisms were not designed with Attribute-Based Credentials in mind, and this shows in the comparison. In both mechanisms the verifier learns the RH of a credential. This makes both mechanisms linkable and traceable, although this also gives verifiers the ability to see if a credential is revoked in newer RI versions. Given that the verifier learns the RH, there is no clear solution to prevent either unlinkability or untraceability in these mechanisms.

Although the Verifiable Encryption was introduced as an anonymity revocation mechanism, it can also be used as a credential revocation mechanism by encrypting the RH under the public key of the issuer or RA. But as the issuer now learns the RH, the scheme becomes traceable. Encrypting the RH to a third party can make the scheme untraceable, but only if the third party does not collude with the issuer. However, doing so requires the verifier to trust the third party to return the correct revocation status. Furthermore, this scenario also requires an additional mechanism that allows the third party to determine the revocation status of the RH. In the N-times unlinkable proofs revocation mechanism, the RA does not play an active role during verification. The pseudonyms however, are only valid for a single epoch/RI version. By generating pseudonyms for a number of future epochs the mechanism can also provide forward-looking consistency. The traceability issue remains however, because the issuer can still generate all future pseudonyms.

The Braavos cryptographic accumulator provides both unlinkability and untraceability, due to the use of proof-of-knowledge protocols. But as the prover in these protocols requires knowledge of the RH, it does not allow a verifier to verify the membership of a handle in another version of the RI. The ADNMP homomorphic accumulator design does provide a way to "update" the non-membership proof, but this conflicts with the untraceability of credentials.

We do want to note that revocation mechanisms other than those described here exist. The described revocation mechanisms were chosen because we expected that they provide at least some required properties. Other mechanisms are similar in functionality and/or variants on the described approaches here. For a more detailed discussion on revocation mechanisms used in PABCSs, we refer to the work of Lapon [39].

Based on the comparison of related work we conclude that current revocation mechanisms are unable to provide forward-looking consistency, while adhering to the privacy properties of PABCSs. The lack of a suitable revocation mechanism for this situation means that less privacy-friendly solutions are used. We conclude that research into revocation mechanisms that address forward-looking consistency in PABCS is needed.

4 The PABC-FLC credential system design

4.1. Overview

In this chapter we present a Privacy-preserving Attribute-Based Credential System (PABCS) with forward-looking consistency, the PABC-FLC design. The design we propose has four types of participants: issuers, users, verifiers and the Non-Revocation Prover (NRP). Of these, the NRP is added in comparison to other credential systems and its function will be explained later. Although we usually describe the system as interactions between these participants, the system we propose is primarily described as a set of algorithms. To keep our design simple and focus on the forward-looking consistency aspect, we only consider a single issuer and do not take into account user pseudonyms or presenting multiple credentials at once. Our design does not exclude these however, it should be easy to see how these can be added. We will continue with a brief description of the algorithms, their intended use and examples of their usage.

Before the system can be used, all participants run their respective key generation algorithm. The system parameters are generated first, using the $SPGen$ algorithm. The issuer uses the $IssKeyGen$ algorithm to generate their signing and revocation keys, while users and the NRP use the $UserKeyGen$ and $NRPKeyGen$ algorithms to generate their own keys. After this setup, users can use the Credential Issuance protocol to obtain a credential from the issuer. A credential is a collection of attributes, agreed upon by both parties, and a signature by the issuer. Each credential contains the user secret key and a revocation handle as first two attributes.

Users can request access to various resources provided by verifiers using these credentials. To do so, they use the $CredPresentation$ algorithm to compute a presentation token (pt). This token reveals a subset of the attributes and proves that they were signed by the issuer. Presentation tokens are verified using the $VerifyCredPresentation$ algorithm. As an example, imagine a student who wants access to a university building. The university requires authentication before granting access to the building. A student can present a credential attesting that they are a student, providing (just) enough information to the university for it to grant access. When using an ABC, this authentication is based on the attributes that describe the user, not (necessarily) the unique identity of the user. In the example, showing that one is a student is enough to be granted access. More importantly, there is no need to reveal a name or unique student number.

Credential systems with a revocation mechanism, give verifiers the additional assurance that a presented credential is still valid. Previously issued credentials can be revoked by the issuer by using the $CredRevoke$ algorithm, for example when the student is no longer enrolled at the univeristy. After revoking a credential, the issuer publishes a new version of the RI to reflect the revocation. The other (unrevoked) credentials are updated using the $RevWitnessUpdate$ algorithm.

To enforce forward-looking consistency, the presentation tokens contain a redeemable revocation token (rrt). The credential verifier can exchange these tokens for a revocation token (rt) with the help of the NRP. The $CredStatusUpdate$ algorithm allows the NRP to exchange one into the other. The resulting revocation token can be verified by the credential verifier, using the $VerifyCredStatusUpdate$ algorithm. As the revocation token is computed against the latest RI, this gives credential verifiers information about the revocation status of the earlier presented credential. This information allows it to enforce forward-looking consistency. As an example, imagine the student from before applies for a student discount for a music subscription service. The company, acting as a credential verifier, can retrieve a new revocation token each month and verify if the student is still enrolled. Without involvement of the student and without learning the (unique) identity of the student in question, this allows the company to determine if the discount should still be applied. In effect, allow the verifier to enforce forward-looking consistency on the student credential.

Regarding the security of the system, we assume that credential verifiers trust the credential issuer to behave honestly when executing all its algorithms and the issuance protocol. This is a justified assumption, because if the verifier does not trust the issuer, there is no good reason for it to trust the credentials it signs. At the same time, the user is not required to trust either the issuer nor the verifier. However, the user does trust the NRP. We justify this, because an impossibility result shows no other solution is possible (see Section 4.2.7). For more details on the security, see Section 4.2.

Our design builds on the *idemix* credential system [49] and uses the Braavos revocation mechanism [3]. We combine these and extend them with the possibility to enforce forward-looking consistency. In terms of computational assumptions we rely on the Strong RSA assumption or weaker assumptions thereof (see Section A.1). Our security analysis is given in the random oracle model [6].

The rest of this chapter is structured as follows. Section 4.2 describes the requirements for a Privacy-preserving Attribute-Based Credential System (PABCS) with forward-looking consistency. These requirements resemble the formulations of [22], but are less formal and have been adapted to our use case. Section 4.3 proposes PABC-FLC, a credential system that adheres to these requirements. Section 4.4 describes the security analysis, showing that the proposed design adheres to the formulated requirements. The security analysis concludes with a short comparison of this work with other revocation mechanisms, as described in the related work (see Chapter 3). An overview of symbols used in this chapter, and their explanation, is given in the appendix in Table A.1.

4.2. Requirements

4.2.1. Security setting

In the context of a credential system where participants communicate with eachother, we assume the existence of secure channels that authenticate the receiving participant. Furthermore, we assume that the issuer and NRP can be contacted at any time and that the verifier can be contacted by a user who wants to authenticate. There is no availability requirement for the user(s). In fact, we are specifically designing a credential system for situations in which this is not possible.

Additionally, we assume that a user does not share their *user secret* with anyone else. In practice, ensuring *non-transferability* of the user secret is a challenging problem. Several approaches to this problem are given in [13, 35].

4.2.2. Correctness

Correctness refers to the notion that the credential system can be used by all honest participants. In the design we propose this means that

- participants can run their key generation algorithm,
- users can obtain credentials from the issuer,
- users can create presentation tokens for unrevoked credentials,
- presentation tokens for unrevoked credentials can be verified,
- unrevoked credentials can be revoked by the issuer,
- the revocation witness for a credential can be updated if another credential is revoked,
- the Non-Revocation Prover (NRP) can provide a revocation token for unrevoked credentials when given a redeemable revocation token,
- revocation tokens for unrevoked credentials can be verified.

4.2.3. Credential unforgeability

Credentials are unforgeable if $VerifyCredential$ only returns true for credentials obtained through the $CredentialIssuance$ protocol, i.e. the credential was signed by the issuer and conforms to the attribute length requirements.

4.2.4. Presentation token unforgeability

Credential verifiers need to be assured that if a presentation token verifies as true, the credential user that generated the token possesses a credential with the presented attributes. Recall that we assume verifiers trust the issuer to behave honestly. Presentation tokens are considered unforgeable if

- the presented credential was issued to the user that presents it,
- the presented credential was signed by the issuer,
- the revealed attributes correspond to the attributes in the credential,
- the redeemable revocation token contains a commitment to the revocation handle.

4.2.5. Revocation token unforgeability

Similar to the unforgeability of presentation tokens, we require that revocation tokens can only be computed for revocation handles that were generated by the issuer and are still contained in the Revocation Information (RI) (i.e. not revoked). Due to the working of the Braavos revocation mechanism, we require that a redeemable revocation token (rrt) used to compute a revocation token (rt) comes from a (valid) presentation token.

4.2.6. Presentation token privacy

We call presentation tokens private if they are multi-show unlinkable and untraceable, when no attributes are revealed. In the context of issuers and verifiers, this means that even when a malicious issuer colludes with malicious verifiers, they are unable to determine which user presented a credential. Multi-show unlinkability means that given two presentation tokens, it is not possible to determine if they were generated from the same or two different credentials. This also implies that given two credentials, it is not possible to determine which of two different credentials was used to generate a presentation token (*unlinkability*). *Untraceability* means that it is not impossible for a (malicious) issuer to determine which credential was used when it is given a presentation token. Definition 2 captures *multi-show unlinkability* and *untraceability* of unrevoked credentials.

Definition 2 (Presentation token privacy)**.** Presentation tokens are private, if for every efficient adversary A there exists a negligible function ν such that the following holds

$$\begin{aligned}
&Pr[b' = b \wedge pt_0 \neq \perp \wedge pt_1 \neq \perp : \\
&spar \leftarrow SPGen(1^\kappa, 1^\lambda), \\
&(pk_{NRP}, sk_{NRP}) \leftarrow NRPKeyGen, \\
&(pk_{Iss}, sk_{Iss}, \sigma_{IssKeyGen}, cred_0, cred_1, st) \leftarrow A, \\
&pt_0 \leftarrow CredPresentation(pk_{Iss}, pk_{NRP}, cred_0, \{\}, 0), \\
&b \leftarrow_R \{0,1\}, pt_b \leftarrow CredPresentation(pk_{Iss}, pk_{NRP}, cred_b, \{\}, 0), \\
&b' \leftarrow A(pt_0, pt_b, st)] \leq \frac{1}{2} + \nu(\kappa, \lambda)
\end{aligned}$$

4.2.7. Revocation token privacy

The privacy of revocation tokens is similar to the requirement for privacy of presentation tokens. It means that given two unrevoked credentials, the revocation tokens generated from a (possibly adversarially generated) redeemable revocation token are multi-show unlinkable. The (multi-show) unlinkability of redeemable revocation tokens is implied by the privacy of presentation tokens, and not included in this definition. Note that only two unrevoked credentials with the same Revocation Information (RI) are considered, this is due to what we describe as the "revoke all but one attack". Definition 3 describes revocation token privacy.

Remark ("Revoke all but one attack"). This attack refers to the situation in which a malicious issuer and malicious verifier collude and break the unlinkability of revocation tokens. See also the definition of revocation privacy in [21]. More specifically, this attack is possible when an adversary has access to an oracle that returns a revocation token for any (reference to a) revocation handle when given any RI. This oracle represents the verifier's ability to enforce forward-looking consistency.

The attack works as follows. During issuance, the adversary can store all issued revocation handles in state st. When given (a reference to) a revocation handle, the adversary can use the stored information to compute a collection of RI's, with each RI containing a single unrevoked revocation handle. It can then use the oracle to obtain revocation tokens for all computed RI's. The returned revocation tokens can be verified to determine which RI contained the referenced revocation handle. This allows the adversary to break unlinkability, since it now knows which revocation handle was used to compute a revocation token.

Definition 3 (Revocation token privacy)**.** Revocation tokens are private, if for every efficient adversary A there exists a negligible function ν such that the following holds

$$
\begin{aligned}
&Pr[b' = b \wedge pt_0 \neq \bot \wedge pt_1 \neq \bot \wedge rt_0 \neq \bot \wedge rt_1 \neq \bot :\\
&spar \leftarrow SPGen(1^\kappa, 1^\lambda),\\
&(pk_{NRP}, sk_{NRP}) \leftarrow NRPKeyGen, NRPStore \leftarrow \{\}\\
&(pk_{Iss}, sk_{Iss}, \sigma_{IssKeyGen}, st) \leftarrow A,\\
&(cred_0, cred_1, w_0, w_1, RI) \leftarrow A(st),\\
&(rh_0, rh_1) \leftarrow (cred_0, cred_1), NRPStore \leftarrow NRPStore \cup \{(rh_0, w_0), (rh_1, w_1)\},\\
&\{pt_i \leftarrow CredPresentation(pk_{Iss}, pk_{NRP}, cred_0, \{\}, 0)\}_{i\in\{0,1\}},\\
&(rrt_0, rrt_1) \leftarrow (pt_0, pt_1),\\
&rt_0 \leftarrow CredStatusUpdate(pk_{Iss}, RI, rrt_0, pk_{NRP}, sk_{NRP}, NRPStore, 0),\\
&b \leftarrow_R \{0,1\}, rt_b \leftarrow CredStatusUpdate(pk_{Iss}, RI, rrt_b, pk_{NRP}, sk_{NRP}, NRPStore, 0),\\
&b' \leftarrow A(pt_0, pt_1, rt_0, rt_b, st)] \leq \frac{1}{2} + \nu(\kappa, \lambda)
\end{aligned}
$$

4.3. Credential system algorithms and protocols

4.3.1. System parameter generation

The system parameter generation algorithm $SPGen(1^\kappa, 1^\lambda) \rightarrow spar$ takes as input computational and statistical security parameters $1^\kappa, 1^\lambda$ and outputs parameters $(1^{\ell_N}, 1^{\ell_\emptyset}, 1^{\ell_c}, 1^{\ell_m}, 1^{\ell_{rh}}, 1^{\ell_e}, 1^{\ell'_e}, 1^{\ell_v}, 1^{\ell_{enc}})$. All algorithms, except *SPGen*, take $spar$ as an implicit input. The *idemix* specification [49] contains a list of suggested parameters. We add the requirement that $\ell_{rh} + \ell_n + \ell_\emptyset + 2\ell_c + 2 < \ell_{enc}$. To match the *idemix* parameters, one can use $\ell_{enc} = 3072$.

4.3.2. Issuer key generation

Before the issuer can issue credentials, it is required to generate appropriate keys for the signature, commitment and revocation schemes. The $IssKeyGen(\ell_{Att}) \to (pk_{Iss}, sk_{Iss}, RI_0)$ algorithm is specified in Algorithm 10 and will be described here briefly.

The algorithm takes as input the number of attributes ℓ_{Att} in the credentials. It outputs a keypair for the signature scheme (pk_{Sig}, sk_{Sig}), a keypair for the revocation scheme (pk_{Acc}, sk_{Acc}), the first version of the Revocation Information (RI) RI_0, and a proof of the correct generation of the public keys $\sigma_{IssKeyGen}$. The public keys and SPK are bundled in the issuer public key pk_{Iss}, the secret keys are similarly bundled in sk_{Iss}.

The algorithm starts by generating a signing key pair for the Camenisch-Lysyanskaya signature scheme [13]. To generate the special RSA modulus, Algorithm 9 is used. It continues by generating a keypair for the CL-RSA-B accumulator [3] and commitment key for the integer commitment scheme [30, 27]. After the generation of all keys, it computes an initial version of the Revocation Information (RI) RI_0. Finally, $SPK_{IssKeyGen}$ (see Protocol 11) attests to the correct setup and is used to sign the public key values.

Algorithm 9 Generate Special RSA modulus

```
1: function GenSpecialRSA($1^{\ell_N}$)
2:     Generate $p = 2p' + 1$ s.t. $p$ has length $\frac{\ell_N}{2}$ and $p, p'$ are both prime
3:     Generate $q = 2q' + 1$ s.t. $q$ has length $\frac{\ell_N}{2}$ and $q, q'$ are both prime
4:     $N \leftarrow pq$
5:     return $(N, (p', q'))$
6: end function
```

Protocol 11 is used in the signature variant by the credential issuer to sign the public keys. The protocol is used to convince the other participants that $Z, \{R_i\}_{i \in Att} \in \langle S \rangle, G \in \langle H \rangle$ by showing that the issuer knows the discrete log of these values regarding generator S. Note that the issuer acts as prover in the protocol. The protocol follows that of [44], which is sound if ℓ_c is chosen large enough. The resulting protocol can be turned into a signature variant using the Fiat-Shamir transformation [29] (see Section 2.2) and can be statistically simulated provided that $\ell_\emptyset$ is chosen large enough.

Other participants of the credential system can verify the correct generation of the issuer key by verifying $\sigma_{IssKeyGen}$. This verification is used implicitly by all (honest) participants when they use the issuer public key.

4.3.3. User key generation

Users generate a user secret m_{us}, before they engage in the issuance protocol. $UserKeyGen \to m_{us}$ is specified in Algorithm 12.

4.3.4. Non-Revocation Prover setup

The Non-Revocation Prover (NRP) is contacted by a verifier when they want to know if an earlier presented credential has been revoked. If the credential was not revoked, the NRP returns a revocation token attesting to this fact. If it was revoked and the NRP communicates

Algorithm 10 Issuer Key Generation

```
1: function IssKeyGen($\ell_{Att}$)
2:     $Att \leftarrow \{us, rh, 1, \ldots, \ell_{Att} - 2\}$                ▷ $Att = \{us, rh\} \cup Att_k$
3:     $(N_{Sig}, sk_{Sig}) \leftarrow$ GenSpecialRSA($1^{\ell_N}$)          ▷ $sk = (p', q'), \#QR_N = p'q'$
4:     $S \leftarrow_R QR_{N_{Sig}}$                                        ▷ s.t. $S$ is a generator
5:     $\{r_i \leftarrow_R \{0,1\}^{\ell_N}\}_{i \in Att \cup \{z\}}$
6:     $Z \leftarrow S^{r_z} \bmod N_{Sig}, \{R_i \leftarrow S^{r_i} \bmod N_{Sig}\}_{i \in Att}$
7:     $pk_{Sig} \leftarrow (Att, N_{Sig}, S, Z, \{R_i\}_{i \in Att})$

8:     $(N_{Acc}, sk_{Acc}) \leftarrow$ GenSpecialRSA($1^{\ell_N}$)          ▷ $sk = (p', q'), \#QR_N = p'q'$
9:     $Acc, G \leftarrow_R QR_{N_{Acc}}$                                   ▷ s.t. $G$ is a generator
10:    $r_g \leftarrow_R \{0,1\}^{\ell_N}$
11:    $G \leftarrow H^{r_g} \bmod N_{Acc}$
12:    $RI_0 \leftarrow (Acc, 0)$
13:    $pk_{Acc} \leftarrow (N_{Acc}, G, H)$

14:    $\sigma_{IssKeyGen} \leftarrow SPK_{IssKeyGen}\{(\{r_i\}_{i \in Att \cup \{z,g\}}):$
         $Z \equiv S^{r_z} \pmod{N_{Sig}}$
         $\bigwedge \{R_i \equiv S^{r_i} \pmod{N_{Sig}}\}_{i \in Att}$
         $\bigwedge G \equiv H^{r_g} \pmod{N_{Acc}}$
         $\}()$

15:    $pk_{Iss} \leftarrow (pk_{Sig}, pk_{Acc}, \sigma_{IssKeyGen})$
16:    $sk_{Iss} \leftarrow (sk_{Sig}, sk_{Acc})$
17:    return $(pk_{Iss}, sk_{Iss}, RI_0)$
18: end function
```

Protocol 11 $PK_{IssKeyGen}$

Prover	**Public**	**Verifier**
$r_z, r_g, \{r_i\}_{i\in Att}$	$N_{Sig}, S, Z, \{R_i\}_{i\in Att}, N_{Acc}, H, G$	
$a_z \leftarrow_R \{0,1\}^{\ell_N+\ell_c+\ell_\emptyset}$		
$\{a_i \leftarrow_R \{0,1\}^{\ell_N+\ell_c+\ell_\emptyset}\}_{i\in Att}$		
$a_g \leftarrow_R \{0,1\}^{\ell_N+\ell_c+\ell_\emptyset}$		
$t_z \leftarrow S^{a_z} \bmod N_{Sig}$		
$\{t_i \leftarrow S^{a_i} \bmod N_{Sig}\}_{i\in Att}$		
$t_g \leftarrow H^{a_g} \bmod N_{Acc}$		
	$\xrightarrow{t_z,\{t_i\}_{i\in Att},t_g}$	
		$c \leftarrow_R \{0,1\}^{\ell_c}$
	$\xleftarrow{c}$	
$s_z \leftarrow a_z + cr_z$		
$\{s_i \leftarrow a_i + cr_i\}_{i\in Att}$		
$s_g \leftarrow a_g + cr_g$		
	$\xrightarrow{s_z,\{s_i\}_{i\in Att},s_g}$	
		Verify $t_z \equiv Z^{-c}S^{s_z} \pmod{N_{Sig}}$
		Verify $\{t_i \equiv R_i^{-c}S^{s_i} \pmod{N_{Sig}}\}_{i\in Att}$
		Verify $t_g \equiv G^{-c}H^{s_g} \pmod{N_{Acc}}$

Algorithm 12 User Key Generation

```
1: function USERKEYGEN
2:     return $m_{us} \leftarrow_R \{0,1\}^{\ell_m}$
3: end function
```

this, the verifier can enforce forward-looking consistency and suspend the service(s) they provided to a user. The key generation algorithm $NRPGenKey \rightarrow (pk_{NRP}, sk_{NRP})$ is further specified in Algorithm 13.

As users need to convey secret information to the Non-Revocation Prover (NRP), the NRP requires a public encryption key that can be used for this purpose. Let $Enc = (KeyGen, Encrypt, Decrypt)$ be a public-key encryption scheme. The function $Enc.KeyGen(1^\kappa) \rightarrow (pk, sk)$ takes the computational security parameter 1^κ as input and returns a keypair (pk, sk). A message m can be encrypted under the public key pk with $Enc.Encrypt(pk, m) \rightarrow ct$, returning a ciphertext message. The ciphertext can be decrypted using the secret key sk using $Enc.Decrypt(sk, ct) \rightarrow m/\perp$. In this design we use RSA-OAEP scheme [5], which is IND-CCA2 under the random oracle assumption and hardness assumption of the RSA problem. Both are no stronger assumptions than what we already assume for other security definitions of the credential system.

Although not part of the key generation algorithm, the NRP keeps track of revocation handle and witness tuples (m_{rh}, w) in the (initially empty) set $NRPStore = \{\}$.

Algorithm 13 NRP Key Generation

```
1: function NRPKeyGen
2:     $(pk_{NRP}, sk_{NRP}) \leftarrow_R Enc.KeyGen(1^{\ell_{enc}})$
3:     return $(pk_{NRP}, sk_{NRP})$
4: end function
```

4.3.5. Credential issuance

During credential issuance, a user receives a new credential from the issuer. Before issuance, both parties must agree on the known attribute values $\{m_i\}_{\forall i \in A_k}$ that will be encoded in the credential. Note that the known attribute values, as all attribute values, are from the set $\{0,1\}^{\ell_m}$. Additionally, the user has user secret m_{us} and the issuer has secret keys sk_{Iss}. Both have access to the issuer public key pk_{Iss} and latest RI. Issuance protocol $\langle User(m_{us}), Issuer(sk_{Iss})\rangle(pk_{Iss}, RI, \{m_i\}_{\forall i \in A_k}) \rightarrow (cred, w)/\perp$ is specified in Protocol 14.

The issuer starts by sending a nonce n_1 to the user. The issuer nonce, like the user nonce introduced later, are used to prove freshness. The user continues by computing the partial credential U, which includes their user secret m_{us}. They prove that U is computed correctly by signing the nonce n_1 with a SPK. The SPK P_1, U and generated nonce n_2 are sent to the issuer. The issuer verifies P_1, aborting if this fails, and generates RH m_{rh} with associated witness w. The revocation handle is chosen from the set of all revocation handles, in this design all odd primes in $\{0,1\}^{\ell_m}$. It follows by computing a signature over the credential and proves knowledge of the singing key with SPK P_2. The signature σ_{Iss}, generated revocation handle m_{rh} associated witness w and SPK P_2 are returned to the user. The user follows by verifying P_2 and checking that the generated values e, m_{rh} are prime numbers of the appropriate length. Next, it checks that the credential was signed correctly and the revocation handle is contained in the accumulator. The protocol ends with the user storing the attribute values, Revocation Witness (Revocation Witness) and associated accumulator value Acc, and the signature itself.

For each credential the user obtains, the revocation handle m_{rh} and witness w are sent to the NRP. Knowledge of these is required for creating a revocation token. The NRP verifies that the membership relation $Acc \overset{?}{\equiv} w^{m_{rh}} \pmod{N_{Acc}}$ holds and stores both values $NRPStore \leftarrow NRPStore \cup (m_{rh}, w_i)$. If the NRP notices that a newer RI version is published by the issuer, it uses the revocation witness update algorithm to update the witness (see Section 4.3.9).

4.3.6. Credential verification

Using algorithm $CredVerify(pk_{Iss}, cred) \rightarrow true/false$ (see Algorithm 15), it can be verified that a credential — meaning a set of attributes and a signature — conforms to the length requirements and was signed by pk_{Iss}.

4.3.7. Credential presentation

Credentials can be used to create presentation tokens. Presentation tokens are requested by verifiers to authenticate users that request access to a service or resource. For example, a music subscription company may request a proof of enrolment at some university before applying a monthly student discount requested by the user. The presentation token shows that the user has a requested proof of enrolment credential and also provides the verifier with a redeemable revocation token. The redeemable revocation token can be exchanged for a revocation token at a later point in time.

The $CredPresentation(pk_{Iss}, pk_{NRP}, cred, Att_r, n_{pres}) \rightarrow pt/\bot$ algorithm is used to create a presentation token for a credential $cred$. Beforehand, the user and verifier agree on the set of attributes A_r that will be revealed. The verifier also specifies a presentation nonce n_{pres}, which prevents replay attacks of the presentation token. The complete algorithm is specified in Algorithm 16.

The algorithm starts by creating a commitment (see Section 2.3) to the RH m_{rh} in the credential. This commitment is later used to tie revocation tokens to the RH from this credential. As the NRP also needs to open the commitment, both m_{rh} and r_1 are encrypted under the public encryption key pk_{NRP}. The ciphertext ct together with the commitment C_{rh} is called a redeemable revocation token (rrt). After computing the rrt, the algorithm follows a regular *idemix* credential presentation [49]. This involves randomizing the credential signature (A, e, v) and computing a SPK that proves the credential is signed by the issuer and contains the revealed attributes. It also proves that the value committed to in C_{rh} equals the value of the revocation handle m_{rh}. The SPK is used to sign the nonce n_{pres}. See Protocol 17 for details on this protocol, as well as the following section(s). At the end of the algorithm, the presentation token pt is returned.

Presentation tokens can be verified using the $VerifyCredPresentation(pk_{Iss}, pt, Att_r, n_{pres}) \rightarrow true/false$ algorithm specified in Algorithm 18.

4.3.8. Credential revocation

Credentials are revoked to prevent a verifier from relying on them during or after authentication. In our design, the issuer is responsible for revoking credentials. One reason that the issuer may decide to revoke a credential, is when the encoded attributes have changed or

Protocol 14 Credential issuance

```
User                                   Public                                          Issuer
$m_{us}$                               $\{m_i\}_{\forall i \in A_k}$
                                       $pk_{Iss} = (pk_{Sig}, pk_{Acc}, \sigma_{IssKeyGen})$
                                       $RI = (Acc, m_{revoked})$                       $sk_{Sig}, sk_{Acc}$
--------------------------------------------------------------------------------------------------------
                                                                                       $n_1 \leftarrow_R \{0,1\}^{\ell_\varnothing}$
                                       $\xleftarrow{n_1}$
$v' \leftarrow_R \{0,1\}^{\ell_N + \ell_\varnothing}$
$U \leftarrow S^{v'} R_{us}^{m_{us}} \bmod N_{Sig}$

$P_1 \leftarrow SPK\{(v', m_{us}):$
    $U \equiv S^{v'} R_{us}^{m_{us}} \pmod{N_{Sig}}$
    $\bigwedge m_{us} \in \{0,1\}^{\ell_m + \ell_\varnothing + \ell_c + 1}$
$\}(n_1)$

$n_2 \leftarrow_R \{0,1\}^{\ell_\varnothing}$
                                       $\xrightarrow{U, P_1, n_2}$
                                                                                       **Verify** $P_1$
                                                 $m_{rh} \leftarrow_R [2^{\ell_{rh}-1}, 2^{\ell_{rh}} - 1]$ s.t. $m_{rh}$ is an odd prime
                                                           $w \leftarrow Acc^{m_{rh}^{-1} \bmod \#QR_{NAcc}} \bmod N_{Acc}$
                                                           $prime\ e \leftarrow_R [2^{\ell_e - 1}, 2^{\ell_e - 1} + 2^{\ell_{e'} - 1}]$
                                                                                       $\tilde{v} \leftarrow_R \{0,1\}^{\ell_v - 1}$
                                                                                       $v'' \leftarrow 2^{\ell_v - 1} + \tilde{v}$
                                                           $Q \leftarrow \frac{Z}{U S^{v''} R_{rh}^{m_{rh}} \prod_{\forall i \in A_k} R_i^{m_i}} \bmod N_{Sig}$
                                                           $A \leftarrow Q^{e^{-1} \bmod \#QR_{NSig}} \bmod N_{Sig}$
                                                                                       $\sigma_{Iss} \leftarrow (A, e, v'')$

                                                                                       $P_2 \leftarrow SPK\{(e^{-1}):$
                                                           $A \equiv Q^{e^{-1}} \pmod{N_{Sig}}\}(n_2)$

                                       $\xleftarrow{m_{rh}, w, \sigma_{Iss}, P_2}$
**Verify** $e \in [2^{\ell_e - 1}, 2^{\ell_e - 1} + 2^{\ell_{e'} - 1}]$ and is prime
$v \leftarrow v'' + v'$
$Q \leftarrow \frac{Z}{S^v \prod_{i \in A} R_i^{m_i}} \bmod N_{Sig}$
$\widehat{Q} \leftarrow A^e \bmod N_{Sig}$
**Verify** $P_2$
**Verify** $Q = \widehat{Q}$
**Verify** $m_{rh} \in [2^{\ell_{rh}-1}, 2^{\ell_{rh}} - 1]$ and is prime
**Verify** $Acc \equiv w^{m_{rh}} \pmod{N_{Acc}}$
**Store** $cred \leftarrow ((A, e, v), \{m_i\}_{i \in Att}), w$
```

Algorithm 15 Credential Verification

```
1: function CREDVERIFY($pk_{Iss}, cred$)
2:     $(pk_{Sig}, pk_{Acc}, \sigma_{IssKeyGen}) \leftarrow pk_{Iss}$
3:     $(Att, N_{Sig}, S, Z, \{R_i\}_{i \in Att}) \leftarrow pk_{Sig}$
4:     $((A, e, v), \{m_i\}_{i \in Att}) \leftarrow cred$

5:     for $a \in Att$ do
6:         if $m_a \notin \{0,1\}^{\ell_m}$ then
7:             return $false$
8:         end if
9:     end for
10:    if $m_{rh} \notin [2^{\ell_{rh}-1}, 2^{\ell_{rh}} - 1]$ then
11:        return $false$
12:    end if
13:    if $e \notin [2^{\ell_e - 1}, 2^{\ell_e - 1} + 2^{\ell_{e'} - 1}]$ or $e$ is not prime then
14:        return $false$
15:    end if
16:    if $Z \not\equiv A^e S^v R_{us}^{m_{us}} R_{rh}^{m_{rh}} \prod_{i \in Att} R_i^{m_i} \pmod{N_{Sig}}$ then
17:        return $false$
18:    end if

19:    return $true$
20: end function
```

Algorithm 16 Credential Presentation Token Generation

```
1: function CREDPRESENTATION($pk_{Iss}, pk_{NRP}, cred, Att_r, n_{pres}$)
2:     $(pk_{Sig}, pk_{Acc}, \sigma_{IssKeyGen}) \leftarrow pk_{Iss}$
3:     $(Att, N_{Sig}, S, Z, \{R_i\}_{i \in Att}) \leftarrow pk_{Sig}$
4:     $(N_{Acc}, G, H) \leftarrow pk_{Acc}$
5:     $((A, e, v), \{m_i\}_{i \in Att}) \leftarrow cred$

6:     if CREDVERIFY($pk_{Iss}, cred$) $\neq true$ then
7:         return $\perp$
8:     end if

9:     $o_{Com} \leftarrow_R \{0,1\}^{\ell_N + \ell_\emptyset}$
10:    $C_{rh} \leftarrow G^{m_{rh}} H^{o_{Com}} \bmod N_{Acc}$
11:    $ct \leftarrow$ ENC.ENCRYPT($pk_{NRP}$, $m_{rh} || o_{Com}$)
12:    $rrt \leftarrow (C_{rh}, ct)$

13:    $r_A \leftarrow_R \{0,1\}^{\ell_N + \ell_\emptyset}$
14:    $A' \leftarrow AS^{r_A} \bmod N_{Sig}$
15:    $v' \leftarrow v - er_A$

16:    $\sigma_{pres} \leftarrow SPK_{CredPresentation}\{(e, v', m_{us}, m_{rh}, \{m_i\}_{i \notin Att_r}, o_{Com}):$
         $\frac{Z}{\prod_{i \in Att_r} R_i^{m_i}} \equiv A'^e S^{v'} R_{us}^{m_{us}} R_{rh}^{m_{rh}} \prod_{i \notin Att_r} R_i^{m_i} \pmod{N_{Sig}}$
         $\bigwedge e - 2^{\ell_e - 1} \in \{0,1\}^{\ell'_e + \ell_\emptyset + \ell_c + 1}$
         $\bigwedge C_{rh} \equiv G^{m_{rh}} H^{o_{Com}} \pmod{N_{Acc}}$
         $\bigwedge \{m_i \in \{0,1\}^{\ell_m + \ell_\emptyset + \ell_c + 1}\}_{i \notin Att_r}$
         $\bigwedge m_{us} \in \{0,1\}^{\ell_m + \ell_\emptyset + \ell_c + 1}$
         $\bigwedge m_{rh} \in \{0,1\}^{\ell_{rh} + \ell_\emptyset + \ell_c + 1}$
         $\}(n_{pres})$
17:    return $pt \leftarrow (A', \{m_i\}_{i \in Att_r}, \sigma_{pres}, rrt)$
18: end function
```

Protocol 17 $PK_{CredPresentation}$

Prover	**Public**	**Verifier**
$e, v', m_{us}, m_{rh}, \{m_i\}_{i \notin Attr_r}, r_{Com}$	$pk_{Iss}, A', \{m_i\}_{i \in Attr_r}, C_{rh}$	
$\{r_i \leftarrow_R \{0,1\}^{\ell_m+\ell_c+\ell_\emptyset}\}_{i \notin Attr_r}$		
$r_{us} \leftarrow_R \{0,1\}^{\ell_m+\ell_c+\ell_\emptyset}$		
$r_{rh} \leftarrow_R \{0,1\}^{\ell_{rh}+\ell_c+\ell_\emptyset}$		
$r_e \leftarrow_R \{0,1\}^{\ell'_e+\ell_c+\ell_\emptyset}$		
$r_{v'} \leftarrow_R \{0,1\}^{\ell_v+\ell_c+\ell_\emptyset}$		
$r_c \leftarrow_R \{0,1\}^{\ell_n+\ell_c+\ell_\emptyset}$		
$t_{cred} \leftarrow A'^{r_e} S^{r_{v'}} R_{us}^{r_{us}} R_{rh}^{r_{rh}} \prod_{i \notin Attr_r} R_i^{r_i} \bmod N_{Sig}$		
$t_{com} \leftarrow G^{r_{rh}} H^{r_c} \bmod N_{Acc}$		
	$\xrightarrow{t_{cred}, t_{com}}$	
		$c \leftarrow_R \{0,1\}^{\ell_c}$
	$\xleftarrow{c}$	
$\{s_i \leftarrow r_i + cm_i\}_{i \notin Attr_r}$		
$s_{us} \leftarrow r_{us} + cm_{us}$		
$s_{rh} \leftarrow r_{rh} + cm_{rh}$		
$s_c \leftarrow r_c + cr_{Com}$		
$s_e \leftarrow r_e + c(e - 2^{\ell_e - 1})$		
$s_{v'} \leftarrow r_{v'} + cv'$		
	$\xrightarrow{\{s_i\}_{i \in Attr_r}, s_{us}, s_{rh}, s_c, s_e, s_{v'}}$	
		Verify $\{s_i \in \{0,1\}^{\ell_m+\ell_c+\ell_\emptyset+1}\}_{i \in Attr_r}$
		Verify $s_{us} \in \{0,1\}^{\ell_m+\ell_c+\ell_\emptyset+1}$
		Verify $s_{rh} \in \{0,1\}^{\ell_{rh}+\ell_c+\ell_\emptyset+1}$
		Verify $s_e \in \{0,1\}^{\ell'_e+\ell_c+\ell_\emptyset+1}$
		$B \leftarrow \frac{Z}{(A')^{2^{\ell_e-1}} \prod_{i \notin Attr_r} R_i^{m_i}} \bmod N_{Sig}$
		Verify $t_{cred} \equiv B^{-c} A'^{s_e} S^{s_{v'}} R_{us}^{s_{us}} R_{rh}^{s_{rh}} \prod_{i \in Attr_r} R_i^{s_i} \pmod{N_{Sig}}$
		Verify $t_{com} \equiv C_{rh}^{-c} G^{s_{rh}} H^{s_c} \pmod{N_{Acc}}$

Algorithm 18 Verify Credential Presentation Token

```
1: function VERIFYCREDPRESENTATION($pk_{Iss}, pt, Attr_r, n_{pres}$)
2:     $(A', \{m_r\}_{r \in Attr_r}, \sigma_{pres}, rrt) \leftarrow pt$
3:     $(C_{rh}, ct) \leftarrow rrt$
4:     verify $\{m_r \in \{0,1\}^{\ell_m}\}_{r \in Attr_r}$
5:     return verify $\sigma_{pres}$                                ▷ See Protocol 17
6: end function
```

turned out to be incorrect. Another reason may be that the issuer no longer wants a user to use the credential.

To revoke revocation handle m_{rh} and produce new Revocation Information (RI) RI_{i+1}, the algorithm $CredRevoke(pk_{Iss}, sk_{Iss}, RI_i, m_{rh}) \rightarrow RI_{i+1} = (Acc_{i+1}, m_{rh}/\perp$ is used. This algorithm is specified in Algorithm 20.

During credential issuance, a RH m_{rh} generated by the issuer is encoded in the credential. At the same time, the user is provided with a Revocation Witness (Revocation Witness) that can be used to prove the handle is contained in the accumulator value Acc. By removing the handle m_{rh} from the accumulator, its membership can no longer be proven by a user. (A credential user cannot update their witness if the revocation handle is removed from the accumulator, nor is it able to compute a witness by itself.) After removing the handle, the issuer publishes the new accumulator and updates the set of revoked handles. The set of revoked handles is used by other users to update their witness, see Section 4.3.9. The algorithm specified in Algorithm 19 follows that of the CL-RSA-B accumulator [3].

Algorithm 19 Credential Revocation [3]

```
1: function CREDREVOKE($pk_{Iss}, sk_{Iss}, RI_i, m_{rh}$)
2:     $(pk_{Sig}, pk_{Acc}, \sigma_{IssKeyGen}) \leftarrow pk_{Iss}$
3:     $(sk_{Sig}, sk_{Acc}) \leftarrow sk_{Iss}$
4:     $N_{Acc} \leftarrow pk_{Acc}$
5:     $(p', q') \leftarrow sk_{Acc}$
6:     $(Acc, RevokedHandle) \leftarrow RI_i$
7:     if $m_{rh} \notin [2^{\ell_{rh}-1}, 2^{\ell_{rh}} - 1]$ or not prime then
8:         return $\perp$
9:     end if                                         ▷ Verify $m_{rh}$ is a revocation handle
10:    $Acc_{i+1} \leftarrow Acc^{m_{rh}^{-1} \bmod p'q'} \bmod N_{Acc}$      ▷ $\#QR_{N_{Acc}} = p'q'$
11:    return $RI_{i+1} \leftarrow (Acc_{i+1}, m_{rh})$
12: end function
```

4.3.9. Revocation witness update

Each time a credential is revoked, a new accumulator value is computed. When the accumulator value changes, the previously computed witness w can no longer be used to prove membership of m_{rh} in the new accumulator value.

The algorithm $RevWitnessUpdate(pk_{Iss}, m_{rh}, w_{i-1}, Acc_{i-1}, RI_i) \rightarrow w_i/\perp$, specified in Algorithm 20, computes the witness value for a new accumulator version. (Provided that the credential in question is not the one getting revoked.) It follows the algorithm specified for the CL-RSA-B accumulator [3]. Note that witness updates are linear in the number of revocations and do not depend on the total number of credentials.

When the NRP uses the witness update algorithm to update the set $NRPStore$, it replaces the witness in each tuple with the output of the algorithm.

Algorithm 20 Revocation Witness Update [3]

```
1: function RevWitnessUpdate($pk_{Iss}, m_{rh}, w_{i-1}, Acc_{i-1}, RI_i$)
2:     $(pk_{Sig}, pk_{Acc}, \sigma_{IssKeyGen}) \leftarrow pk_{Iss}$
3:     $(Acc_i, m_{revoked}) \leftarrow RI_i$
4:     if $m_{rh} = m_{revoked}$ then
5:         return $\perp$
6:     end if
7:     Compute Bézout coefficients $b, c$ s.t. $bm_{rh} + cm_{revoked} = 1$
8:     $w_i \leftarrow w_{i-1}^c Acc_{i-1}^b \bmod N_{Acc}$
9:     return $w_i$
10: end function
```

4.3.10. Credential status update

The defining aspect of forward-looking consistency is that a verifier can determine the revocation status of a credential that was presented before. The verifier can contact the NRP, provide a redeemable revocation token obtained from a presentation token and ask if the credential is still unrevoked. The NRP can compute a revocation token attesting to the fact that it was not, if that is indeed the case. The verifier can contact the NRP at a decision moment, when it is possible to take action if the earlier presented credential turn out to be revoked. In the example of subscription discounts, when the company sends out a monthly invoice it can use this moment to check if the criteria for a discount are still met or if these should no longer be applied. This is checked by verifying that the presented credential was not revoked.

The NRP uses the $CredStatusUpdate(pk_{Iss}, RI, rrt, pk_{NRP}, sk_{NRP}, NRPStore, n_{rrt}) \rightarrow rt/\perp$ algorithm to compute a revocation token rt for a redeemable revocation token rrt. The NRP received a witness from the user after credential issuance and stored this in the $NRPStore$ (see Section 4.3.5). The NRP also kept the witness up-to-date when newer versions of the RI were published. The algorithm is given in Protocol 21.

After receiving a nonce $n_{rrt} \in_R \{0,1\}^{\ell_\varnothing}$ and redeemable revocation token ciphertext ct from a verifier, the NRP begins by decrypting the ciphertext ct. If decryption fails or the encrypted RH is not known in $NRPStore$, the algorithm aborts. The NRP continues with reconstructing the commitment C_{rh}. Following the (re)construction of the commitment, two other commitments are made to the witness and the randomizer of the witness commitment. These commitments and the β, δ values are used in the SPK proof of membership. The signature proof-of-membership follows that of [14] and is used to sign the nonce n_{rrt}.

The revocation token can be verified using the $CredStatusVerify(pk_{Iss}, C'_{rh}, rt) \rightarrow true$ $/false$ algorithm, specified in Algorithm 22. In addition to the revocation token rt, it also takes the commitment C'_{rh} from the earlier presentation token and nonce n_{rrt} as input. The commitment from the stored rrt is used to verify signature $\sigma_{CredStatusUpdate}$.

Algorithm 21 Credential Status Update

```
1: function CREDSTATUSUPDATE($pk_{Iss}, RI, ct, pk_{NRP}, sk_{NRP}, NRPStore, n_{rrt}$)
2:     $(pk_{Sig}, pk_{Acc}, \sigma_{IssKeyGen}) \leftarrow pk_{Iss}$
3:     $N_{Acc} \leftarrow pk_{Acc}$
4:     $Acc \leftarrow RI$
5:     $d \leftarrow Enc.Decrypt(sk_{NRP}, ct)$
6:     if $d = \perp$ then
7:         return $\perp$
8:     end if
9:     $m_{rh} || r_1 \leftarrow d$
10:    $(m_{rh}, w) \in NRPStore$, if this tuple does not exist, output $\perp$.
11:    $C_{rh} \leftarrow G^{m_{rh}} H^{r_1} \bmod N_{Acc}$
12:    $r_2 \leftarrow_R \{0,1\}^{\ell_N + \ell_\varnothing}$
13:    $r_3 \leftarrow_R \{0,1\}^{\ell_N + \ell_\varnothing}$
14:    $\beta \leftarrow m_{rh} r_2$
15:    $\delta \leftarrow m_{rh} r_3$
16:    $C_w \leftarrow w H^{r_2} \bmod N_{Acc}$
17:    $C_r \leftarrow G^{r_2} H^{r_3} \bmod N_{Acc}$

18:    $\sigma_{CredStatusUpdate} \leftarrow SPK\{(m_{rh}, w, r_1, r_2, r_3, \beta, \delta):$
         $C_{rh} \equiv G^{m_{rh}} H^{r} \pmod{N_{Acc}}$
         $\bigwedge m_{rh} \in \{0,1\}^{\ell_{rh} + \ell_\varnothing + \ell_c + 1}$
         $\bigwedge C_r \equiv G^{r_2} H^{r_3} \pmod{N_{Acc}}$
         $\bigwedge Acc \equiv C_w^{m_{rh}} (\frac{1}{G})^{\beta} \pmod{N_{Acc}}$
         $\bigwedge 1 \equiv C_r^{m_{rh}} (\frac{1}{G})^{\beta} (\frac{1}{H})^{\delta} \pmod{N_{Acc}}$
         $\}(n_{rrt})$

19:    return $rt \leftarrow (C_w, C_r, \sigma_{CredStatusUpdate})$
20: end function
```

Algorithm 22 Credential Status Verify

```
1: function CREDSTATUSVERIFY($pk_{Iss}, RI, C_{rh}, rt, n_{rrt}$)
2:     $(C_w, C_r, \sigma_{CredStatusUpdate}) \leftarrow rt$
3:     return verify $\sigma_{CredStatusUpdate}$                ▷ See Appendix A of [14]
4: end function
```

4.4. Security analysis

4.4.1. Correctness

The PABC-FLC design satisfies correctness, which can be seen from the following observations. The system parameter generation $SPGen$ algorithm and other key generation algorithms $IssKeyGen$, $NRPKeyGen$, $UserKeyGen$ do not abort. Credential users can run Protocol 14 with the credential issuer to obtain a credential. The protocol does not abort if both parties behave honestly. The $CredPresentation$ algorithm can be used to compute a presentation token and does not abort if the issuer key is generated correctly and the credential is obtained through the credential issuance protocol. Following the completeness of the $SPK_{CredIssuance}$, the resulting presentation token always returns true when verified using $VerifyCredPresentation$ and if the same pk_{Iss}, Att_r, n_{pres} values are used. The credential issuer, having knowledge of pk_{Iss}, can revoke a revocation handle, generated during the credential issuance protocol, using the $CredRevoke$ algorithm. The witness w for a previously unrevoked credential $cred$ and revocation information RI_i can be used to compute a witness for RI_{i+1} using the $RevWitnessUpdate$ algorithm, provided that the revocation handle in the credential is not the one that was revoked. The $CredStatusUpdate$ algorithm always computes a revocation token if the ciphertext ct from the presentation token was computed correctly and the revocation handle is contained within the most recent version of the accumulator (i.e. the credential is unrevoked). Note that an honest credential user sends a witness for the membership relation to the NRP after the credential issuance protocol. Finally, the verification algorithm $CredStatusVerify$ alway returns true for revocation tokens computed using $CredStatusUpdate$, provided the same pk_{Iss}, RI, rrt and nonce n_{rrt} are used.

4.4.2. Credential unforgeability

The unforgeability of the Camenisch-Lysyanskaya signatures relies on the Strong RSA assumption, as proven in [12].

4.4.3. Presentation token unforgeability

Theorem 1. *The PABC-FLC credential system has presentation token unforgeability under the Strong RSA assumption in the random oracle model.*

Proof (sketch). In this proof sketch we argue that any PPT adversary, without knowledge of the issuer secret key, has negligible probability to forge presentation tokens. Consider the $IssKeyGen$ algorithm. Per the statistical zero-knowledge property of the $PK_{IssKeyGen}$ protocol [44], this signature can always be simulated and gives the adversary no additional information about the secret keys of the issuer. Consider the credential issuance protocol. A user proves knowledge of their user secret m_{us} using SPK_{P_1}, if this fails, the protocol aborts. The adversary has only negligible probability in ℓ_n, ℓ_c to produce a valid SPK_{P_1}, assuming the Strong RSA assumption holds, in the random oracle model and provided that the prover (user) is not aware of the factorization of N_{Sig}. The user secret is encoded in a credential by the issuer, along with a revocation handle and the other agreed upon attributes. The rest of the protocol does not abort if the issuer is honest and end with the user receiving a signed credential. Consider the $CredPresentation$ algorithm. Any adversary that wants to forge a presentation token is required to provide a valid $SPK_{CredPresentation}$. The SPK proves that the prover knows a credential with the specified attributes, the credential is signed by the issuer and that the commitment C_{rh} contains the value m_{rh} from the credential. Note that these attributes also include the user secret attribute. As we assume that users do not share their user secret with others, this proves that the presented credential was issued to the same user that computes the presentation token. The adversary has only negligible probability in ℓ_n, ℓ_c to produce a valid $SPK_{CredPresentation}$, assuming the Strong RSA assumption holds, in the random oracle model and provided that the prover (user) is not aware of the factorizations of N_{Sig} and N_{SAcc}. Since the signature scheme is unforgeable under the Strong RSA assumption, the adversary has a negligible probability to produce a valid $SPK_{CredPresentation}$, except for when it has a credential signed by the issuer, the credential contains the user secret of the user, the credential contains the revealed attributes and the commitment contains the revocation handle m_{rh}. □

4.4.4. Revocation token unforgeability

Theorem 2. *The PABC-FLC credential system has revocation token unforgeability under the Strong RSA assumption in the random oracle model.*

Proof (sketch). Like the unforgeability of presentation tokens, we argue that any PPT, without knowledge of the issuer secret key, has negligible probability to forge presentation tokens. Recall that for unforgeability, the issuer is trusted but the NRP is not. Also recall that the NRP has a list of revocation witnesses and keeps these up-to-date when the issuer publishs a new version of the RI. Consider the redeemable revocation token $rrt = (C_{rh}, ct)$. The verification algorithm uses C_{rh} to verify $SPK_{CredStatusUpdate}$. This ensures that the commitment used by the NRP equals that of the rrt. The algorithm aborts if this is not the case. The binding property of the commitment, which holds under the integer factorization assumption [27], means the committed value is the revocation handle from the presented credential. Consider $SPK_{CredStatusUpdate}$. The SPK proves that the value committed to in C_{rh} is contained in the accumulator of the given RI [14]. A prover (the NRP) can cheat the protocol with negligible probability in ℓ_n, ℓ_c, assuming the Strong RSA assumption holds, in the random oracle model and provided that the prover (NRP) is not aware of the factorization of N_{Acc}. Soundness of the CL-RSA-B accumulator [3] states that computing a (new) witness for the proof-of-membership relation is not possible when the revocation handle gets revoked in a (new) version of the accumulator. This means that if the issuer revokes a credential, the NRP is no longer able to prove membership in the accumulator. If the NRP does not know a witness for the revocation handle, it aborts the protocol. We conclude that the adversary has negligle chance in ℓ_n, ℓ_c to forge revocation tokens, assuming the hardness of the Strong RSA assumption (see Section A.1.2) and in the random oracle model [6]. □

4.4.5. Presentation token privacy

Theorem 3. *The PABC-FLC credential system has presentation token privacy in the random oracle model under the RSA assumption.*

Proof. We show that a PPT adversary is unable to make a distinction between two presentation tokens pt, pt'. Recall a presentation token as $pt = (A', \{m_i\}_{i \in Att_r}, \sigma_{pres}, rrt)$. Per definition, the set of revealed attributes is empty $Att_r = \emptyset$.

First, we focus on the redeemable revocation token $rtt = (C_{rh}, ct)$. The encryption algorithm Enc (RSA OAEP) provides ciphertext indistinguishability under chosen plain text attacks in the random oracle model under the RSA assumption. As the adversary does not know sk_{NRP}, the ciphertexts ct, ct' in the two tokens are indistinguishable from eachother. Therefore, ct cannot be used to distinguish between the (adversarially generated) values m_{rh}, m'_{rh}. C_{rh} and C'_{rh} are both statistically hiding commitments when $G \in \langle H \rangle$ [27]. This is assured during the generation of the presentation tokens, as any algorithm using the (adversarially generated) issuer public key pk_{Iss} implicitly verifies $\sigma_{IssKeyGen}$. The (possibly malicious) issuer has negligible probability to produce a forged signature $\sigma_{IssKeyGen}$ if ℓ_c is chosen large enough.

We continue with $A' \leftarrow AS^{r_A} \bmod N_{Sig}$. Although A is part of the credential provided by the adversary, it is verified that the credential is valid and therefore that $A \in \langle S \rangle$. With $r_A \in \{0,1\}^{\ell_N + \ell_\emptyset}$, S^{r_A} is taken statistically indistinguisable from the order of S. A' is therefore also indistinguishable from any other element of $\langle S \rangle$, like all other (valid) credentials.

We continue with σ_{pres}. Aside from A', C_{rh}, the other public input values to the SPK come from the same pk_{Iss}. The message that is signed in the SPK variant is per definition the same. Completeness and (honest-verifier) statisical zero-knowledge of the $PK_{CredPresentation}$ protocol is easy to see, provided that the prover is convinced that $Z, S, \{R_i\}_{i \in Att} \in \langle S \rangle$. As described earlier, the adversary is able to convince the prover that this is the case only with negligible probability in ℓ_c.

Given that the two presentation token(s) are indistinguishable from other (valid) presentation tokens, the best guess of the adversary on which credential was used, is now limited to a random guess of $\frac{1}{2}$, plus a negligible function in terms of the security parameter(s). □

4.4.6. Revocation token privacy

Theorem 4. *The PABC-FLC credential system has revocation token privacy in the random oracle model under the RSA assumption.*

Proof. This proof follows that of presentation token privacy by showing that the distributions of two (valid) revocation tokens are indistinguishable from eachother. Recall revocation tokens as $rt = (C_w, C_r, \sigma_{CredStatusUpdate})$. Also recall that the adversary knows $cred_0, cred_1, RI$ as it generated these. Like before, the set of revealed attributes is empty, pk_{Iss}, pk_{NRP}, RI are the same, and the nonces are equal per definition. Per Theorem 3, the distribution of pt_0, pt_1 is equal for all (valid) credentials in the random oracle model and assuming the RSA assumption. Both credentials are clearly unrevoked, because $rt_0 \neq \perp, rt_1 \neq \perp$.

If we focus on C_w, C_r, both variables are statistically hiding commitments. As for C_{rh}, part of the presentation token, the prover is assured that the commitments are hiding if $\sigma_{IssKeyGen}$ verifies. The adversary has a negligible probability in ℓ_c to cheat this SPK since $pt_0 \neq \perp, pt_1 \neq \perp$.

Moving on to $\sigma_{CredStatusUpdate}$, indistinguishability of this variable depends on the (honest-verifier) statistical zero-knowledge of $PK_{CredStatusUpdate}$. This protocol follows that of [14],

Mechanism	**FLC**	**Unlinkable**	**Untraceable**	**RI (size/freq)**	**Online**
CRL [37]	✓	✗	✗	R / R	V
OCSP [38]	✓	✗	✗	✗	I, V
VE [16]	✓	✓	✗	✗	I, V
N-times [10]	✗	✓	✗	R × U / E	V
Braavos [3]	✗	✓	✓	1 / R	U, V
ADNMP [1]	✓	✓	✗	1 / R	U, V
PABC-FLC	✓	✓	✓	1 / R	NRP, V

Table 4.1: Comparison of the PABC-FLC design with related work.

but differs in that we do not have an additional commitment to m_{rh}. Showing statistical zero-knowledge follows the standard technique, see Section 2.2 for more details. □

4.4.7. A comparison with related work

We give a brief comparison between the presented PABC-FLC design and related work, as described in Section 3. Remember first of all that we introduced a new role in the credential system, the NRP. We assume that the NRP does not collude with either the issuer or any verifier, as doing so breaks the *multi-show unlinkability* and *untraceability* properties (both captured in the privacy definitions). We also require that the NRP can be contacted at any time to produce a new credential status update. At the same time, the user is no longer required to keep track of the revocation witness and can remain offline after credential issuance. The RI size and update frequency are equal to the Braavos construction, which is also used in the PABC-FLC credential system. Given that a verifier can contact the NRP for a new revocation token, proving that a credential is not revoked in a newer version of the RI, our design also satisfies the forward-looking consistency requirement. A full comparison is given in Table 4.1.

5
Performance evaluation

5.1. Introduction

As part of this research, we also take a look at the experimental runtime of the PABC-FLC credential system. We do so by comparing the PABC-FLC design, which provides forward-looking consistency, against a regular credential system. We consider our PABC-FLC design *without* revocation as regular credential system. This comparison gives a clear indication of the overhead incurred from the revocation mechanism with forward-looking consistency.

The proof-of-concept implementation used in the experiments is written in Go 1.14 [50] and makes use of the standard Go big integer implementation. Our implementation uses the safe prime generation algorithm of [48], but otherwise matches the algorithm and protocol descriptions of this work. (Unfortunately, we were unable to use other parts of the research library described in [48], as this required significant changes to the implementation. Nonetheless, we do appreciate the effort of the authors to facilitate further research on privacy-preserving cryptography.) The system parameters are based on those of *idemix* [49], with the addition of $\ell_{enc} = 3072$. The runtime measurements were performed on a laptop with an Intel Core i7-7560U @ 2.40GHz CPU and 16 GB of RAM, using the standard Go benchmarking functions.

We compare the runtime of the two credential systems for following algorithms and protocols: issuer key generation, credential issuance, creating presentation tokens and verifying presentation tokens. We also measure the runtime of creating and verifying revocation tokens, which do not exist in a standard credential system.

5.2. Issuer key generation

Runtime measurements of the $IssKeyGen$ algorithm indicate the (one time) cost for the issuer. We measure the average runtime of 10 executions and consider both our credential system and a standard credential system without revocation mechanism. As the number of attributes in a credential can influence the runtime, we considered two scenarios. One without attributes in the credential, the other with 20 additional attributes. Note that these numbers does not include the user secret and revocation handle attributes. The results are given in Table 5.1. Recall that the issuer needs to find suitable primes for the RSA modulus, which is expected to contibute to a high runtime. The addition of the revocation mechanism is expected to double this, due to the generation of the additional RSA modulus for the revocation mechanism. The measurements clearly show that adding a revocation mechanism adds to the setup cost for the issuer, which is at least doubled in runtime in both cases. At the same time there is a decrease in overhead percentage in the credential with 20 attributes, which indicates a smaller but linear cost of additional attributes. We consider the increase in runtime reasonable, as the issuer is expected to run the algorithm only once and before it begins to issue credentials.

Table 5.1: Runtime measurements issuer key generation

Attributes	Regular credential system	PABC-FLC	Difference
0	15.6 sec	34.7 sec	+19.1 sec (+122%)
20	17.5 sec	36.0 sec	+18,5 sec (+106%)

5.3. Credential issuance

To measure the runtime of credential issuance, we compare our credential system with a regular credential system. Additionally, we vary the number of attributes from 0 to 20 in steps of 5. The measurements are the average runtime of 50 executions of the credential issuance protocol. The results are given in Table 5.2. Recall that the issuance protocol includes prime number generation for both the credential signature and, if the revocation mechanism is used, the revocation handle. The results indicate no clear difference between the two credential systems, which is considerd surprising given the additional work on both the issuer and user side of the protocol. This may be explained by the work already required in a regular credential system, for example signing the credential. We conclude that our design does not influence the runtime of credential issuance compared to a regular credential system.

Table 5.2: Runtime measurements credential issuance

Attributes	Regular credential system	PABC-FLC	Difference
0	448 ms	405 ms	-43 ms (-10%)
5	413 ms	537 ms	+124 ms (+30%)
10	539 ms	511 ms	-28 ms (-5%)
15	640 ms	643 ms	+3 ms (0%)
20	640 ms	672 ms	+32 ms (+5%)

5.4. Presentation token generation and verification

When measuring the runtime of the $CredPresentation$ algorithm in the two credential systems, we also vary the number of credential attributes from 0 to 20. The measurements were obtained by averaging the runtime of 50 CredPresentation algorithm executions and are given in Table 5.3. During the presentation, we do not reveal any attributes. Since revealing an attribute is faster than not revealing it, the scenario of not revealing any attribute exemplifies the worst case scenario. The results indicate that both the number of attributes and the revocation mechanism have a clear influence on the runtime. Like the original *idemix* credential system, the runtime increases with the number of attributes in the credential. Our PABC-FLC credential system additionally adds 20 ms to each presentation. This is a 32% increase for credentials without (additional) attributes, but this overhead percentage decreases for credentials with more attributes.

Table 5.3: Runtime measurements credential presentation

Attributes	**Regular credential system**	**PABC-FLC**	**Difference**
0	59 ms	78 ms	+19 ms (+32%)
5	72 ms	92 ms	+20 ms (+28%)
10	85 ms	105 ms	+20 ms (+24%)
15	99 ms	119 ms	+20 ms (+20%)
20	111 ms	134 ms	+23 ms (+21%)

Presentation tokens are verified using the $VerifyCredPresentation$ algorithm. We consider the same scenarios as before and take the average runtime of 50 runs of the algorithm. The results are given in Table 5.4. Similar to the previous results on generating presentation tokens, we see that verifying a token takes about the same time. Also similar, is that the PABC-FLC design adds about 20 ms to the runtime, regardless of the number of attributes in a credential. Compared to additional runtime of more credential attributes, which is 2,75 ms per attribute, the cost of adding a revocation mechanism is similar to the addition of 7-8 credential attributes. We consider this a reasonable increase for both algorithms.

Table 5.4: Runtime measurements credential presentation verification

Attributes	**Regular credential system**	**PABC-FLC**	**Difference**
0	58 ms	78 ms	+20 ms (+34%)
5	73 ms	91 ms	+18 ms (+25%)
10	86 ms	104 ms	+18 ms (+21%)
15	100 ms	119 ms	+20 ms (+20%)
20	112 ms	132 ms	+20 ms (+18%)

5.5. Additional algorithms

The PABC-FLC credential system adds several algorithms not used in a regular credential system. , we make no comparison to a regular credential system, but do measure their runtime.

The $CredStatusUpdate$ algorithm used by the NRP has an average runtime of 85 ms (50 runs). Verifying revocation tokens, using the $VerifyCredStatusUpdate$ algorithm, takes on average 55 ms (50 runs). Both of these are considered reasonable compared with, for example, verifying a presentation token (78+ ms).

6 Discussion and future work

6.1. Discussion

This thesis started out by asking how a PABCS can be used in situations that require forward-looking consistency. This work, especially the PABC-FLC design, provides new insights to answer previously unanswered research questions. In this discussion, we return to the research questions formulated in the introduction. We revisit each question and give an answer based on the results of this work. But before answering these research questions, we first discuss the results and limitations of this work.

This work presents the PABC-FLC design. To the best of our knowledge, this is the first Privacy-preserving Attribute-Based Credential System (PABCS) to provide forward-looking consistency. The PABC-FLC design shows that PABCSs *can* be used in situations that require this strong consistency level.

We also describe why existing revocation mechansisms are unable to provide both forward-looking consistency and privacy. This is because we assume a possible collusion between the issuer and verifier(s), and because we assume that users remain offline after a credential presentation. In this setting, where the issuer also acts as the Revocation Authority (RA), no solution can satisfy all of these requirements (see Section 4.2.7).

The PABC-FLC design (see Chapter 4) avoids this impossibility by introducing a new role in the credential system, the NRP. This new participant does not affect the unforgeability of credentials or credential presentations, meaning that neither the issuer nor the verifiers need to trust the NRP to behave honestly. Credential users however, do need to trust the NRP as otherwise their privacy is not guaranteed. This raises the question of who would be trusted by all users. Considering that most users already trust a software implementation of the credential system to work as intended, one suggestion is to have the organization developing this software run the NRP. In the future work section, we elaborate on other approaches. For example by having multiple NRPs in the system or by reconsidering the assumption that users remain offline.

The introduction of the NRP is argueably the most influential design decision that was made. We will briefly describe the other decisions made in the PABC-FLC design. We build on the *idemix* credential system as this is one of the few that supports multi-show credentials. Furthermore, quite a few revocation mechanisms were designed specifically to work with this credential system. One of these revocation mechanisms is the Braavos accumulator, a state-of-the-art accumulator that achieves the lowerbound of required witness updates. Both *idemix* and the Braavos accumulator do not require a trusted setup, outside the verifier trusting the issuer, which sets them apart from other credential systems and revocation mechanisms. The choice to use RSA-OAEP as the encryption algorithm stems from the RSA assumption the scheme requires. This is a weaker assumption compared to the Strong RSA assumption of the *idemix* credential system, meaning the unforgeability of the system does not require any additional hardness assumption(s).

The security assumptions mentioned in the previous paragraph also play a role in the security definitions and analysis. Previous work, especially [21], provide formal definitions on the unforgeability and privacy of a PABCS. Unfortunately, the introduction of the NRP prevents us from using the definitions and results of this work. Additionally, we wanted to avoid the trusted setup that is assumed in this work. Given the limited time and scope of this thesis, we decided to use less formal definitions than those specified in earlier work. Likewise, we only provide a proof sketch on the unforgeability definitions.

In the performance evaluation (see Chapter 5), we compare our proof-of-concept implementation against the same implementation *without* revocation mechanism. We consider this a fair comparison as our goal is to show that the PABC-FLC design can be used instead of a credential system that does not provide forward-looking consistency. A comparison of our proof-of-concept implementation, which lacks performance optimizations, against other implementations would also prevent us from reliably computing the additional overhead incurred. Furthermore, performance results from related work cannot be compared directly with ours as the environment (smartcard, laptop, server) of the experiments differs.

After discussing the results and limitations of this work, we return to the research questions we set out to answer.

RQ1 How can a revocation mechanism be used to determine the revocation status of an earlier presented credential?

To answer this question, we conducted a literature survey on existing revocation mechanisms (see related work, Section 3). Existing mechanisms implicitly assume that the verifier determines the revocation status of a credential when one is presented to them. In this work, we consider situations in which the verifier needs to determine the revocation status at a moment *after* the credential is presented. Since the revocation status of a credential can change over time, from unrevoked to revoked, the revocation status at this later moment can differ from the revocation status at presentation time.

Although the existing revocation mechanisms can be used to determine the revocation status of a credential at presentation time, not all of them can be used after that moment. Specifically the Braavos accumulator and “N-times unlinkable proofs” mechanisms cannot be used this way, because a verifier can only determine if a credential is revoked in one specific version of the Revocation Information (RI). The other revocation mechanisms allow the verifier to either contact the issuer for the revocation status of a credential (OCSP, VE), or provide the verifier with information that can be used to check the revocation in newer versions of the RI (CRL, ADNMP). We conclude that a revocation mechanism must do either of these, if it wants to provide forward-looking consistency. The PABC-FLC design allows the verifier to contact the NRP, which returns the revocation status of a credential in a verifiable manner.

RQ2 How do revocation mechanisms remain privacy-preserving in the context of multi-show credentials?

Multi-show credentials are credentials that can be presented multiple times. We consider these credentials privacy-preserving if they are *multi-show unlinkable*. *Unlinkable* means that a verifier is unable to link a credential presentation to one of two credentials. *Multi-show unlinkable* means that it is not possible to determine if two different presentations were generated by the same credential. Regarding revocation mechanisms, this means that any information revealed during a credential presentation must be *multi-show unlinkable*. This immediately excludes any mechanism that reveals a single unique identifier to the verifier.

The revocation mechanisms that are *multi-show unlinkable*, take one of the follow approaches: (1) The credential presentation contains a ciphertext of the RH and requires another participant to decrypt the ciphtertext and return the revocation status of the credential. This is how Verifiable Encryption (VE) works. (2) The credential presentation contains a pseudonym derived from the RH. This is how the "N-times unlinkable proofs" mechanism works. (3) The credential presentation contains a proof-of-knowledge that the RH is (not) contained in the set of (un)revoked handles. This is how the accumulator mechanisms work. The PABC-FLC design combines the first and last approach. A redeemable revocation token given to the verifier, as part of the credential presentation, consists of a ciphertext and statistically-hiding commitment. The revocation token, returned to the verifier by the NRP, is a (zero-knowledge) Signature Proof-of-Knowledge (SPK) that only reveals that a credential is revoked. Neither the redeemable revocation tokens nor the revocation tokens can be linked to a specific credential by a verifier (or issuer).

RQ3 How does a revocation mechanism remain untraceable when the issuer and verifier collude?

The untraceability of the revocation mechanisms means that the issuer cannot link a credential presentation to a specific credential (issuance moment). Since we are interested in a revocation mechanism that works for all verifiers (global) and allows the issuer to revoke a credential ("issuer-driven"), this requires the use of a RH that allows the issuer to identify the credential it wants to revoke. As a consequence, credential presentations should not contain any information that can be linked to this RH. This excludes mechanisms that reveal the RH directly (CRL, OCSP) or indirectly in a way that still allows the issuer to link it to the RH (VE, N-Times).

The cryptographic accumulators are untraceable, since they use a zero-knowledge proof-of-membership that only proves the RH is not revoked. The homomorphic ADNMP accumulator also provides the ability to delegate the computation of the non-membership proof, but as the authors mention, this conflicts with the linkability (and untraceability). We conclude that the best way to satisfy untraceability is by using a (zero-knowledge) proof-of-membership that proves the credential is not revoked. For this reason, the PABC-FLC design uses the Braavos accumulator and associated Signature Proof-of-Knowledge (SPK) algorithms.

However, we should note that we assume that the issuer also acts as the Revocation Authority (RA) for the credentials it issues. If these roles were separated *and* if one assumes that these participants do not collude, we expect that a number of the existing mechanisms can also provide untraceability. At the same time however, a system that separates these roles needs to ensure that if the issuer wants to revoke a credential, the RA actually revokes this credential. Our PABC-FLC design does not have this issue, as the issuer itself publishes the RI.

RQ4 How can a revocation mechanism that provides forward-looking consistency be used with an existing credential system?

In Chapter 4, we present the PABC-FLC design, which shows one approach on how a revocation mechanism can be combined with an existing credential system. We explain our choice for *idemix* (as existing credential system) and the Braavos accumulator (part of the revocation mechanism) at the beginning of this section. Here we focus only on how these were combined. To clarify, we answer this question by explaining the approach used in the PABC-FLC design. We do not claim that this is the only approach.

The *idemix* crededential system is built around the Camenisch-Lysyanskaya signature scheme (see Section 2.4). This signature scheme works in a special RSA modulus group and can be used in a (zero-knowledge) Signature Proof-of-Knowledge (SPK) (see Section 2.2) to prove knowledge of message and signature. This is how a credential is presented. The credential presentation is combined with a revocation mechanism by creating a commitment to one of the signed messages, the RH. In the case of the Braavos accumulator, this commitment is then used in a proof-of-membership to convince the verifier that the committed value is contained in the accumulator. Because both the Braavos accumulator and Damgård-Fujisaki commitments work in a special RSA modulus group like the Camenisch-Lysyanskaya signature scheme, their different SPKs can be combined without additional cost.

As discussed in the related work chapter, the Braavos accumulator by itself does not provide any forward-looking consistency. For this reason we introduced the NRP. During the credential presentation we not only create a commitment to the RH, but also encrypt the RH and commitment opening under the public key of the NRP. The NRP can in turn compute a proof-of-membership, based on the RH and revocation witness received earlier, to convince the verifier that the RH is contained in the latest version of the accumulator. In other words, to prove that the credential is not revoked.

To recap, the PABC-FLC design builds on the *idemix* credential system. We combine *idemix* with a revocation mechanism by creating a commitment to the RH during a credential presentation. The commitment is tied to the credential by combining the SPKs of both.

RQ5 How does the runtime of a PABCS with forward-looking consistency compare against a regular credential system?

In Chapter 5, we give the results of an experimental performance evaluation of a proof-of-concept implementation of the PABC-FLC design. This evaluation shows that the runtime of the issuer setup is more than doubled (+112%), compared to a credential without revocation mechanism. It also shows that computing a credential presentation, and verification thereof, is increased by 20 ms. Percentagewise the overhead decreases from +32%, for a credential without attributes, to +21% for a credential with 20 attributes. The time for the NRP to compute a new credential status update, a revocation token, is 85 ms. It takes the verifier 55 ms to verify the resulting revocation token. Given that the setup is only ran once, we conclude that the PABC-FLC design has a reasonable overhead compared to a regular credential system.

At the same time, we should note that our proof-of-concept implementation is not optimized. Any optimization can reduce both the runtime of the revocation mechanism or the credential system itself, which may result in both an increase or decrease of the overhead as a percentage. The absolute runtime is reduced in either case.

How to design a privacy-preserving revocation mechanism for determining the revocation status after presentation in Privacy-preserving Attribute-Based Credential systems?

Considering the answers we formulated to the subquestions, we now return to the main research question. In this work, we researched for the first time how a Privacy-preserving Attribute-Based Credential System (PABCS) can provide forward-looking consistency. We describe the requirements that a revocation mechanism has to fulfill and how existing mechanisms do not satisfy these requirements. We also show that without the introduction of the NRP role, no design can satisfy all of our requirements. Given the introduction of this NRP, the PABC-FLC design shows one approach to design a revocation mechanism with forward-looking consistency and how this is used with an existing credential system. In the PABC-FLC design, a verifier can determine the revocation status of a credential at any moment after the credential was presented. This provides the verifier the ability to enforce forward-looking consistency, without compromise on the privacy aspects of the credential system.

6.2. Future work

As discussed in the previous section, this work has several limitations and poses new questions regarding forward-looking consistency in PABCSs. Here we suggest three directions for future research. In the first suggestion, we describe how the PABC-FLC design can be extended to use multiple NRPs. We expect that this can reduce the trust users need to have in a single NRP. Another direction relates to the business side of using the PABC-FLC design and needs to clarify *if* and *when* forward-looking consistency is required in a situation. The final suggestion we have for researchers interested in this, is to first conduct a case study to clarify the exact requirements in a specific situation before using our (or any) PABCS. We expect that if a situation does not require all the requirements we set for the PABC-FLC design, many other solutions can be designed as well and that these should be considered too.

Extending our design with multiple NRPs The PABC-FLC design assumes that there is only one Non-Revocation Prover (NRP). Credential users must trust the NRP to not collude with other participants, otherwise their privacy is not guaranteed. The design can be extended by allowing multiple NRPs in the system. This allows users to choose a NRP they trust or even act as the NRP themselves. As a reminder, neither the verifier nor the issuer is required to trust the NRP and the unforgeability of the system is guaranteed even if the NRP behaves malicious.

However, the implication of multiple NRPs is that verifiers need to know *which* NRP to contact. This information can be provided along with the redeemable revocation token, part of the presentation token, but it should be clear that the verifier now learns more information than it would otherwise have. This information can in fact be used to distinguish between two users and/or credentials that use different NRPs, violating the unlinkability definition.

When is forward-looking consistency required Another direction for research is to design a framework that can be used to analyse situations to determine if forward-looking consistency is required. And if so, what would be the appropriate moment for a verifier to check the revocation status of a credential. Although we give several example scenarios, we do not take into account the benefits for a verifier that requires forward-looking consistency. It should be clear that the number of times a verifier needs to contact the NRP is limited to the number of credentials that are revoked, i.e. the number of RI updates. But contacting the NRP that often, may not be appropriate. A framework that takes into account the goals of the verifier, decision moments where forward-looking consistency can make a difference, and the benefits of doing so, can help a verifier to determine if and when it should contact the NRP.

Reconsideration of our requirements This thesis assumes that multi-show credentials are useful, issuer(s) and verifiers collude, credentials require a revocation mechanism, the issuer always acts as the Revocation Authority (RA) and that a user cannot be contacted by a verifier after a credential presentation. These assumptions influence the requirements we set for the PABC-FLC system and this work in general. Although this allows our design to be used in a wide variety of situations, not all situations have the same requirements. We expect that if any of these requirements is reconsiderd, other approaches to provide forward-looking consistency in a PABCS become possible. If for example the credential verifier can contact the user at any point in time, it can also request a new credential presentation. If the credential presentation is combined with a 'regular' revocation mechanism, this can replace the function of the NRP in the credential system. Therefore, we recommend that researchers first conduct a case study to determine if all requirements of this work are also required. If that is not the case, we suggest they explore other designs. But if that *is* the case, then we recommend they consider using our PABC-FLC design.

A
Appendix

A.1. Computational assumptions

A.1.1. RSA assumption

Definition 4 (RSA problem [45])**.** Given is (N, Y, e) with N the composite of two prime numbers P, Q and e prime to $\phi(n)$. Output (X) such that $Y \equiv X^e \pmod{N}$.

A.1.2. Strong RSA assumption

The strong RSA problem is a generalization of the well-known RSA problem that allows the adversary to choose e. The assumption states that it is hard for any Probabilistic Polynomial-Time (PPT) machine to solve the strong RSA problem with more than negligible probability in the length of N.

Definition 5 (Strong RSA problem [30])**.** Given is (N, Y) with N the composite of two prime numbers P, Q. Output (X, e) with $e \geq 2$ such that $Y \equiv X^e \pmod{N}$.

A.2. Overview symbols

Symbol	**Explanation**
$spar$	System parameters, generated by $SPGen$.
ℓ_N	Length of the (special) RSA modulus.
ℓ_m	(Maximum) length of attribute values.
ℓ_{Att}	Number of attributes in a credential, including the user secret and revocation handle.
ℓ_e	Beginning of the e values interval. $\ell_e > \ell_m + 1$.
$\ell_{e'}$	Length of the interval of e values.
$\ell_\emptyset$	Length of the statistical zero-knowledge security parameter.
ℓ_c	Length of challenges in the SPK protocols. ℓ_c must be smaller than the order of the group.
ℓ_v	Length of v as used in the Camenish-Lysyanskaya signature scheme. $\ell_v = \ell_n + \ell_m + \ell_c$.
ℓ_{rh}	Length of the revocation handle(s). $\ell_{rh} \leq \ell_m$.
Att_k	Set of credential attributes known to both user and issuer.
$Att = Att_k \cup \{us, rh\}$	Set of all credential attributes.
Att_r	Set of credential attributes that is revealed during credential presentation. $Att_r \subset Att_k$.
R_i	Base value of attribute i, as used in the Camenisch-Lysyanskaya signature scheme.
m_i	Attribute value of attribute i.
m_{us}	User Secret attribute value.
m_{rh}	Revocation Handle attribute value.
RI_i	Revocation Information at version i. The version starts at 0, when the accumulator is generated, and increases monotonically with each version.
Acc	Accumulator value, part of the revocation information.
N_{Sig}, N_{Acc}	(Special) RSA modulus of respectively the Camenisch-Lysyanskaya signature scheme and CL-RSA-B accumulator.

Table A.1: Overview of symbols and their usage.

List of Algorithms

Abbreviations

ABC Attribute-Based Credential (ABC) A credential that allows a user to authenticate based on the attributes in the credential.

CRL Certificate Revocation List (CRL) A list of revoked credential serial numbers (revocation handles), used in the WebPKI.

NRP Non-Revocation Prover (NRP) The (additional) participant responsible for generating revocation tokens given a redeemable revocation token.

OCSP Online Certificate Status Protocol (OCSP) A protocol to retrieve the revocation status of a X.509 certificate from the issuing Certificate Authority, as used in the WebPKI.

PABCS Privacy-preserving Attribute-Based Credential System (PABCS) A privacy-preserving Attribute-Based Credential system. Credentials in this system allow the selective disclosure of individual attributes. Credential presentations are unlinkable and untraceable, even if a credential is presented multiple times.

PPT Probabilistic Polynomial-Time (PPT) The complexity class of algorithms that run in Probabilistic Polynomial Time with regards to the input size.

RA Revocation Authority (RA) The participant responsible for revoking a credential, identified by a Revocation Handle.

Revocation Witness Revocation Witness (Revocation Witness) The additional input required to compute revocation tokens for a Revocation Handle.

RH Revocation Handle (RH) A unique identifier, encoded as an attribute in a credential, that is used to revoke a credential.

RI Revocation Information (RI) The (public) information published by the Revocation Authority, that allows verifiers and/or users to check the revocation status of a specific Revocation Handle. It also contains information for users that allows them to update the Revocation Witness, if the revocation mechanism requires this.

SPK Signature Proof-of-Knowledge (SPK) See Section 2.2.

Bibliography

[1] Tolga Acar and Lan Nguyen. “Revocation for delegatable anonymous credentials”. In: *International Workshop on Public Key Cryptography*. Springer. 2011, pp. 423–440.

[2] Man Ho Au, Willy Susilo, and Yi Mu. “Constant-size dynamic k-TAA”. In: *International Conference on Security and Cryptography for Networks*. Springer. 2006, pp. 111–125.

[3] Foteini Baldimtsi et al. “Accumulators with applications to anonymity-preserving revocation”. In: *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2017, pp. 301–315.

[4] Endre Bangerter. “Efficient zero knowledge proofs of knowledge for homomorphisms.” In: (2005).

[5] M. Bellare. “Optimal Asymmetric Encryption-How to Encrypt with RSA”. In: *Eurocrypt’94*. 1995.

[6] Mihir Bellare and Phillip Rogaway. “Random oracles are practical: a paradigm for designing efficient protocols”. In: *Proceedings of the 1st ACM conference on Computer and communications security*. 1993, pp. 62–73.

[7] David Bernhard, Olivier Pereira, and Bogdan Warinschi. *How not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios*. Cryptology ePrint Archive, Report 2016/771. `https://eprint.iacr.org/2016/771`. 2016.

[8] Stefan Brands. *Rethinking public key infrastructures and digital certificates: building in privacy*. Mit Press, 2000.

[9] Philippe Camacho. *On the Impossibility of Batch Update for Cryptographic Accumulators*. Cryptology ePrint Archive, Report 2009/612. `https://eprint.iacr.org/2009/612`. 2009.

[10] Jan Camenisch, Manu Drijvers, and Jan Hajny. “Scalable revocation scheme for anonymous credentials based on n-times unlinkable proofs”. In: *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*. ACM. 2016, pp. 123–133.

[11] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. “An accumulator based on bilinear maps and efficient revocation for anonymous credentials”. In: *International Workshop on Public Key Cryptography*. Springer. 2009, pp. 481–500.

[12] Jan Camenisch and Anna Lysyanskaya. “A signature scheme with efficient protocols”. In: *International Conference on Security in Communication Networks*. Springer. 2002, pp. 268–289.

[13] Jan Camenisch and Anna Lysyanskaya. “An efficient system for non-transferable anonymous credentials with optional anonymity revocation”. In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2001, pp. 93–118.

[14] Jan Camenisch and Anna Lysyanskaya. “Dynamic accumulators and application to efficient revocation of anonymous credentials”. In: *Annual International Cryptology Conference*. Springer. 2002, pp. 61–76.

[15] Jan Camenisch and Markus Michels. “Separability and Efficiency for Generic Group Signature Schemes”. In: *CRYPTO ’99 Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*. 1999, pp. 413–430.

[16] Jan Camenisch and Victor Shoup. “Practical verifiable encryption and decryption of discrete logarithms”. In: *Annual International Cryptology Conference*. Springer. 2003, pp. 126–144.

[17] Jan Camenisch and Markus Stadler. “Efficient group signature schemes for large groups”. In: *Annual International Cryptology Conference*. Springer. 1997, pp. 410–424.

[18] Jan Camenisch and Els Van Herreweghen. “Design and implementation of the idemix anonymous credential system”. In: *Proceedings of the 9th ACM conference on Computer and communications security*. ACM. 2002, pp. 21–30.

[19] Jan Camenisch et al. “Concepts and languages for privacy-preserving attribute-based authentication”. In: *IFIP Working Conference on Policies and Research in Identity Management*. Springer. 2013, pp. 34–52.

[20] Jan Camenisch et al. “Fast keyed-verification anonymous credentials on standard smart cards”. In: *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer. 2019, pp. 286–298.

[21] Jan Camenisch et al. *Formal Treatment of Privacy-Enhancing Credential Systems*. Cryptology ePrint Archive, Report 2014/708. `https://eprint.iacr.org/2014/708`. 2014.

[22] Jan Camenisch et al. “Formal Treatment of Privacy-Enhancing Credential Systems”. In: *Revised Selected Papers of the 22nd International Conference on Selected Areas in Cryptography - SAC 2015 - Volume 9566*. Vol. 2014. 2015, pp. 3–24.

[23] David Chaum. “Security without identification: Transaction systems to make big brother obsolete”. In: *Communications of the ACM* 28.10 (1985), pp. 1030–1044.

[24] Geoffroy Couteau, Thomas Peters, and David Pointcheval. *Removing the Strong RSA Assumption from Arguments over the Integers*. Cryptology ePrint Archive, Report 2016/128. `https://eprint.iacr.org/2016/128`. 2016.

[25] Ronald Cramer. “Modular design of secure yet practical cryptographic protocols”. In: *Ph. D. Thesis, CWI and University of Amsterdam* (1996).

[26] Ivan Damgård. *On Σ-protocols*. 2010.

[27] Ivan Damgård and Eiichiro Fujisaki. “A statistically-hiding integer commitment scheme based on groups with hidden order”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2002, pp. 125–142.

[28] Yevgeniy Dodis and Aleksandr Yampolskiy. “A verifiable random function with short proofs and keys”. In: *International Workshop on Public Key Cryptography*. Springer. 2005, pp. 416–431.

[29] Amos Fiat and Adi Shamir. “How to prove yourself: Practical solutions to identification and signature problems”. In: *Conference on the Theory and Application of Cryptographic Techniques*. Springer. 1986, pp. 186–194.

[30] Eiichiro Fujisaki and Tatsuaki Okamoto. “Statistical zero knowledge protocols to prove modular polynomial relations”. In: *Annual International Cryptology Conference*. Springer. 1997, pp. 16–30.

[31] Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. 2001.

[32] Oded Goldreich, Silvio Micali, and Avi Wigderson. “Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems”. In: *Journal of the ACM* 38.3 (1991), pp. 690–728.

[33] S. Goldwasser, S. Micali, and C. Rackoff. “The knowledge complexity of interactive proof systems”. In: *SIAM Journal on Computing* 18.1 (1989), pp. 186–208.

[34] Jens Groth and Amit Sahai. “Efficient non-interactive proof systems for bilinear groups”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2008, pp. 415–432.

[35] Daniel Hardman and Lovesh Harchandani. *Preventing Transferrability with ZKP-based Credentials*. URL: `https://github.com/WebOfTrustInfo/rwot9-prague/blob/master/topics-and-advance-readings/zkp-safety.md` (visited on 08/09/2020).

[36] IETF. *RFC2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. URL: `https://tools.ietf.org/html/rfc2459` (visited on 01/15/2020).

[37] IETF. *RFC5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. URL: `https://tools.ietf.org/html/rfc5280` (visited on 08/06/2019).

[38] IETF. *RFC6960: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. URL: `https://tools.ietf.org/html/rfc6960` (visited on 08/16/2019).

[39] Jorn Lapon et al. “Analysis of revocation strategies for anonymous Idemix credentials”. In: *IFIP International Conference on Communications and Multimedia Security*. Springer. 2011, pp. 3–17.

[40] Jorn Lapon et al. “Performance analysis of accumulator-based revocation mechanisms”. In: *IFIP International Information Security Conference*. Springer. 2010, pp. 289–301.

[41] Lan Nguyen. “Accumulators from bilinear pairings and applications”. In: *Cryptographers’ Track at the RSA Conference*. Springer. 2005, pp. 275–292.

[42] David Pointcheval and Jacques Stern. “Security proofs for signature schemes”. In: *EUROCRYPT’96 Proceedings of the 15th annual international conference on Theory and application of cryptographic techniques*. 1996, pp. 387–398.

[43] Stichting Postfilter. *Nationaal Post Register*. URL: `https://www.postfilter.nl/` (visited on 09/02/2019).

[44] Guillaume Poupard and Jacques Stern. “Security analysis of a practical “on the fly” authentication and signature generation”. In: *theory and application of cryptographic techniques* 1403 (1998), pp. 422–436.

[45] R. L. Rivest, A. Shamir, and L. Adleman. “A method for obtaining digital signatures and public-key cryptosystems”. In: *Communications of The ACM* 21.2 (1978), pp. 120–126.

[46] Claus-Peter Schnorr. “Efficient signature generation by smart cards”. In: *Journal of cryptology* 4.3 (1991), pp. 161–174.

[47] Mehrnoosh Shakarami and Ravi Sandhu. “Safety and consistency of subject attributes for attribute-based pre-authorization systems”. In: *NCS. Springer, Heidelberg* (2019).

[48] Miha Stopar et al. “emmy – Trust-Enhancing Authentication Library”. In: *IFIP International Conference on Trust Management.* 2019, pp. 133–146.

[49] Security team. “Specification of the Identity Mixer Cryptographic Library v2.3.0”. In: (2010).

[50] *The Go Programming Language.* `https://golang.org/`. Accessed: 2020-10-08.