Techniques in Allowing Multi-Show in Digital Credentials

by

Jinnan Fan

Thesis submitted in partial fulfillment of the requirements
For the Master of Applied Science degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Cryptographic credential systems provide some possible solutions to the problem of privacy leakage of users in the "virtual" world. This thesis presents a privacy-preserving method which can enable the cryptographic credentials to have the capability of anonymous multi-show.

Our approach builds on the work of Brands from the year 2000 which proposed a Digital Credential system that can protect users' privacy. This system is efficient but not perfect, since the Digital Credentials in that system can only be shown once to avoid linkability. We propose the use of a malleable signature technique to transform Brands' Digital Credentials from single-show to multi-show capability.

In this thesis, we describe our modified issuing and showing protocols and discuss the security properties of our proposed scheme. We have a basic implementation (proof of concept) to support our concept and analysis of timing results is also provided. In the end, we point out a number of future directions which can be used to complement or improve this approach.

Acknowledgements

First, I would like to thank my amazing supervisor Dr. Carlisle Adams, who led me into this fantastic area. This thesis would not have been finished without his ideas, advice and feedback. Concurrently, thanks to my friends and colleges at University of Ottawa for all their kindnesses and help. Finally, I also want to thank my lovely husband, who always supports me under any condition.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Many current credential schemes suffer from the disadvantage that the transactions of a credential will be linked if this credential has been shown more than once. To date, some researchers have focused on the non-linkability of cryptographic credentials which prevent privacy leakage. Furthermore, some infrastructure providers have products that implement the technologies in this area.

1.1 Motivation

This research is related to the problem of leaking individuals' privacy. The "Internet" seems to know everything about ourselves, when we use online systems. For example, if you have searched or bought some makeup products, you will find that the frequency of the appearance for cosmetic related advertisements would increase during the use of "Internet", which includes your Youtube channels, Facebook pages and even Google results. Somehow, your online activities have been linked together when browsing the internet.

Sometimes, this kind of "linking" even provides convenience and benefits to us in everyday life. However, we should be given the ability to control when, how, and to what extent we want to allow our behavior to be profiled, even for the sake of convenience.

By providing unlinkability of cryptographic credentials, we get that control. There is research that fouses on addressing the problem of linking transactions that the same credential is involved in. Several of them have been implemented by infrastructure providers. For example, IBM's Idemix credential scheme [17, 64], by Camenisch and Lysyanskaya, is probably the most well-known unlinkable multi-show anonymous credential scheme, and Microsoft's U-Prove [11, 50], by Brands, implements the Digital Credential scheme discussed in this thesis but does not provide unlinkability.

Compared to Idemix, U-Prove is more efficient. In addition, its problem of linking transactions the same credential made could be addressed by properly randomizing the Digital Credentials. In this thesis, we use malleable signatures to extend Brands' Digital Credential in a way that prevents transactions from being linked when the same credential is shown multiple times. We preserve the previous property that prevents the individual users from illegally manipulating the credentials. Our scheme is efficient and maintains user privacy (online or offline).

1.2 General Introduction to Credential Systems

1.2.1 General Concepts

- **Credential.**

  A credential is an object that a third party issues to an individual, which contains some attributes or usage privileges. The individual can later show the credential and claim those corresponding privileges. Some examples of credentials include paper-based credentials (*e.g.*, diplomas, academic degrees, driver's licenses, etc.), digital-based credentials (*e.g.*, user names, passwords, etc.), and particularly, cryptographic credentials (*e.g.*, public key certificates, X.509 certificates, Brands' Digital Credential [12], Camenisch and Lysyanskaya's anonymous credential [17], etc.).

- **Cryptographic Credentials.**

A credential in a cryptosystem is granted by an issuing organization to an individual user, which represents some specific attributes and privileges that can be shown to a verifying organization to gain access and permissions based on the specifics of the credential. A credential may be involved in some transactions (*e.g.*, issuing, showing, etc.), and may also contain usage policies, including issued data, expiry date, etc.

- **Digital Credential [12].**

  Generally, Digital Credentials are the digital equivalent of paper-based credentials issued by trusted parites. In this thesis, Digital Credentials specifically refer to Brands' Digital Credentials [12]. They are issued to applicants by trusted parties and then shown to verifiers by credential holders. Digital credentials are much more powerful than their physical counterparts (*e.g.,* users can selectively disclose a portion of the data contained in their Digital Credentials). More details are available in Section 2.2.

  In line with cryptographic tradition and Brands' Digital Credential model, we will use fictitious characters Alice as a Digital Credential holder and also the user in our system, and Bob (and David) as a Digital Credential verifier.

1.2.2 Main Entities, Roles and Actions

A basic model of a cryptographic credential system allows a user to obtain a credential from an issuer, and at a later time, show it to a verifier. This section gathers some main entities, operations, and additional functionality [9] to be used in the credential system throughout this thesis.

1. *User.* The user is an entity who participates in the issuing transactions and showing transactions (*i.e.*, a user is granted the credential by an issuer and then shows the credential to a verifier). The user holds personal attributes, credentials and associated additional data. In this thesis, the user (noted as Alice) is assumed to be capable of

communicating with the issuer to encode his/her attributes into the credentials and constructing the corresponding multi-show credential.

2. *Issuer.* The issuer is generally an organization, which issues a credential to an individual user. It may verify the validity of the user's attributes during the issuing process. In some systems, the issuer is able to revoke a previously issued credential. In this thesis, the issuer assists the user to create a credential that contains the user's valid attributes and then cryptographically signs the credential to grant it to the user. The certificate authority (CA) and other specific authorities (*e.g.*, the hospital) can play the role of the issuer.

3. *Verifier.* The verifier is the organization with whom the user interacts in the showing process. It makes sure that the shown credentials are properly signed, currently valid and rightfully encoded with the claimed privileges. In this thesis, the verifier (noted as Bob) sends a challenge (or a nonce) which can be a random integer, to the user in the showing process due to security concerns. Then it verifies if the shown credential and signature are valid by using the corresponding parameters obtained from the issuer.

4. *Issuance of a Credential.* The issuance of a credential occurs between the issuer and the user in the issuing process. To be more specific, the issuing organization may collaborate with the user to encode the user's personal attributes into a credential and cryptographically signs the credential.

5. *Showing of a Credential.* The showing of a credentials occurs between the user and the verifier in the showing process. To be more specific, an individual user, possessing a particular credential, shows it to a verifying organization to claim a privilege. Then the verifying organization checks the validity of the credential and the claimed privilege.

6. *Central Authority.* Some systems include a participant playing the role of a trusted central party, or third party. In Brands' Digital Credential system, central author-

ities, referred to as Credential Authorities (CAs) [11, 12], issue credentials to users and revoke previously issued credentials.

7. *Revocation of a Credential.* Some systems allow a credential to be revoked (usually by the issuer). A credential is invalid if it has been revoked.

8. *Pseudonym.* A pseudonym is a nickname between an individual user and an organization, which can hide the user's identity from the organization but still allows an ongoing relationship to be established between them. A user may hold more than one pseudonym.

9. *Multi-show Unlinkability* [43]. Some credential schemes have Multi-show Unlinkability (*e.g.,* [17]). In this thesis, multi-show unlinkability of a credential means that the credential can be shown to different verifiers without being linked to each other even if the issuers, the verifiers, and any other parties collude.

10. *User-unlinkability.* In this thesis, user-unlinkability of a credential means that the credential can be shown to different verifiers without being traced back to the user (the credential holder) even if the issuers, the verifiers, and any other parties collude.

1.3 Thesis Goals

This thesis has the goal of addressing the problem of multi-show unlinkability to Brands' Digital Credentials [11, 12]. Thus, we seek to create a protocol that allows Digital Credential multi-show:

1. Every transaction Alice makes will not be traced back to her (*i.e.,* user-unlinkability) (which is achievable with Brands' Digital Credential system).

2. Any two transactions Alice makes will not be linked to each other (*i.e.* multi-show unlinkability).

Transactions include issuance communications between Alice and the CA, and also showing communications between Alice and the verifiers, no matter whether they are the same verifiers or not.

1.4 Thesis Contributions

This thesis makes the following contributions:

1. **Protocol Extension: Multi-show Unlinkable Digital Credential**

   Avoid both kinds of linking attacks from the CA and the verifiers. With our improved protocol, multi-show unlinkability of Digital Credentials has been entirely achieved.

2. **System Simplification: CA Removal**

   Because of the previous property, there is no longer a need for the CA in our system. Instead, Digital Credentials can be directly issued by the actual authority (*e.g.*, the hospital).

3. **Simple Mathematics involved: Credential Creation**

   Alice can create her multi-show Digital Credentials by applying very simple calculations to her issued original Digital Credentials. Additionally, as the mathematical basis of our protocol, we propose an efficient method to find a generator of a cyclic subgroup (see Section 2.1.1.2 and 2.1.5 for definitions) of a given group, where the subgroup has a sufficiently large order (size).

4. **Turn Bad into Good: the Use of Malleability**

   Malleability is usually an undesirable property in an encryption algorithm, which allows an attacker to modify the plaintext just by conducting the corresponding modification on the ciphertext. However, we take advantage of the malleability of the RSA digital signature scheme to construct our multi-show credential. In the meantime, we also prevent Alice from forging valid-looking credentials and CA signatures on data that the CA has not signed, even with the use of malleability.

1.5 Thesis Outline

This thesis is structured as follows:

- Chapter 1 briefly introduces the main concepts in cryptographic credentials and the Digital Credential system as well as the need for privacy improvement in the credential system.
- Chapter 2 provides the required technical background for this thesis, including:
  - An introduction to selected concepts and assumptions in number theory, which underlie this thesis.
  - An introduction to Brands' Digital Credential system, which is the foundation of this thesis.
  - An introduction to malleable signatures and other related work (*e.g.,* Camenisch and Lysyanskaya's anonymous credentials) that can allow the credential multi-show capacity.
- Chapter 3 describes the contributions of this thesis, including the detailed algorithms for generating the proper cyclic group and constructing multi-show credentials, mathematics examples and security analysis for the proposed techniques.
- Chapter 4 introduces the proof of concept implementation for our proposed protocols by using a Python script as well as a discussion of the results.
- Finally, Chapter 5 summarizes the main findings of our work and provides some directions for further research.

Chapter 2

Background

In this chapter, we will start by gathering some key theoretical and cryptographic constructs that are used throughout this thesis. Then we will present a related literature review in the area of Digital Credentials and anonymous credential schemes. We will also introduce the concept of malleable signatures, used in our method.

2.1 Technical Concepts

2.1.1 Mathematics and Number Theory

This section introduces some foundations of mathematics and number theory, upon which the cryptographic primitives and protocols related to this thesis are built. The following definitions are adapted from [46, 61, 62], in which detailed presentations are available.

2.1.1.1 Prime Numbers and Primality Test

An integer $p > 1$ is a *prime* number if and only if its only divisors are ± 1 and $\pm p$. Prime numbers play a critical role in number theory and cryptography. For many cryptographic algorithms (such as the RSA cryptosystem and our techniques), it is necessary to select a

very large prime number (*e.g.,* with 1024 bits or more) at random. Unfortunately, there is no simple and efficient way of accomplishing this.

We will introduce the most popular (efficient) probabilistic test, known as Miller-Rabin, and will use this algorithm in our implementation. There are two properties of prime numbers that we will need.

First property: p is prime and a is a positive integer less than p. If either $a \bmod p = 1$ or $a \bmod p = -1$, then $a^2 \bmod p = 1$. Conversely, if $a^2 \bmod p = 1$, then $(a \bmod p)^2 = 1$, which is true only for $a \bmod p = 1$ or $a \bmod p = -1$.

Second property: Let p be a prime greater than 2. We can then write $p - 1 = 2^k q$ with $k > 0$, q odd. Let a be any integer in the range $1 < a < p - 1$. Then one of the two following conditions is true.

1. $a^q \bmod p = 1$, or equivalently, $a^q \equiv 1 \pmod{p}$;
2. There is some number j in the range $(1 \leq j \leq k)$ such that $a^{2^{j-1}q} \bmod p = -1 \bmod p = p - 1$, or equivalently, $a^{2^{j-1}q} \equiv -1 \pmod{p}$.

The pseudo code for primality testing by using the Miller-Rabin algorithm is as follows. The procedure TEST takes a candidate integer n as input and returns the result `composite` if n is definitely not a prime, and the result `inconclusive` if n may or may not be a prime.

Listing 2.1: TEST(n)

```
1. Find integers k, q, with k>0, q odd, so that n-1=2^k q;
2. Select a random integer a, 1<a<n-1;
3. If a^q mod n=1 then return ("inconclusive");
4. For j=0 to k-1 do
5. If a^(2^j q) mod n=n-1 then return ("inconclusive");
6. return ("composite").
```

Given an odd integer n that is not prime and a randomly chosen integer a with $1 < a < n - 1$, the probability that TEST will return `inconclusive` is less than 1/4. Thus,

we can repeatedly invoke TEST(n) for t times (*i.e.*, t different values of a are chosen), and the probability that TEST will fail to detect that n is not prime is less than $(1/4)^t$. If, at any point, TEST returns `composite`, then n is determined to be nonprime (TEST may be invoked less than t times).

In practice, an error probability of $(1/2)^{80}$ is usually tolerated, when using the Miller-Rabin algorithm to generate probable primes. With better estimates for the error probability, it is much less than $(1/4)^t$ when n is large enough. For example, when generating a 1000-bit probable prime, the Miller-Rabin algorithm with $t = 3$ repetitions will suffice (see Note 4.49 in [46]).

A result from number theory states that the primes near n are spaced on the average one every $\ln(n)$ integers, which means we need to test on the order of $\ln(n)$ integers before a prime is found. Because only odd numbers are primes (except 2), the correct figure is $0.5\ln(n)$. For example, if we want to find a prime with 1024 bits, we need to test $0.5\ln(2^{1024}) \approx 355$ numbers randomly generated.

Strong primes were once recommended for the RSA public-key cryptosystem [57]. However, in 2000, Rivest and Silvermany proved that strong primes offer negligible protection beyond that offered by random primes, as well as that the real protection comes from choosing large enough prime numbers. Even though using strong primes will not harm the protection, but extra effort is needed to generate them because strong primes are somewhat rare. Thus, RSA Security does not currently recommend the use of strong primes in the RSA key generation [58]. So, we will not make extra effort to generate strong primes in this thesis. Various definitions of strong prime are gathered in [58] as well as proof details.

2.1.1.2 Group

A *group* $\mathbb{G}$ is a set of elements with a binary operation denoted by $\cdot$ which is generic, including addition and multiplication, such that the following axioms are obeyed:

- *Closure*: $\forall a, b \in \mathbb{G}, a \cdot b \in \mathbb{G}$.

- *Associative*: $\forall a, b, c \in \mathbb{G}, (a \cdot b) \cdot c = (a \cdot b) \cdot c$.
- *Identity element*: There is an element $e \in \mathbb{G}$ such that $a \cdot e = e \cdot a, \forall a \in \mathbb{G}$.
- *Inverse element*: For each $a \in \mathbb{G}$, there is an element $a' \in \mathbb{G}$ such that $a \cdot a' = a' \cdot a = e$.

A group is *cyclic* if every element of $\mathbb{G}$ is a power a^k (k is an integer) of a fixed element $a \in \mathbb{G}$. The element a is a *generator* of $\mathbb{G}$, which generates the group $\mathbb{G}$.

2.1.1.3 Multiplicative Group

Multiplicative group is commonly used in cryptosystems as well as in this thesis. This important cryptographic concept will be introduced by starting with some basic definitions.

If a is an integer and n is a positive integer, we define $a \bmod n$ to be the remainder when a is divided by n. The integer n is called the *modulus*. For example, $11 \bmod 7 = 4$. Two integers a and b are said to be congruent modulo n (written as $a \equiv b \pmod{n}$), if $a \bmod n = b \bmod n$. For example, $15 \equiv 5 \pmod{10}$.

The *integers modulo* n, denoted $\mathbb{Z}_n$, is the set of nonnegative integers less than n:

$$\mathbb{Z}_n = \{0, 1, \cdots, (n-1)\}$$

Addition, subtraction, and multiplication in $\mathbb{Z}_n$ are performed modulo n. The set $\mathbb{Z}_n^*$ (defined as below) is a group of *order* $\phi(n)$ under the operation of multiplication modulo n (called *multiplicative group*), with identity element 1.

The *greatest common divisor* of integers a and b, denoted $\gcd(a, b)$, is the largest positive integer that divides both a and b.

The *multiplicative group* of $\mathbb{Z}_n$ is $\mathbb{Z}_n^*$, such that $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n | \gcd(a, n) = 1\}$. In particular, if n is a prime, then $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n | 1 \leq a \leq n-1\}$ (prime numbers are usually denoted as p and q). $\mathbb{Z}_q$, $\mathbb{G}_q$ and $\mathbb{Z}_p^*$ are particularly used in Brands' Digital Credential scheme [11, 12].

Another particular form of the general group $\mathbb{Z}_n^*$ which has been commonly used in cryptographic systems (especially in the RSA cryptosystem) is when $n = p \cdot q$ and p, q are

both prime numbers. In this thesis, $\mathbb{Z}_n^*$ ($\mathbb{Z}_n$) where n is a Blum integer will be used to for computation in both the RSA cryptosystem and Brands' Digital Credentials (see details in 2.1.6).

Euler totient (phi, ϕ) function: $\phi(n)$ denotes the number of integers in the interval $[1, n]$ which are relatively prime to n for $n > 1$. Two important properties of $\phi(n)$ are as follows:

- When p is a prime, $\phi(p) = p - 1$;
- If $\gcd(m, n) = 1$, $\phi(mn) = \phi(m)\phi(n)$.

The *order* of $\mathbb{Z}_n^*$ is defined as the number of elements in $\mathbb{Z}_n^*$, namely $|\mathbb{Z}_n^*|$. Thus, $|\mathbb{Z}_n^*| = \phi(n)$ follows the definitions of both Euler phi function and multiplicative group $\mathbb{Z}_n^*$. With *Euler's theorem*, if $a \in \mathbb{Z}_n^*$, then $a^{\phi(n)} \equiv 1 \pmod{n}$.

The *order* of $a \in \mathbb{Z}_n^*$, denoted as $ord(a)$, is the least positive integer t such that $a^t \equiv 1 \pmod{n}$. If the $ord(\alpha) = \phi(n)$, then α is said to be a *generator* of $\mathbb{Z}_n^*$ and the group $\mathbb{Z}_n^*$ is *cyclic*.

2.1.2 The RSA Problem

Fundamental Theorem of Arithmetic: Every integer $n \geq 2$ has a factorization as a product of prime powers:

$$n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k},$$

where the p_i are distinct primes, and the e_i are positive integers.

Integer Factorization Problem (FACTORING): given a positive integer n, find its prime factorization; that is, write $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ where the p_i are pairwise distinct primes and each $e_i \geq 1$.

The security of many cryptographic techniques relies on the intractability of the *factoring* problem, including the RSA public-key encryption scheme and signature scheme (which will be used in this thesis).

For the RSA cryptosystem, the message m space and ciphertext c space are both $\mathbb{Z}_n$ where $n = pq$ is a product of two randomly chosen distinct odd prime numbers.

The *RSA Problem* (RSAP): given a positive integer n that is a product of two distinct odd primes p and q, a positive integer e such that $\gcd(e, (p-1)(q-1)) = 1$, and an integer c, find an integer m such that $m^e \equiv c \pmod{n}$.

The intractability of the *RSAP* forms the basis for the security of the RSA signature scheme.

2.1.3 The Discrete Logarithm Problem

Discrete Logarithm Problem (DLP): given a prime p, a generator α of $\mathbb{Z}_p^*$, and an element $\beta \in \mathbb{Z}_p^*$, find the integer x, $0 \leq x \leq p-2$, such that $\alpha^x \equiv \beta \pmod{p}$.

Generalized Discrete Logarithm Problem (GDLP): given a finite cyclic group $\mathbb{G}$ of order n, a generator α, and an element $\beta \in \mathbb{G}$, find the integer x, $0 \leq x \leq n-1$, such that $\alpha^x = \beta$. Note that the difficulty of GDLP is independent of the generator.

The security of many cryptographic techniques depends on the intractability of the DLP, including Brands' Digital Credential system [11, 12] and the Diffie-Hellman key agreement [33].

The most obvious algorithm for GDLP is exhaustive search by successively computing $\alpha^0, \alpha^1, \alpha^2, \cdots$ until β is obtained. Even though there are more efficient ways to solve GDLP, it is still believed intractable under some particular constructions for $\mathbb{G}_q$ [9, 12]. One example is that $\mathbb{G}_q$, used in Brands' Digital Credential system, is a subgroup of $\mathbb{Z}_p^*$, such that $q|p-1$ with security parameters $|p| = 1024$ and $|q| = 160$.

More General GDLP: given a finite group $\mathbb{G}$ and elements $\alpha, \beta \in \mathbb{G}$, find an integer x such that $\alpha^x = \beta$, provided that such an integer exists.

Note that this problem is probably harder to solve, in general, than GDLP.

Bach [4] states that the DLP in $\mathbb{Z}_n^*$ (where n is a composite integer) is at least as difficult as the problem of factoring n. More specifically, the DLP in $\mathbb{Z}_n^*$ polytime reduces

to the combination of the integer factorization problem and the DLP in $\mathbb{Z}_p^*$ for each prime factor p of n. The intractability of the DLP in a finite and non-cyclic group $\mathbb{Z}_n^*$, where n is a Blum integer, underlies the protocols in this thesis.

2.1.4 Diffie-Hellman Problem

Diffie-Hellman Problem (DHP): given a prime p, a generator α of $\mathbb{Z}_p^*$, and elements $\alpha^a \bmod p$ and $\alpha^b \bmod p$, find $\alpha^{ab} \bmod p$.

If the DLP in $\mathbb{Z}_p^*$ could be efficiently solved, then the DHP could also be efficiently solved. That is, the DHP polynomial time reduces to the DLP. Shmuely [60] states that the DHP in $\mathbb{Z}_n^*$ (where $n = pq$ and $p\&q$ are odd primes) is at least as difficult as the problem of factoring n.

The Diffie-Hellman key agreement (D-H Exchange) provided the first practical solution to the key distribution problem, which allows two parties to establish a shared secret key over an open (unsafe) channel without meeting in advance or sharing any keying material [33]. This protocol is used in this thesis for sharing secret parameters between Alice and the CA.

The pseudo code for the basic Diffie-Hellman protocol between user A and B is as follows:

Listing 2.2: Diffie-Hellman Key Agreement (basic version)

1. An appropriate prime p and generator α of $\mathbb{Z}_p^*$ $(2 \leq \alpha \leq p-2)$ are selected and published;
2. A randomly chooses a secret $x, 1 \leq x \leq p-2$, and sends B $\alpha^x \bmod p$;
   B randomly chooses a secret $y, 1 \leq y \leq p-2$, and sends A $\alpha^y \bmod p$;
3. B receives α^x and computes the shared key as $K = (\alpha^x)^y \bmod p$;
   A receives α^y and computes the shared key as $K = (\alpha^y)^x \bmod p$.

Besides the most common groups used in practice, the multiplicative group $\mathbb{Z}_p^*$, the Diffie-Hellman protocol can be applied in any groups in which both the DLP is hard and

exponentiation is efficient. Thus this key agreement protocol can be used in the group $\mathbb{Z}_n^*$ where n is a Blum integer.

2.1.5 Subgroup

A non-empty subset $\mathbb{H}$ of a group $\mathbb{G}$ is a *subgroup* of $\mathbb{G}$, if $\mathbb{H}$ is itself a group with respect to the operation of $\mathbb{G}$, denoted as $\mathbb{H} \leq \mathbb{G}$. If $\mathbb{H} \neq \mathbb{G}$, then $\mathbb{H}$ is called a *proper* subgroup of $\mathbb{G}$, denoted as $\mathbb{H} < \mathbb{G}$.

Lagrange's Theorem: If $\mathbb{G}$ is a finite group and $\mathbb{H}$ is a subgroup of $\mathbb{G}$, then $|\mathbb{H}|$ divides $|\mathbb{G}|$.

In fact, every subgroup of a cyclic group is also cyclic.

The method of determining the order of a group element a in a (multiplicative) finite group $\mathbb{G}$ of order n, given the prime factorization $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$, is to compute a^t when t is equal to each positive factor of n and find the least t that makes $a^t = 1$ hold.

The method of finding a generator α of a cyclic group $\mathbb{G}$ of order n, given the prime factorization $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$, is to randomly choose a group element α and check if $\alpha^t = 1$ holds when t is equal to any positive factor (except n itself) of n. If this is the case, then randomly choose another group element α until $\alpha^t \neq 1$ holds for all the positive factors (less than n) of n.

In this thesis, it is desirable to have an element with high order in group $\mathbb{Z}_n^*$, which can generate a multiplicative subgroup of $\mathbb{Z}_n^*$ with large enough size. If $n = pq$, where p and q are distinct odd primes, then $\mathbb{Z}_n^*$ is a non-cyclic group of order $\phi(n) = \phi(p)\phi(q) = (p-1)(q-1)$ (see Section 2.1.1.3). Thus, there is no generator α that can generate all the elements in $\mathbb{Z}_n^*$. The maximum order of an element in $\mathbb{Z}_n^*$ is $\text{lcm}(p-1, q-1)$, which stands for the least common multiple of $p-1$ and $q-1$ [42]. Clearly:

$$\text{lcm}(p-1, q-1) \leq \frac{(p-1)(q-1)}{2}.$$

2.1.6 Blum Integers

A *Blum integer* is a composite integer n, which is the product of distinct prime integers p and q, where $p \equiv q \equiv 3 \pmod 4$.

To be more specific (from [42]), the numbers $N = P \cdot Q$ for which both following assumptions hold are called Blum integers:

1. P and Q are of equal size.
2. $P = Q = 3 \pmod 4$

Intractability Assumption [71]: No family of polynomial-size Boolean circuits can factor a polynomial fraction of the Blum integers.

Definition [42]: Let $g \in \mathbb{Z}_N^*$. The *exponentiation modulo a composite* function is defined by:

$$f_{g,N}(X) = g^X \pmod N.$$

Its inverse, the *discrete logarithm modulo a composite*, is defined only for $Z \in \mathbb{G}$ by:

$$f_{g,N}^{-1}(Z) = X,$$

for the unique $x \leq ord_N(g)$ s.t. $Z = f_{g,N}(X)$, where $\mathbb{G} = \{Z | \text{there exists } X \in \mathbb{Z}_N^* \text{ s.t. } Z = g^X \pmod N\}$ (*i.e.,* an element $g \in \mathbb{Z}_N^*$ generates a cyclic group $\mathbb{G} \subset \mathbb{Z}_N^*$ with order $ord_N(g)$).

Under the intractability assumption, the exponentiation modulo a Blum integer, $f_{g,N}(X)$, is a one-way function. In addition, under the commonly assumed intractability of factoring a Blum integer, Håstad, Schrift, and Shamir proved that all its bits are individually hard [42].

2.2 Digital Credential

The Digital Credentials technology is based on fundamental concepts proposed by Chaum [31], including blind signatures [22, 23], untraceable electronic cash [28], group signature

schemes [30], pseudonym credential systems [24, 27] and one-show blinding [25]. It also builds on work of commitment schemes by several researchers (see, for example, Brassard et al. [14], Chaum and Van Antwerpen [29, 26], Chaum [26], Pedersen [51, 52], and Van Heyst and Pedersen [66], etc.).

2.2.1 Introduction

In [11, 12], Brands proposed *Digital Credentials* as a privacy enhancing technology for end users. A Digital Credential is a data structure that allows its holder to determine for herself when, how, and to what extent she is willing to reveal her attributes to others, and to what extent others can link or trace this information. This means that users in this system do not need to trust third parties to protect their privacy (in particular, even if all parties in the system have unlimited computing power, they cannot learn more than what users willingly disclose).

Digital Credentials involve three parties: users, verifiers, and a CA. A user's attributes and another user-selected parameter can be considered as her private key; these are then encoded into the corresponding public key (*i.e.*, the Digital Credential). The digital signature of a trusted CA on the public key enables users to selectively disclose their specific attributes or even some properties of the attribute values while keeping the remaining attributes completely hidden from other parties, including parties in the transaction and any eavesdroppers that listen to the communication channel [2]. Figure 2.1 shows the overview of Brands' credential system.

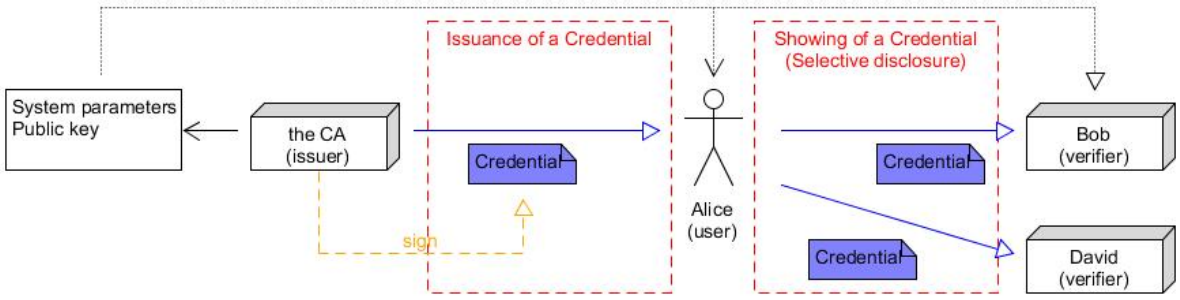

Figure 2.1: Overview of Brands' Digital Credential System

Brands proposed a brilliant way to enable the CA to encode attributes in a Digital Credential [12]. Here is the method of encoding $l \geq 1$ attributes, $(x_1, \cdots, x_l)$, in Alice's Digital Credential public key h:

- The attribute tuple

  $$(x_1, \cdots, x_l, \alpha)$$

  can also be seen as Alice's private key for the Digital Credential. Alice randomly chooses α from $\mathbb{Z}_q$ and never reveals it, which ensures that only she knows the entire secret key. The number α is called a blinding factor.

  Note that $x_1, ..., x_l$ must all be numbers from $\mathbb{Z}_q$. The way of mapping meaningful attribute information to numbers in $\mathbb{Z}_q$ is up to the CA. For example, the CA can make a public list (*e.g.*, Alice's gender can be expressed by a single bit, her credit card number can be the real digits, and her citizenship can be represented by a number between 0 and 266) and hash attributes that cannot be represented by a number in $\mathbb{Z}_q$ by using a collision-intractable hash function.

- The Digital Credential public key is

  $$h = h_0^{\alpha} g_1^{x_1} \cdots g_l^{x_l}.$$

  The elements $h_0, g_1, \cdots g_l$ have been randomly generated by the CA from a group $\mathbb{G}_q$ of prime order q (h_0 should not be equal to 1).

At the same time, the following basic security properties hold:

1. Regardless of the choice of l, assuming it is infeasible to compute discrete logarithms in $\mathbb{G}_q$, Alice cannot compute a Digital Credential public key for which she knows more than one secret key.

2. Assuming it is infeasible to compute discrete logarithms in $\mathbb{G}_q$, Bob cannot compute any secret key when presented with Alice's Digital Credential public key, regardless of which property or set of properties of $(x_1, \cdots, x_l)$ Alice discloses to him.

2.2.2 Proof of Knowledge

Informally, a *Proof of Knowledge* is a protocol by means of which one party can convince another that it "knows" a secret [12]. If the latter party does not learn anything at all about the secret other than the fact that the former party knows the secret, then this is a zero-knowledge proof of knowledge

The Schnorr protocol [59] provides a *proof of knowledge* for the discrete logarithm, which can be considered as the special case $l = 1$ in Brands' Digital Credential [12]. Given a group $\mathbb{G}_q$ of order q with generator g and $y = g^x \bmod p$, where g, y, p, q are public values, Alice proves knowledge of x to Bob by engaging in the following protocol:

1. Alice selects random value $r \in \mathbb{Z}_q$ and sends commitment $t = g^r \bmod p$ to Bob;

2. Bob then selects a random challenge $a \in \mathbb{Z}_q$ and sends it to Alice;

3. Alice creates response $s = r + ax \bmod q$ by using challenge a and the secret x;

4. Bob accepts if $g^s \equiv ty^a \pmod p$.

2.2.3 Brands' Restrictive Blinding

Brands proposed a technique called *restrictive blinding* (which is not a special case of Chaum's blind signature paradigm) in order to hide the Digital Credential from the CA,

so that the CA cannot later trace Alice's transactions [12]. Since Alice needs to reveal her Digital Credential and the corresponding signature issued by the CA when disclosing her attributes to others, the CA should not see either the credential public key or the signature when issuing the credential (*i.e.*, the CA needs to blindly sign Alice's Digital Credential). Using restrictive blind signatures, Brands proposed a basic issuing protocol as shown in Figure2.2. The whole package that Alice receives in the issuing protocol is called a Digital Credential.

Figure2.2 shows the basic issuing protocol from Brands [12]. In this protocol, the tuple $(x_1, \cdots, x_l)$ is Alice's private key (in particular, these correspond to her attributes) for the Digital Credential, and she will receive the CA's signature on the corresponding public key h. α_1 is a credential blinding factor randomly chosen and kept secret by Alice. α_2 and α_3 are also randomly chosen and kept secret by Alice and are used in the subsequent signature blinding computation. $\mathcal{H}(\cdot)$ is a strong one-way hash function. The outputs of $\mathcal{H}(\cdot)$ are less than q. As shown in the protocol, the CA generates a random number w_0 and then constructs the challenge a_0. Note that $(g_0, \cdots g_l, h_0)$ are system parameters (known by all parties) and $(y_1, \cdots, y_l, x_0)$ are private values known only by the CA.

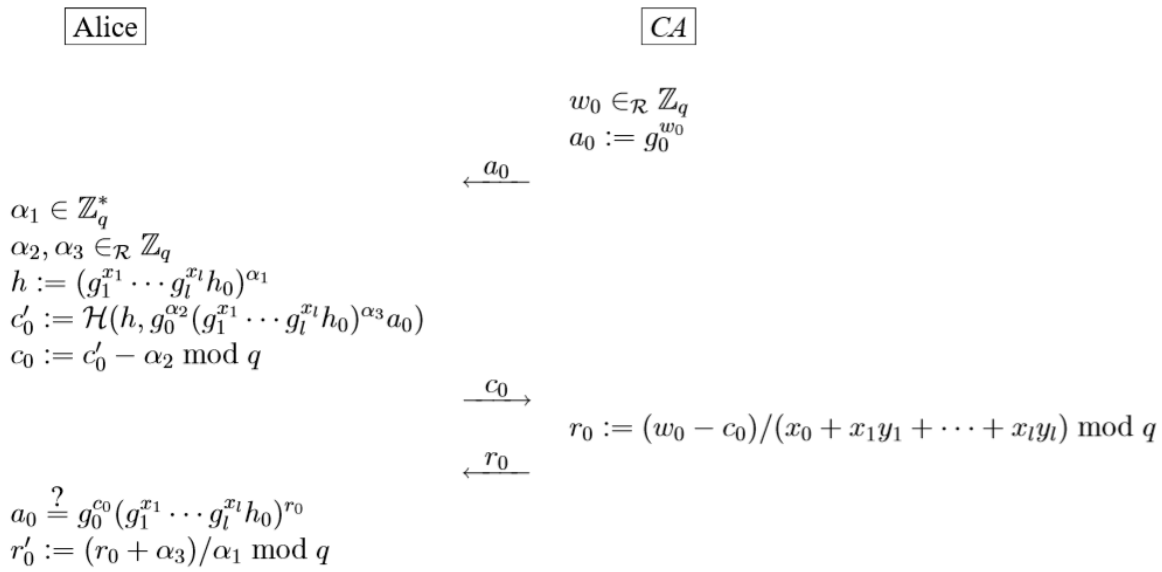

Figure 2.2: Brands' Basic Issuing Protocol [12]

After Alice and the CA complete this issuing protocol, Alice will hold the Digital Credential public key h and the CA's signature (c'_0, r'_0), which she can then show to Bob. In the meantime, all that the CA sees in this issuing process are (a_0, c_0, r_0), which are different from what Alice is going to disclose in a showing protocol to verifiers.

From the above Figure2.2, we can see that the CA must also know the value of Alice's attributes $(x_1, \cdots, x_l)$ in this basic issuing protocol. To address this problem, Brands proposed another issuing protocol with attribute hiding, which allows the CA to recertify previously issued Digital Credentials and then to issue new ones without knowing the attributes they contain (for environments where this level of privacy may be important).

2.2.4 Brands' Showing Protocol

In Brands' design, to show a Digital Credential to Bob, Alice transmits to him the Digital Credential public key, h, the CA's digital signature on the public key, $sig(h)$, and her digital signature, $(a, r_1, \cdots, r_{l+1})$.

Alice must convince Bob that she knows a secret key, $(x_1, \cdots, x_l, \alpha)$, corresponding to a Digital Credential public key $h = h_0^{\alpha} g_1^{x_1} \cdots g_l^{x_l}$. The values $r_1, \cdots, r_{l+1}$ are Alice's *response(s)* to Bob's *challenge*, the number c. The verification relation for them holds if Alice knows the entire secret key $(x_1, \cdots, x_l, \alpha)$ (this method is derived from a *proof of knowledge*). Thus, Alice's digital signature $(a, r_1, \cdots, r_{l+1})$ is also called a *signed proof*. Note that the tuple $(x_1, \cdots, x_l)$ is called a *DL-representation* (denoted as DLRep) of $h = g_1^{x_1} \cdots g_l^{x_l}$ with respect to base $(g_1, \cdots, g_l)$. The DLRep is intractable under appropriate construction (details available in [11]).

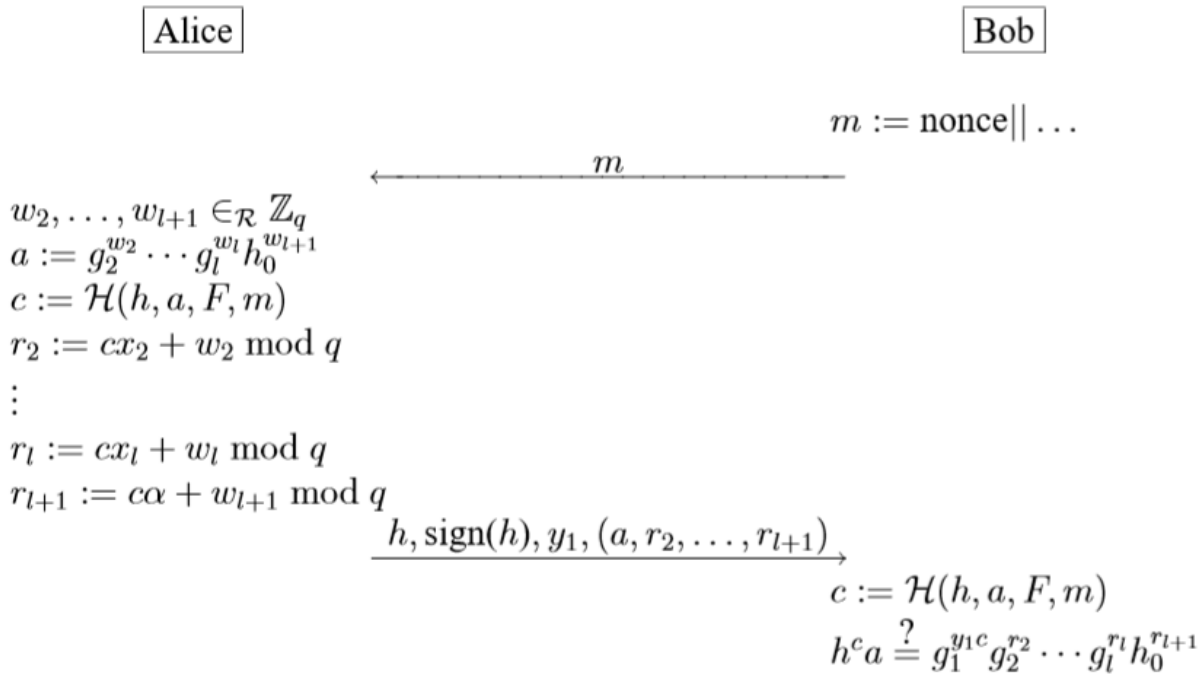

Figure 2.3: Brands' Signed Proof of $x_1 = y_1$ [12]

In Figure 2.3, Alice selectively discloses a value $y_1 \in \mathbb{Z}_q$, which is supposed to be equal to x_1, while keeping the rest of her attributes hidden. She reveals the value of x_1 and does a *signed proof* for what remains in h (*i.e.,* she proves knowledge of the secret key $(\alpha, x_2, \cdots, x_l)$ to $h/g_1^{y_1} = h_0^{\alpha} g_2^{x_2} \cdots g_l^{x_l}$ by generating the response $(r_2, \cdots, r_{l+1})$).

More specifically, the symbol $\in_{\mathcal{R}}$ means that the variable on its left is chosen independently and uniformly at random from the set specified on its right. The number a is called Alice's *initial witness*. The symbol F that is hashed when forming c represents a unique description of the attribute property that Alice demonstrates. m is an arbitrary message that at minimum contains a nonce of Bob to prevent a replay attack. The tuple $(a, r_2, \cdots, r_l)$ is Alice's digital signature on the message m.

2.3 Malleable Signature

This thesis builds on the concept of *malleable signatures* as proposed and discussed in a growing body of research papers. Chase et al. [20, 21] gave new (extended) definitions of malleable signatures and malleable zero-knowledge proofs, which allow them to construct malleable signatures with a wider range of transformation classes and then construct delegatable anonymous credentials from these signatures. Practical implementations of malleable signature schemes have also been demonstrated by various researchers (*e.g.*, Pöhls et al. [54] implemented one redactable and three sanitizable signature schemes on secure, but computationally bounded, smart cards).

Generally, a signature scheme is *malleable* if, for a given message and its signature, it is possible to efficiently modify the signature to be a valid signature on a related message without using the private signing key [21]. Specifically, a signature scheme is malleable in this thesis if, for a given Digital Credential public key and the CA's signature on it, modifying the signature to be an another valid signature on the corresponding modification of the Digital Credential public key can be done efficiently without using the CA's private key or changing the attributes contained in the original credential.

There are other types (instantiations) of malleable signatures that have been defined and explored in previous papers, but none of them are affiliated with Brands' Digital Credential scheme and, particularly, none have been used to make this credential scheme multi-show. Examples include sanitizable signatures [3, 15, 35], redactable signatures [44, 16, 45, 63], and homomorphic signature schemes [44, 65].

Note that sanitizable signatures, redactable signatures, and homomorphic signatures are types of malleable signatures that are not suitable for the functionality we wish to achieve in this thesis. What we require is that if s is a valid signature on a credential public key h, h can be modified to h' and s can be modified to s' (without knowledge of the private signing key) such that s' is a valid signature on h', and $\{h, h'\}$ are unlinkable and $\{s, s'\}$ are unlinkable.

Ateniese et al. [3] presented the notion of sanitizable signatures, which allow another

party to modify designated portions of a document and then produce a valid signature on the modified document without help from the signer. Brzuska et al. [15] then constructed a sanitizable signature scheme with perfect unlinkability between sanitized message-signature pairs of the same document. Fleischhacker et al. [35] then developed this by re-randomizing the message signing and verification keys.

The concept of redactable signature schemes was first introduced in [44, 63] , which allows removing particular portions from a signed document without invaliding the signature and producing the corresponding signature on the redacted document. This concept was viewed as an instance of a homomorphic signature scheme by Steinfeld et al. [44], while Johnson et al. [63] termed it as "Content Extraction Signature". Further research on this topic includes, for example, Camenisch et al. [16] who proposed unlinkable redactable signatures to construct an efficient and practical anonymous credential system, and Ma et al. [45] who presented an efficient construction of authenticated data redaction with fine-grained redaction control.

The specifications of a homomorphic signature scheme include a message space $\mathcal{M}$, a signed message space $\mathcal{Y}$, a set of private keys $\mathcal{K}$, and a set of public keys $\mathcal{K}'$. A homomorphic signature is defined as follows [44, 65]:

Assume we have a signature algorithm Sig and a verification algorithm Vrf, together with a binary operation "$\cdot$". Then we say that Sig is a homomorphic signature with respect to $\cdot$ if it comes with an efficient family of binary operations $*_{k'} : \mathcal{Y} \times \mathcal{Y} \to \mathcal{Y}$ such that, having the messages $m, m' \in \mathcal{M}$ and the signatures $y, y' \in \mathcal{Y}$ for which

$$Vrf(k', m, y) = ok = Vrf(k', m', y')$$

for a public key $k' \in \mathcal{K}'$, then there is a secret key $k \in \mathcal{K}$ such that

$$y *_{k'} y' = Sig(k, m \cdot m') \quad and \quad Vrf(k', m \cdot m', y *_{k'} y') = ok.$$

Fuchsbaue and Hanser et al. [38, 36] proposed a type of structure-preserving signature scheme on equivalence classes, which allows to sign group element vectors and to consistently randomize the message and its signature. Based on this novel signature scheme,

some constructions of multi-show attribute-based anonymous credential systems were also provided.

Wei et al. [70] proposed bounded vector signatures which allow a user to increase the value embedded in any component of the signed vectors to a pre-defined bound without access to the signing key. Backes et al. [5] introduced delegatable functional signatures which allow the signer to delegate the signing capability to another party and also specify how this party can modify the signature or further delegate its capability. Blömer et al. [10] identified a new primitive called *dynamically malleable signatures*, in which the set of allowed transformations is not static but can be changed over time for each signature.

2.4 Related Work

The related work involved with smart devices has been excluded in this section as the use of smartcards is not considered in this thesis. The main drawback of using smartcards is that it is expensive to both build and distribute a smartcard for each user. In addition, it is difficult to revoke a smartcard token.

Compared to the related work that has achieved multi-show unlinkability, a big advantage of our credential system is that it is built on much simpler mathematics. This makes it easy to understand, implement and analyze.

2.4.1 Self-blindable and Attribute-based Credentials

An attribute-based credential (ABC) scheme allows a user, given a set of attributes, to prove ownership of these attributes to a verifer and selectively disclose some of them while keeping the others secret [56]. The most famous ABC schemes that have been implemented are Idemix and U-Prove.

Verheul [67] introduced self-blindable credentials, which allow users to modify a credential each time before they show it to the verifiers without invaliding it. The modifications

are done in such a way that the multiple transactions cannot be linked to each other [43]. Verheul [67] also provided two simple anonymous credential systems built upon this concept.

Based on the definitions of both above credential schemes, the credential scheme introduced in this thesis can be categorized as a self-blindable attribute-based credential scheme.

In 2004, Hanzlik et al. [39] proposed an extension to the U-Prove system which enables the U-Prove tokens to be multi-show by applying the building block provided by Verheul in [67]. Unfortunately, this extension has been found forgeable by Verheul et al. [69] in 2005.

2.4.2 Linkability and Malleability in Self-blindable Credentials

In 2015, Hoepman et al. [43] introduced the definitions of *unlinkability* and *unforgeability*. Informally, *unlinkability* means that an adversary gets to see two traces and cannot decide whether they belong to the same user with non-negligible advantage. *Unforgeability* means that an adversary cannot generate new valid credentials based on existing ones with non-negligible probability.

[43] also gave a criterion which can indicate if a self-blindable credential scheme is linkable or forgeable (malleability is a particular kind of forgeability which occurs by exploiting existing credentials). A number of broken schemes were also examined in [43]. They claimed the common theme in these failures is that the dependence of the public key and signature on the private key of the credential (particularly, the verification equation used in the showing protocol) can often be exploited to achieve linkability or malleability.

They were concerned with how the blinded public key and blinded signature depend on the private key and blinding factor:

1. Suppose the public key is linear in both the private key and blinding factor. If the signature scheme is linear in the second but not the first argument, then there is

linkability. If the signature is linear in both arguments, then there is malleability.

2. Suppose both the public key and the signature are liner in the private key. If it is infeasible for a malicious user to calculate the corresponding private key, then the system would not be malleable.

3. In the case that the public key and the signature are both nonlinear in the private key, or both nonlinear in the blinding factor, it is not necessary to send the public key to the verifier at all. Thus, the public key can play no role in either linkability and malleability. IBM Identity Mixer (Idemix) has applied this approach. However, Idemix is not considered to be self-blindable.

They concluded that it would be possible to create self-blindable credential schemes that are unlinkable and unmalleable but it might be difficult. This thesis introduces an extension to Brands' Digital Credential system, which would become a self-blindable, unlinkable and unmalleable credential scheme (the details for security analysis are available in Section 3.5).

In our case, the public key and the signature are both not linear in the private key or the blinding factor (which is the third case listed above). The credential (public key and signature) would be randomized each time before being sent to the verifier. To be more specific, the original issued credential would never be shown to anyone while the randomized credential (which contains the exact same meaningful data) would be sent to the verifier. This can make [12] both self-blindable and unlinkable. In the meantime, we also prevent the users from forging a new valid credential which contains different information. (Details are available in the next chapter.)

In 2017, Ringers et al. [56] provided an unlinkable attribute-based credential scheme based on the concept of *self-blindability*. However, this scheme is not subject to the criteria (shown above) proposed in their previous work [43], as the criteria only hold for deterministic signature schemes while the one in [56] is non-deterministic.

Pussewalage et al. [55] proposed a multi-show attribute-based credential scheme, influenced by the standard U-prove credential, and then developed an access control scheme

for a multi-domain collaborative e-health environment. Their scheme requires an online issuer in the verification protocol which will damage the convenience and efficiency, while the issuer is not involved in our verification process.

2.4.3 Anonymous Credential Systems

Since Brands' work [11, 12], a number of variations and alternative credentials schemes have been published, particularly those based on the unlinkable anonymous credentials of Camenisch and Lysyanskaya (see [17] for the original proposal).

Anonymous credentials [17] construct a pseudonym of the user for use with a specific organization. Each user will have her own master secret key. Each pseudonym is tagged with a value, called a validating tag, which is statistically independent of the user's master secret key. A credential will be issued by the organization to a pseudonym. Proof of possession of a credential is achieved through a statistical zero-knowledge proof of knowledge of a correctly-formed validating tag and its corresponding credential. The actual credential is never revealed in a showing protocol (which is why it is multi-show); thus, Camenisch and Lysyanskaya's credential system is quite different from Brands' Digital Credential system.

Idemix relies on the Camenisch–Lysyanskaya (CL) signature scheme [18] proposed in 2002, which is under the Strong RSA assumption. Camenisch and Lysyanskaya [19] then, in 2004, described another signature scheme based on elliptic curves and bilinear mappings. Brands' system and the CL-based systems (the optimized CL-RSA system of [18] and the CL-DL system of [19]) are compared in [13]. Both systems have advantages and disadvantages. The security assumptions are less well-accepted and the showing of credentials is more expensive for CL-based systems (zero-knowledge proofs can involve significant computation and may not be ideally suited to all situations and environments), while Brands' scheme has not achieved multi-show unlinkability. In this thesis, we propose a simple mechanism to make the more efficient Brands' credentials multi-show.

Besides the CL-based systems, there are other multi-show anonymous credentials based on different novel techniques (*e.g.*, RSA-representation [53], structure-preserving signatures

on equivalence classes [38, 36], unlinkable redactable signatures [16], and algebraic Message Authentication Codes in prime-order groups [7]). In 2013, Baldimtsi and Lysyanskaya [6] defined *blind signatures with attributes* and, from it, constructed the single-use anonymous credentials.

Furthermore, there are various delegatable anonymous credentials built on different types of malleable signature schemes [10, 20, 21, 32]. There has also been research on revoking anonymous credentials. In 2007, Brands et al. [13] proposed a practical solution for revoking anonymous credentials issued to unknown users. Bhaskar et al. [8] pointed out a linkability attack by a corrupt revocation authority and verifiers one year later.

Recently, Yu et al. [72] stated that it was crucial for a cryptographic system to achieve both anonymity and accountability simultaneously. They also summarized two directions for recent research in anonymous credential systems:

1. Decentralized anonymous credential systems (DACS) constructed by applying the blockchain technique, which was first proposed by Garman et al. [37].

2. Lattice-based anonymous credential systems, which are believed to be secure against quantum computers.

2.5 Chapter Summary

This chapter has provided an overview of the required background knowledge for this thesis, including number theory, cryptographic assumptions and Brands' Digital Credential system, as well as the related work in malleable and anonymous signature schemes which have achieved multi-show unlinkability. The main work is listed and compared in Table 2.1 (Note that the "✓" represents the work has confirmed the corresponding property or capability, while the "?" represents that we are unsure if the work has the corresponding property or capability).

Table 2.1: Main Related Work

	Attribute -Based	Selective Disclosure	Multi-show Unlinkability	Self-Blindability	Unforgeability	Simple Mathematics
Idemix	✓	✓	✓		✓	
U-Prove	✓	✓			✓	✓
Verheul [67]			?	✓	✓	
Ringers et al. [56]	✓	✓	✓	✓	✓	
Pussewalage et al. [55]	✓	?	✓	?	✓	
Our Scheme	✓	✓	✓	✓	✓	✓

Chapter 3

Multi-Show Unlinkability Extension for Digital Credentials

This scheme was first introduced in [34]. This chapter provides more details to the protocols and some specific steps to the algorithms.

3.1 Limitations of the Existing Digital Credential Protocol

In a general Digital Credential system, let us say Alice has her own Digital Credential public key h, which contains her attributes, (*e.g.*, age (25), credit card number (x), etc.). In order for her credential to be valid, she needs to get the CA's signature on her credential public key (*i.e.*, $sig(h)$). Then if Alice wants to buy alcohol from Bob, she will need to prove that she is of legal age and thus she will show 25, h and $sig(h)$ to Bob. If she also wants to buy a new laptop from David, she will show x, h and $sig(h)$ to David.

From this simple example of credential usage, we can identify at least two kinds of linking attacks that can be used to track Alice:

1. The CA can collude with Bob so that Bob learns Alice's credit card number, and/or

the CA can collude with David so that David learns Alice's age (termed as *issuer-linking attack*).

2. Bob and David can collude directly so that they each learns Alice's age and credit card number (termed as *verifier-linking attack*). (Actually, Alice has never reveal her identity to either Bob or David. Thus, they only know the age and this credit card number belong to the same person while are not aware that this person is Alice.)

In Brands' proposed Digital Credential system, the issuer-linking attack is mitigated through the use of *blind signatures*: the CA does not see Alice's actual credential public key and signature; thus, the CA cannot recognize when she uses it with Bob (or David) and so cannot collude with them. On the other hand, the verifier-linking attack cannot be avoided. Alice needs to show her credential public key h when she needs to reveal any of the attributes contained in h, even though she may reveal different attributes to different verifiers. In this case, Bob/David would learn that all these actions are performed by the same individual and all the transactions involving this specific Digital Credential are linked.

Note that if Alice discloses attributes or any combinations of attributes that are unique to her (*e.g.*, her passport number, or her name and home address), then she can be tracked regardless of the protections put in place in the issuing and showing protocols themselves. This is not a problem that can be solved through technical means; hence we do not address this situation in this thesis. (In general, Alice needs to be careful about what attributes she reveals to which parties if she is concerned about protecting her privacy.)

In summary, by applying Brands' techniques, Digital Credentials can only be used once (*i.e.*, no Multi-show Unlinkability).

3.2 High-Level Description of the Proposed Technique

In Brands' original proposal, the blinding process happens once (*i.e.*, between Alice and the CA) during the issuing protocol, so that the CA cannot later trace Alice's movements

as she uses her credential and signature. However, once Alice obtains her issued credential and signature, she would use these with all verifiers. Her transactions can therefore be linked across different verifiers; furthermore, collusion among verifiers is possible so that each of them can learn more about Alice. Our proposal is to have the blinding process happen in the showing protocol with every transaction. In this way, different verifiers will not know that they have interacted with the same entity (*i.e.*, Alice) and so linking of her transactions and collusion among verifiers are both prevented. (Note that it also remains true that the CA cannot trace Alice's movements because Alice is using a randomized credential every time.)

In order to achieve the blinding process in the showing protocol, it is necessary for Alice to not only randomize the credential, but also correspondingly randomize the CA's signature so that this "new" credential can be verified. In other words, Alice requires a "new" CA signature, unlinkable to the original signature, which can verify the "new" credential using the CA's public key, but of course without requiring the CA's private key to create this "new" signature.

To address this counterintuitive problem, we build on the concept of malleable signatures discussed in Section 2.3. The challenge is to find an efficient malleable signature construction that allows the signature to be blinded / randomized in such a way that

1. It is provably unlinkable to the original signature;
2. It can be verified using the original (*i.e.*, the CA's) public key; and
3. It shows integrity and authenticity of only a specific piece of data (*i.e.*, the corresponding randomized credential). In particular, Alice must not be able to use malleability to construct a valid-looking CA signature on data that the CA would not have signed; it must be a signature on a randomized credential that contains all and only her original attributes.

Our construction uses the malleability of the RSA digital signature scheme. We demonstrate our proposal by four parts: *Initialization*, *Issuing* protocol, *Showing* protocol and

Verification. Figure 3.1 presents the high-level description of our construction. It also works when there are multiple users in our system.

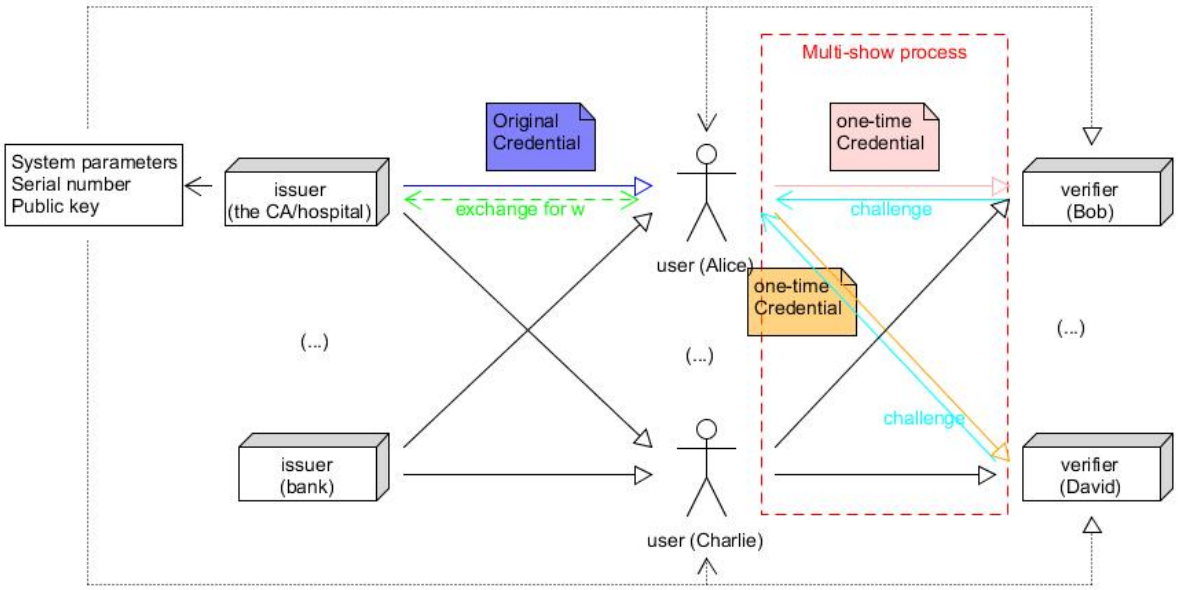

Figure 3.1: High-level Description of Our Construction

There are basically two steps that need to be done in *Initialization*:

1. The CA generates appropriate system parameters and makes them available to all users in the system.

2. The CA generates its key pair for credential signing and verification purposes.

The main steps of our proposed *Issuing* protocol are as follows:

1. Alice has her *original credential* public key h, which contains all her attributes, such as her age (20), credit card number (x), etc.

2. The CA then adds its serial number (a fixed value) as the last attribute of Alice's original credential.

3. Alice and the CA communicate with each other to get w via a Diffie-Hellman (D-H) key exchange. (If they fully trust each other, it is not necessary to do D-H key exchange to obtain w; rather, this parameter could be generated by either of them and simply given to the other.)

4. The CA signs both h and w to get sig_h and sig_w, respectively. The CA then sends these signatures to Alice.

Note that either Alice or the CA could create Alice's credential public key h and show it to the other. In practice, they need to meet in person to ensure that these attributes do in fact belong to Alice and this credential has not been comprised. In other words, the integrity of the credential must be guaranteed but confidentiality between Alice and the CA is typically less important. Note that if attributes need to be added, deleted, or modified, a new credential will need to be issued by the CA. This is identical to the physical credentials that we use today. For example, we can view the personal information presented on a driver's license as attributes. Any data (*e.g.*, the home address) on an existing license that needs to be modified will require a new driver's license to be issued.

The main steps of our proposed *Showing* protocol are as follows:

1. Alice uses her original credential public key, her parameter w, and the values sig_h and sig_w to generate a *one-time showing credential* public key h_{show} and a corresponding signature sig_{show}.

2. Alice sends the value(s) of the attribute(s) she wants to show, the CA's serial number, h_{show}, and sig_{show} to the verifier. (We assume that the position of the attributes in the credential (*i.e.*, the position i of each attribute) will be standardized and known to all verifiers.)

3. If Alice wants to show any attribute contained in the credential to other verifiers, she needs to repeat Steps 1 and 2.

And the main steps of proposed *Verification* are as follows:

1. Bob or other verifiers obtain the verification key (public key) and the serial number from the appropriate CA.
2. Verify if the one-time showing credential public key and signature are valid or not.
3. Verify if the disclosed attribute(s) is (are) valid or not.

As with Brands' original proposal, our system involves three related parties: the CA, *users*, and *verifiers*. The CA has an RSA key pair for credential signature purposes (with private signing exponent d and public verification exponent e); it issues each user one signed Digital Credential. The user, Alice, uses her original credential public key h to create a one-time showing credential public key and the corresponding one-time showing signature, which will be transmitted to a verifier. A verifier, Bob or David, will get the verification public key from the appropriate CA and use it to verify whether the one-time showing credential public key and the corresponding signature are valid or not. He will then verify if the user's disclosed attributes are valid or not.

3.3 Removal of the CA

As a side benefit for our method, we no longer need a CA entity in our Digital Credential System. An actual authority (*e.g.*, the hospital and the bank) can directly issue the original credential to Alice. This simplifies the credential system, but still preserves all of the functionality benefits provided by the CA.

Specifically, from Brands' basic issuing protocol, the CA has two important features: Bind and Blind. "Bind" means to bind the Digital Credentials (public keys) to their corresponding users, and "Blind" is to avoid the CA and Bob/David from colluding to trace Alice's transactions. By applying Brands' *Restrictive Blinding*, the CA does not know that the Digital Credential was issued to Alice. Therefore, the issuer-linking attack can be avoided, and the user-unlinkablility is achievable as well.

Furthermore, Brands' Digital Credential can only be shown once, so a CA is needed to refresh the credentials in order to avoid the verifier-linking attack (*i.e.*, the user uses

different credentials with every showing transaction). However, our proposed technique can easily prevent against both of issuer-linking attack and verifier-linking attack and achieve both user-unlinkablility and mutli-show unlinkablility (see details in Section 3.5).

There are some benefits derived from the removal of the CA. Firstly, if users can directly communicate with the actual authorities and get valid Digital Credentials, the process of issuing protocol would be more efficient to the users. For instance, if Alice wants to obtain a credential with her birthday encoded from a traditional CA, she will need to convince the CA about her birthday by (maybe) showing her birth certificate (which is from the hospital she was born in). Without the CA involved, Alice would be given her credential with her birthday (her gender and other related attributes) encoded directly from the hospital when she was born. This process, done with the hospital, can be applied to other credentials issued to Alice by their corresponding authorities (like the bank). She just needs to go through the exact same process with the bank as she has done with the hospital.

In addition, all the costs (*e.g.,* the establishment for the institutions, or the salaries paid to the officers) generated by the CAs would be saved. Just like in Bitcoin system [47], without going through financial institutes in the online payment system, the costs for constructing and maintaining the banks would be saved.

However, this would cause some drawbacks. For example, with the system that involves the CA, Alice only needs to obtain one Digital Credential that can possess ten different attributes issued by the same CA. By using the system without the CA, Alice would need ten different Digital Credentials issued by ten different authorities, while each credential only contains one attribute. It is obvious that without the use of the CA, Alice (the user) would need to spend extra effort to manage and safely store the additional information of the credentials (such information includes specifics of the credentials and the corresponding attributes that were encoded in, the authorities which the credentials were issued from along with their corresponding serial numbers), since she may have many different credentials from different authorities. It also requires more effort for Bob (the verifier) to manage and obtain different serial numbers and verifying keys from different authorities.

From another perspective, when using the system that involves the CA, if the CA's private (issuing) key is compromised, Alice's ten attributes issued by that private key may also be compromised. The attacker that has the CA's private key can issue a credential with different attributes for Alice, and this fake credential will appear to be valid to a verifier. On the other hand, when using the system that does not involve the CA, if only one of the ten authorities has a compromised private key, Alice will just have one of her ten attributes compromised. If Alice's private key is compromised (and it is highly probable that Alice will sign all her credentials with the same private key), it is possible that all her attributes will be compromised in the system either with or without a CA.

In conclusion, we would gain more benefits in a system without the CA. Actual authorities (*e.g.*, the hospital) in our system could perform the functionality of the CA. They will generate system parameters and conduct an issuing protocol with the users. In the following part of this thesis, we will use the "hospital" as an instance to refer to our actual authorities, which will perform the functionality of the CA in our proposed technique.

3.4 Detailed Proposal

In this section, we will describe our proposal in more detail, explain the related mathematics, and give some specific examples.

3.4.1 Technical Processes

The whole algorithm will be presented as the same process as we demonstrated in Section 3.2 with the specifics.

3.4.1.1 Desired Subgroup Construction

This section describes the technique behind the construction of our *desired subgroup*, noted as $\mathbb{G}_s$ (a cyclic group with the order of s), and the algorithm for generating $\mathbb{G}_s$ will be presented in the next section.

As we can see from the introduction of Brands' Digital Credential, all the related results fall into a group $\mathbb{G}_q$ with a prime order of q. From the RSA signature construction $sig = h^d \bmod n$, we can see that the signature sig falls into a multiplicative group $\mathbb{Z}_n^*$ where n is a product of two prime numbers. However, both calculations related to Brands' Digital Credentials and the RSA signatures should be within the same group.

This can be achieved if n is a Blum integer (see Section 3.5.2). Thus, both the results of the Digital Credential public keys and the RSA signatures will fall into the same multiplicative group $\mathbb{Z}_n^*$, where n is a Blum integer.

However, the multiplicative group $\mathbb{Z}_n^*$ is not a cyclic group, which means there is no element that can generate all the elements in $\mathbb{Z}_n^*$. So it is desired to find a generator that can generate a large enough (for security reasons) cyclic group $\mathbb{G}_s < \mathbb{Z}_n^*$ (*i.e.,* $\mathbb{G}_s$ is a proper subgroup of $\mathbb{Z}_n^*$).

Based on some properties of the multiplicative group $\mathbb{Z}_n^*$,

$$\phi(n) = (p-1)(q-1),$$

$$gen^{\phi(n)} \equiv 1 \pmod{n}.$$

And the properties of subgroup,

$$s \leq \text{lcm}(p-1, q-1) \leq \frac{(p-1)(q-1)}{2} = \frac{\phi(n)}{2},$$

$$gen^{s} \equiv 1 \pmod{n}.$$

We can see that s is a factor of $\phi(n)$ (*i.e.,* $s|\phi(n)$).

Here is the construction of the integer n:

$$n = p \cdot q,$$

$$p = 2xp' + 1,\ q = 2yq' + 1,$$

where p, q, p' and q' are primes, x and y are odd numbers. Thus, $\phi(n) = (p-1)(q-1) = 2xp' \cdot 2yq' = 4xyp'q'$.

From this construction, we can see that the set of (p, q) already satisfies the requirement of $p \equiv q \equiv 3 \pmod 4$ for constructing a Blum integer n (we take p as an example here and q can be proved in the same way):

It is obvious that both p' and x are odd integers.

Assume

$$x = 2i + 1,\ p' = 2j + 1,$$

where i and j are positive integers.

Then

$$p = 2xp' + 1 = 2 \cdot (2i + 1) \cdot (2j + 1) + 1 = 8ij + 4i + 4j + 3 = 4 \cdot (2ij + i + j) + 3,$$

$$p \bmod 4 = [4 \cdot (2ij + i + j) + 3] \bmod 4 = 3.$$

With this construction of n, the order s of the subgroup $\mathbb{G}_s$ can reach its maximum $\phi(n)/2$, since $\text{lcm}(p-1, q-1) = \text{lcm}(2xp', 2yq') = 2xyp'q' = \phi(n)/2$ where $\gcd(x, y) = 1$ (*e.g.,* both x and y are primes). If x and y are not relatively prime to each other, then $\text{lcm}(p-1, q-1) < \phi(n)/2$.

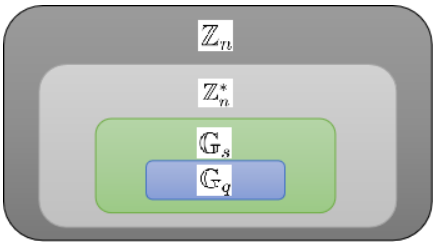

Figure 3.2: $\mathbb{G}_q$, $\mathbb{G}_s$, $\mathbb{Z}_n^*$, and $\mathbb{Z}_n$

The relationships among $\mathbb{G}_q$, $\mathbb{G}_s$, $\mathbb{Z}_n^*$, and $\mathbb{Z}_n$ are shown in Figure 3.2:

- $\mathbb{Z}_n$ is an additive group which is a set of integers less than n;
- $\mathbb{Z}_n^*$ is a multiplicative group containing the elements in $[1, n]$ that are relatively prime to n, which means $\mathbb{Z}_n^*$ is a subset of $\mathbb{Z}_n$ (*i.e.,* $\mathbb{Z}_n^* \subset \mathbb{Z}_n$);

- $\mathbb{G}_s$ is a subgroup of $\mathbb{Z}_n^*$, and $\max(s) = \phi(n)/2$ where $\gcd(x, y) = 1$;
- $\mathbb{G}_q$ is a specific instance of $\mathbb{G}_s$ where $q < \phi(n)/2$ is a prime, which is also a subgroup of $\mathbb{Z}_p^*$ used in Brands' scheme. Both $\mathbb{G}_s$ and $\mathbb{G}_q$ are cyclic multiplicative groups.

3.4.1.2 Desired Subgroup Generation

In this section, we present the algorithm for generating a desired subgroup $\mathbb{G}_s$ and the system parameters:

1. Randomly choose prime numbers $p\&q$, such that

$$|p| = |q| \quad \text{and} \quad p \equiv q \equiv 3 \pmod 4.$$

2. Compute

$$n = p \cdot q.$$

3. Find the generator (called *gen*), such that *gen* generates $\mathbb{G}_s$ where $\mathbb{G}_s < \mathbb{Z}_n^*$ should be a large enough group.

4. Any system parameter (*e.g.*, g_1) can be generated as

$$g_1 = gen^a \bmod n,$$

with random $a \in \mathbb{G}_s$.

To be more specific, from the constructions of the Digital Credential public keys and the RSA signatures, we can see that all the related results will be calculated within the *subgroup* $\mathbb{G}_s$. To maintain security, the order s of $\mathbb{G}_s$ should be large enough.

Here are the specific steps for finding a desired generator *gen* which can generate $\mathbb{G}_s$ with a large enough order s, termed as *Desired Subgroup Generation* (DSG) Algorithm:

1. Randomly choose prime numbers p' and q', such that $|p'| = |q'|$.

2. Let

$$p = 2xp' + 1,\ q = 2yq' + 1,$$

where x and y are primes with the same size (*e.g.*, 17, 19, 23, 29, and 31 are primes with the size of 5 bits).

Manipulate x, y and p', q' to make both p and q prime numbers:

- Increase the value of x from its lower bound to make p a prime number and $|p| = |p'| + |x|$. If the value of x has reached its upper bound, then randomly choose another prime number p'.
- Increase the value of y from its lower bound to make q a prime number as well as $|q| = |q'| + |y|$ and $q \neq p$. If the value of y has reached its upper bound, then randomly choose another prime number q' with the same size.

3. Then

$$n = p \cdot q,$$

$$\phi(n) = (p-1)(q-1) = 2xp' \cdot 2yq' = 4xy \cdot p'q'.$$

4. Let $\mathbb{S}$ be the the set of all factors of $\phi(n)$ and $\mathbb{F}$ be the set of all factors of $4xy$.

Then $\mathbb{S} = (p'\mathbb{F}) \cup (q'\mathbb{F}) \cup (p'q'\mathbb{F})$. (Note that $p'\mathbb{F}$ means a set, all the elements in which are the product of p' times each of the elements in $\mathbb{F}$.)

5. Choose a limit for the generator (*e.g.*, [2,19]) and let $gen = 2$.

6. Validate the following equation by applying k in incremental order. The first k that makes the equation valid is the the order s of group $\mathbb{G}_s$.

$$gen^k \stackrel{?}{\equiv} 1 \pmod{n},\ k \in \mathbb{S}.$$

7. Check if s is large enough (*e.g.*, our desired group order is $\phi(n)/2$ which means $s = \phi(n)/2$).

8. If s is less than our desired group size, then increase the value of gen and go through Step 6 to 8 until we have a satisfactory subgroup size s and the corresponding subgroup $\mathbb{G}_s$.

9. If the value of gen has reached its upper bound and the desired subgroup size has still not been satisfied, then go back to Step 1 and repeat all the steps until a desired gen of the required minimum order is found.

In Step 2, it is best to choose x and y such that they have few factors in order to reduce the number of factors in $4xy$, so that $\mathbb{F}$ (and ultimately $\mathbb{S}$) will be small (which can speed up Step 6). However, if both x and y are primes, then there are fewest factors in $4xy$.

In Step 5, we set a bound (*i.e.,* [2,19]) for gen to choose from in order to speed up the operation of finding the desired s. Since gen is small, the computation of $gen^k \bmod n$ will take relatively less time.

Note that this algorithm is flexible in the order s of subgroup $\mathbb{G}_s$. In practice, it may take more time to find the gen of a maximum order. However, it is not always necessary for gen to be equal to the maximum order, so either $s < \text{lcm}(p-1, q-1)$ or $s < \phi(n)/2$ may satisfy the security requirement.

3.4.1.3 Authority Key Pairs

We use the basic RSA digital signature algorithm for creating the authority (hospital) key pair in our malleable signature construction, which means there is no padding or hashing in the signing and verification processes. Also, the parameters p and q generated by the DSG algorithm, discussed in the previous section, should be sufficient if they are large enough (such as 1024 bits).

The simple steps of RSA key generation for our protocol are just as follows:

1. Obtain these parameters of our group $\mathbb{Z}_n^*$ after the desired subgroup has been generated: $\phi(n)$, p', q', $4xy$ (keep these parameters safely stored; otherwise, the RSA signature will be broken. Because $s|\phi(n)$, s for $\mathbb{G}_s$ also needs to be safely stored).

2. Randomly select public key: $e \in (1, \phi(n))$, *s.t.* $\gcd(e, \phi(n)) = 1$.

3. Compute private key: $d \in (1, \phi(n))$, *s.t.*, $e \cdot d \equiv 1 \pmod{\phi(n)}$.

4. The private key d will be used by the hospital to sign users' credential and the public key e will be obtained by the verifiers to verify if the credential is issued by the hospital.

Specifically, in Step 2, the factors of $\phi(n)$ consist of all the factors of $4xy$, p' and q'. Thus, if $\gcd(e, 4xy) = 1$, $e \neq q'$, $e \neq p'$, then $\gcd(e, \phi(n)) = 1$. Note that the most common choice of e is 65537 $(2^{16} + 1)$; two other popular choices are 3 and 17. These specific choices of e have only two "1"s in their binary format, which can speed up the operation of the RSA algorithm using the public key (verification) [62]. From the previous section, we can see that none of the factors of $4xy$, p' and q' will be equal to 65537. So we will choose 65537 as the public key e in our proof of concept implementation.

3.4.1.4 Issuing Protocol

Based on Brands' Digital Credential system, Alice's original issued credential public key can be constructed as:

$$h = h_0^{\alpha} g_1^{x_1} g_2^{x_2} \cdots g_{l-1}^{x_{l-1}} g_l^{x_l},$$

where $h_0, g_1, ..., g_l \in \mathbb{G}_s$ are system parameters, $x_1, ..., x_{l-1}, x_l \in \mathbb{Z}_s$ are Alice's attributes and α is Alice's secret value. While $(x_1, ..., x_{l-1})$ are Alice's actual attributes (for example, gender $x_1 = 1$ and age $x_2 = 25$), x_l is the hospital's serial number (a fixed attribute) which is treated as one of Alice's attributes. Note that Alice randomly chooses her secret value α and will never disclose it.

w will be constructed as the same pattern as h:

$$w = h_0^{w_0} g_1^{w_1} g_2^{w_2} \cdots g_l^{w_l},$$

where $w_0, ..., w_l \in \mathbb{G}_s$ are randomly chosen by Alice (or the CA) or are obtained by D-H Exchange.

The detailed issuing protocol is as follows. We can see that all the results will fall into the subgroup $\mathbb{G}_s$ and obviously $\mathbb{Z}_n^*$.

Alice		the CA/Hospital
$h = h_0^{\alpha} g_1^{x_1} g_2^{x_2} \cdots g_{l-1}^{x_{l-1}} \bmod n$		
$w = h_0^{w_0} g_1^{w_1} g_2^{w_2} \cdots g_l^{w_l} \bmod n$		
	$\xrightarrow{h, w}$	
		Add the fixed value x_l:
		$h = h_0^{\alpha} g_1^{x_1} g_2^{x_2} \cdots g_l^{x_l} \bmod n$
		Generate signatures:
		$sig_h = h^d \bmod n$
		$sig_w = w^d \bmod n$
	$\xleftarrow{sig_h, sig_w}$	
Original signatures:		
sig_h, sig_w		

3.4.1.5 Selective Disclosure Multi-Show Protocol

The detailed showing protocol is as follows. We can see that Bob chooses a random integer $a > 0$ and then sends this challenge to Alice. Alice randomly chooses an integer as her secret value $b > 1$, and then uses (a, b) to construct her one-time showing credential public key h_{show} and signature sig_{show}. Let us say that Alice wants to disclose her attribute x_1 (she needs to send x_1 and x_l at the same time) to Bob while keeping her the other attributes hidden. She then needs to construct the response tuple $(r_0, r_1, ..., r_l)$ by using a, b, her secret value α, her remaining hidden attributes $(x_2, ..., x_{l-1})$, and the the tuple $(w_0, ..., w_l)$. The values of x_1 and x_l are denoted as y_1 and y_l in the following showing protocol.

Alice | Bob

$a \in_{\mathcal{R}} \mathbb{Z}_n$

$\xleftarrow{\quad a \quad}$

$b \in_{\mathcal{R}} \mathbb{Z}_n$

Construct:

$$h_{show} = h^a \cdot w^b \bmod n$$

$$sig_{show} = (sig_h)^a \cdot (sig_w)^b \bmod n$$

Compute:

$$r_0 = (a \cdot \alpha + b \cdot w_0)$$

$$r_1 = g_1^{(b \cdot w_1)} \bmod n$$

$$r_2 = (a \cdot x_2 + b \cdot w_2)$$

$$\vdots$$

$$r_{l-1} = (a \cdot x_{l-1} + b \cdot w_{l-1})$$

$$r_l = g_l^{(b \cdot w_l)} \bmod n$$

$\xrightarrow{\quad h_{show}, sig_{show}, y_1, y_l, (r_0, r_1, \cdots r_l) \quad}$

Obtain e, x_l from the Hospital:

$$y_l \stackrel{?}{\equiv} x_l$$

$$(sig_{show})^e \stackrel{?}{\equiv} h_{show}$$

$$h_0^{r_0} \cdot r_1 g_1^{(a \cdot y_1)} \cdot g_2^{r_2} \cdots g_{l-1}^{r_{l-1}} \cdot r_l g_l^{(a \cdot y_l)} \stackrel{?}{\equiv} h_{show}$$

The way we construct the response tuple $(r_0, r_1, ..., r_l)$ is based on DL representation as we mentioned in Chapter 2, which is quite similar to Brands' construction of the tuple (see Section 2.2.4).

In our protocol, if Alice wants to disclose the value of x_j (it works in the same way if Alice wants to disclose more than one attributes), she calculates the response values $\{r_i\}$ $i \in [0, l]$ where

$$r_i = \begin{cases} a \cdot \alpha + b \cdot w_0 & \text{if } i = 0; \\ a \cdot x_i + b \cdot w_i & \text{if } i \in [1, l-1] \text{ and } i \neq j; \\ g_i^{(b \cdot w_i)} & \text{if } i \in \{j, l\}. \end{cases}$$

Note that we can do both $a, b \in_{\mathcal{R}} \mathbb{Z}_s$ and $r_i = a \cdot x_i + b \cdot w_i \bmod s$, if $i \in [1, l-1]$ and $i \neq j$ ($r_0 = a \cdot \alpha + b \cdot w_0 \bmod s$ if $i = 0$), to reduce the computation time for the verification (the exponent part would be smaller). But for maintaining the security of RSA signature scheme (see Section 3.4.1.3), we need to keep hiding the value of s from both Alice and Bob.

3.4.1.6 Verification

At the end of the showing process, Bob obtains the authority verification key (public key) e and the authority serial number x_l from the hospital. Firstly, he needs to verify if the one-time showing credential public key and signature are valid or not.

We can see from the construction of the one-time showing signature:

$$sig_{show} = (sig_h)^a \cdot (sig_w)^b = (h^d)^a \cdot (w^d)^b = (h^a)^d \cdot (w^b)^d = (h^a \cdot w^b)^d = h_{show}^d,$$

that $(sig_{show})^e$ should be equal to its corresponding credential public key h_{show} if both of them are valid.

Secondly, Bob needs to verify if the value (y_1) of the disclosed attribute (x_1) is valid or not. Bob constructs a credential public key (labeled as h_{verif}) by using the tuple $(r_0, r_1, ..., r_l)$ and the disclosed values y_1, y_l from Alice, and the system parameters $(h_0, g_1, ..., g_l)$:

$$h_{verif} = h_0^{r_0} \cdot r_1 g_1^{(a \cdot y_1)} \cdot g_2^{r_2} \cdots g_{l-1}^{r_{l-1}} \cdot r_l g_l^{(a \cdot y_l)}.$$

If the value of x_1 has not been compromised (*i.e.,* $y_1 = x_1$), then

$$h_{verif} = h_0^{a\alpha + bw_0} g_1^{ax_1 + bw_1} g_2^{ax_2 + bw_2} \cdots g_{l-1}^{ax_{l-1} + bw_{l-1}} g_l^{ax_l + bw_l} \equiv h^a \cdot w^b \equiv h_{show}.$$

3.4.2 Example of a Malleable Signature

In this section, we will present a simple example to show how this malleable signature scheme works. Given all the requirements for choosing the parameters n, p, q we described before, the smallest sizes of these parameters are 10, 5, 5 bits respectively. We choose the parameters with the smallest sizes because they are the easiest to compute by hand; hence allow us to give the following example. Note that all the results will fall into the multiplicative group $\mathbb{Z}_n^*$ (more specifically, its cyclic subgroup $\mathbb{G}_s$).

Initialization:

There are just two primes with the size of 5 bits, 23 and 31, such that

$$p = 23 \ (10111), \ q = 31 \ (11111);$$

$$n = p \times q = 713 \ (10110 \ 01001).$$

Then

$$\phi(n) = (p-1)(q-1) = (23-1)(31-1) = 22 \times 30 = 660.$$

The hospital chooses the generator gen as

$$gen = 3.$$

By listing all the numbers in $\mathbb{Z}_n^*$ generated by $gen = 3$, the hospital has the subgroup $\mathbb{G}_s$ with the order of $s = 330$.

Then the hospital chooses the system parameters by raising gen to the power of random positive integers and lets them be widely known and used by all users/verifiers in our credential system,

$$g_1 = 3^1 = 3, \ g_2 = 3^4 = 81, \ g_3 = 3^3 = 27, \ h_0 = 3^7 \equiv 48 \pmod{713}.$$

For its authority key pair, let the signing (issuing) key be

$$e = 7, \text{ where } e \in (1, 660) \text{ and } \gcd(e, 660) = \gcd(7, 660) = 1.$$

Then compute for the verification key:

$$d = 283, \text{ where } d \in (1, 660) \text{ and } e \cdot d = 7 \times 283 \equiv 1 \pmod{660}.$$

Issuing:

Choose the values of attributes and the secret α, and obtain the serial number (*e.g.*, 123) of the hospital. Let

$$x_1 = 5,\ x_2 = 11,\ x_3 = 123,\ \alpha = 7.$$

Then construct the original credential public key h:

$$h = g_1^{x_1} \cdot g_2^{x_2} \cdot g_3^{x_3} \cdot h_0^{\alpha} = 3^5 \times 81^{11} \times 27^{123} \times 48^7 \equiv 611 \pmod{713}.$$

Alice (or the hospital) uses random numbers to construct the parameter w:

$$w_1 = 13,\ w_2 = 7,\ w_3 = 5,\ w_0 = 10;$$

$$w = g_1^{w_1} \cdot g_2^{w_2} \cdot g_3^{w_3} \cdot h_0^{w_0} = 3^{13} \times 81^7 \times 27^5 \times 48^{10} \equiv 450 \pmod{713}.$$

The hospital signs both h and w:

$$sig_h = h^d = 611^{283} \equiv 416 \pmod{713};$$

$$sig_w = w^d = 450^{283} \equiv 531 \pmod{713}.$$

We can see that these signatures are valid:

$$(sig_h)^e = 416^7 \equiv 611 \equiv h \pmod{713};$$

$$(sig_w)^e = 531^7 \equiv 450 \equiv w \pmod{713}.$$

Showing:

Alice constructs a one-time showing credential public key h_{show} and the signature sig_{show}, and then shows them to Bob.

Let

$$a = 9,\ b = 3;$$

$$h_{show} = h^a \cdot w^b = 611^9 \times 450^3 \equiv 542 \pmod{713};$$

$$sig_{show} = (sig_h)^a \cdot (sig_w)^b = 416^9 \times 531^3 \equiv 554 \pmod{713}.$$

Verification:

Bob uses the verification key e obtained from the authority to verify whether the received one-time showing signature is valid or not:

$$(sig_{show})^e = 554^7 \equiv 542 \equiv h_{show} \pmod{713}$$

Obviously, it is valid.

If Alice wants to continue showing the same Digital Credential to David, she just needs to choose another pair of a and b, and repeats the process above.

Let

$$a = 4,\ b = 5.$$

One-time credential public key h_{show}:

$$h_{show} = h^a \cdot w^b = 611^4 \times 450^5 \equiv 578 \pmod{713}.$$

One-time signature sig_{show}:

$$sig_{show} = (sig_h)^a \cdot (sig_w)^b = 416^4 \times 531^5 \equiv 351 \pmod{713}.$$

Verify:

$$(sig_{show})^e = 351^7 \equiv 578 \equiv h_{show} \pmod{713}.$$

We can see that different h_{show} and sig_{show} constructed from the same original h will be considered as valid during the showing process when communicating with others.

Selective Disclosure:

Specifically, we apply selective disclosure of the specific revealed attributes, along with the serial number, in the showing process. Here is an example when $a = 9,\ b = 3$. Alice wants to reveal the value (5) of x_1 and the value (123) of x_3 to Bob.

She computes:

$$r_1 = g_1^{(b \cdot w_1)} = 3^{(3\times 13)} \equiv 246 \pmod{713},$$

$$r_2 = (a \cdot x_2 + b \cdot w_2) = (9 \times 11 + 3 \times 7 = 120,$$

$$r_3 = g_3^{(b \cdot w_3)} = 27^{(3\times 5)} \equiv 371 \pmod{713},$$

$$r_0 = (a \cdot \alpha + b \cdot w_0) = (9 \times 7 + 3 \times 10) = 93.$$

Alice sends her responses (her signature): $r_0,\ r_1,\ r_2,\ r_3$ along with 5 and 123 to Bob.

Bob verifies if the values of disclosed attributes, 5 and 123, are valid or not:

$$r_1 \cdot g_1^{(a\cdot x_1)} \cdot g_2^{r_2} \cdot r_3 \cdot g_3^{(a\cdot x_3)} \cdot h_0^{r_0} = 246 \times 3^{9\times 5} \times 81^{120} \times 371 \times 27^{9\times 123} \times 48^{93} \equiv 542 \equiv h_{show} \pmod{713}$$

Obviously, they are both valid.

(Optional) Diffie-Hellman Exchange:

If Alice chooses to collaborate with the hospital to obtain $w_0,\ w_1, \ldots, w_l$, she needs to do D-H Exchange for every single w_i. As we mentioned before, this process is optional in our protocol.

Let's say the system parameters are:

$$g = 3,\ p = n = 713.$$

Alice and the hospital collaborate to get the value of w_1.

Alice randomly chooses a. Then she computes A and sends it to the hospital:

$$a = 5,$$

$$A = g^a \bmod p = 3^5 \bmod 713 = 243.$$

The hospital randomly chooses b. Then the authority computes B and sends it to Alice:

$$b = 6,$$

$$B = g^b \bmod p = 3^6 \bmod 713 = 16.$$

Alice receives B and computes her value of w_1:

$$w_A = B^a \bmod p = 16^5 \bmod 713 = 466.$$

The hospital receives A and computes the value of w_1:

$$w_B = A^b \bmod p = 243^6 \bmod 713 = 466.$$

Obviously, we can see that the value of the parameters both parties get are the same:

$$w_1 = w_A = w_B.$$

3.4.3 Storage of Information

It is necessary and important to avoid leaking information during the communication between different parties, and those parameters related to the group information, including $\phi(n)$, s, $p\&q$, $p'\&q'$, $x\&y$, and $4xy$, should be safely stored. Eavesdroppers can observe all traffic on the channel, but should not be able to learn any of these parameters. In such a way, the multi-show unlinkability of the Digital Credentials used in our system will be guaranteed.

The DSG algorithm will run on the authority (*i.e.*, the hospital) side and the results will be stored on this side. Thus, the system parameters (*i.e.*, $g_1, \cdots g_l, h_0, n$) will be generated and distributed by the authorities (*i.e.*, the hospital). The authorities also need to generate an asymmetric RSA key pair on their side. The private key will be used for them to issue the Digital Credentials and the public key will be obtained from the authorities and used by the verifiers.

The original credential public keys and signatures will be stored in Alice's own devices. If she loses or damages these devices, then she will need to obtain new credentials from the appropriate authorities. The program for generating one-time showing credential public keys and signatures will run on her devices as well.

3.5 Security and Privacy of the Proposed Technique

Our proposed protocol mitigates both the issuer-linking attack and the verifier-linking attack mentioned in Section 3.1. Alice will never show her original Digital Credential public key h and signature sig_h to any verifiers since she will modify them with every showing execution. Thus, the authority (the hospital) cannot learn which credentials refer to Alice, and there is no way for Bob, David or the authority to collude to trace the transactions. Alice ensures this because she never shows the same credential public key & signature more than once. (Note that Alice's original Digital Credential public key and signature are used as the basis for every new showing transaction; in this sense they are multi-show, but each randomized credential public key & signature pair is used only in a single showing transaction. Thus, Alice derives single-show values for each transaction whenever she wishes, rather than obtaining a single-show value from the authority at issuing time.)

Based on the structures of our protocol and the three parties (Alice, Bob, and the hospital) we mentioned in Section 3.2, we will analyze the related security issues from the following perspectives:

- The reason why we need a large enough group for calculations.
- The reason why we need n for $\mathbb{Z}_n^*$ to be a Blum integer.
- The reason why Alice has two secret parameters α and b.
- The reason why we have a and b when we create the one-time showing credential public key and signature based on the original ones.
- The reason why we add a fixed value as an attribute to Alice's Digital Credential public key h.
- The way to avoid attacks on our signatures when we are just using a basic RSA signing algorithm.

3.5.1 Large Enough Subgroup

It is necessary that the order of $\mathbb{G}_s < \mathbb{Z}_n^*$ should be large enough (*e.g.*, $|s| \approx |n|$ =2048 bits). The main reasons are as follows:

1. If the set $\mathbb{G}_s$ is large, then the credential public key and signature space would be large enough to resist the brute force attack (accomplished by exhaustively searching for all the possible values). Equivalently, computing discrete logarithm is intractable.

2. The authority's private key d should be chosen from a large enough set, so that a direct search for it is also unfeasible.

3. There would be enough choices for p and q which are used to construct $\mathbb{Z}_n^*$. Thus, apart from the security issues, it is easier for the authority to find the proper values for p, q, n, etc.

3.5.2 Blum Integer n for $\mathbb{Z}_n$

As Brands' Digital Credential is in the space $\mathbb{G}_q$ (is a subgroup of $\mathbb{Z}_p^*$ where $q|p-1$) where q is a prime and the RSA signature scheme is in the space $\mathbb{Z}_n^*$ where n is product of two prime numbers, we need to adjust these two spaces for the use of our protocols in this thesis. The set $\mathbb{Z}_n^*$ where n is a Blum integer (a composite which is the product of two distinct primes) is used in the scheme proposed by this thesis. It is obvious that the RSA cryptosystem would work smoothly in the space $\mathbb{Z}_n^*$.

From the description of the Discrete Logarithm Problem (Section 2.1.3), and the Diffie-Hellman Problem (Section 2.1.4) in Chapter 2, we can see that, under the intractability of both the integer factorization problem and the discrete logarithm problem, solving the discrete logarithm problem is more difficult in the set $\mathbb{Z}_n^*$ (where n is a composite) than in the set $\mathbb{Z}_p^*$ (where p is a prime). Furthermore, if n is a Blum integer, then all its bits are individually hard for the factorization problem (Section 2.1.6).

In conclusion, the DLP (as well as the DL Representation) is still believed to be intractable in the set $\mathbb{Z}_n^*$ where n is a Blum integer (and the desired subgroup $\mathbb{G}_s$ has a large enough order). Note that the D-H exchange is in the space $\mathbb{Z}_p^*$ where p is prime. Even though this key agreement scheme is not necessary in our protocol, its space problem has also be solved. Thus, it is reasonable to apply the space sets $\mathbb{Z}_n^*$ and $\mathbb{G}_s$ to underlie the methods described in this thesis.

3.5.3 Secret Parameters

There are just two parameters that Alice chooses/generates herself and never shares with anyone:

1. b from $h_{show} = h^a \cdot w^b$: this keeps others from being able to link the one-time showing credential and Alice's original credential. We will discuss the detailed functionality of b to Alice in Section 3.5.4. Note that a new random value for b will be chosen with every showing process.

2. α from $h = h_0^{\alpha} g_1^{x_1} \cdots g_l^{x_l}$: this may be randomly chosen by Alice, or derived from Alice's biometric or her passphrase. If Alice is the only entity that knows α, then no other entities can prove knowledge/possession of the attributes contained in h (this property was proven in Brands' original scheme). Equally, Alice cannot deny that she has the corresponding attributes bounded in her Digital Credential.

3.5.4 Functionality of a and b

When Alice creates h_{show} and sig_{show} during the showing protocol, she needs to communicate with Bob about the parameter a, but she will never share the parameter b.

Functionality of a : The randomness from Bob prevents Alice from forging signatures, particularly sig_{show} (see details in Section 3.5.6).

functionality of b : The randomness from Alice prevents others from relating h_{show} and sig_{show} to her original h and sig_h.

Specifically, from the structure of $h_{show} = h^a \cdot w^b$ and $sig_{show} = (sig_h)^a \cdot (sig_w)^b$, we can see that:

- Bob knows the value of a, h_{show} and sig_{show}.
- The hospital knows the value of h, sig_h, w and sig_w. (Note that, from the issuing protocol, the hospital knows the specific pair, h and w, that belongs to Alice.)

If either Bob or the hospital learns the value of b (or, in the degenerate case, if $b = 1$), they can collude to know the specific h_{show} and sig_{show} that correspond to the original h and sig_h, which means that Alice's one-time showing credential public key and signature will be learned and traced back to her original credential public key and signature issued by the hospital, or even herself.

3.5.5 Functionality of the Fixed Attribute

It is necessary to bind a fixed-value attribute into Alice's original credential to prevent Alice from illegally manipulating her issued credential (and the authority's signature on this credential).

As we presented in previous sections, Alice's original issued credential public key is as follows:

$$h = h_0^{\alpha} g_1^{x_1} g_2^{x_2} \cdots g_l^{x_l}.$$

Suppose that she raises h to the power m:

$$h^m = h_0^{\alpha m} g_1^{x_1 m} g_2^{x_2 m} \cdots g_l^{x_l m}.$$

Now, Alice has her new valid-looking original credential public key:

$$h' = h_0^{\alpha'} g_1^{x'_1} g_2^{x'_2} \cdots g_l^{x'_l},$$

where $h' = h^m,\ \alpha' = \alpha m,\ x'_1 = x_1 m,\ x'_2 = x_2 m, ..., x'_l = x_l m.$

She can compute a corresponding signature as follows:

$$sig'_h = sig_h^m = (h^m)^d = (h_0^{\alpha m} g_1^{x_1 m} g_2^{x_2 m} \cdots g_l^{x_l m})^d = (h_0^{\alpha'} g_1^{x'_1} g_2^{x'_2} \cdots g_l^{x'_l})^d.$$

Alice is able to obtain any value of the attributes by manipulating the value of m appropriately. The corresponding sig'_h would appear to be perfectly valid to a verifier. The fake pair of original credential public key and signature, h' and sig'_h, can then be easily modified to be a valid-looking one-time showing pair, h'_{show} and sig'_{show}, by following the steps in Section 3.4.1.5.

To avoid this attack, the value of x_l is fixed (it is the authority's serial number). In this case, h'_{show} would be immediately detected as being compromised at verification time. To be specific, there are four possible situations during the selective disclosure process:

1. Alice discloses the values of real attribute x_1, the real serial number x_l, and the real one-time showing credential public key h_{show}, then the verification,

   $$h_0^{r_0} \cdot r_1 \cdot g_1^{(a \cdot x_1)} \cdot g_2^{r_2} \cdots r_l \cdot g_l^{(a \cdot x_l)} = h_{show},$$

   means that the disclosed attributes are valid (as demonstrated in Section 3.4.1.5).

2. Alice discloses the values of fake attribute x'_1, the corresponding fake serial number x'_l, and the fake one-time showing credential public key h'_{show}, then the verification,

   $$h_0^{r_0} \cdot r_1 \cdot g_1^{(a \cdot x'_1)} \cdot g_2^{r_2} \cdots r_l \cdot g_l^{(a \cdot x'_l)} = h'_{show},$$

   means that this fake credential public key, h'_{show}, and the disclosed attributes appear valid. But it is easy for Bob to see that the serial number is fake (because Bob knows the real serial number of the corresponding authority); thus, x'_1 is fake as well.

3. Alice discloses the values of fake attribute x'_1, the real serial number x_l, and the fake one-time showing credential public key h'_{show} (each of the remaining attributes used for computing the response tuple $(r_0, r_1, ..., r_l)$ can either be fake or genuine), then the verification,

   $$h_0^{r_0} \cdot r_1 \cdot g_1^{(a \cdot x'_1)} \cdot g_2^{r_2} \cdots r_l \cdot g_l^{(a \cdot x_l)} \neq h'_{show},$$

   means that the disclosed attributes are invalid.

4. Alice discloses the values of fake attribute x'_1, the real serial number x_l, and the real one-time showing credential public key h_{show} (each of the remaining attributes used for computing the response tuple $(r_0, r_1, ..., r_l)$ can either be fake or genuine), then the verification,

$$h_0^{r_0} \cdot r_1 \cdot g_1^{(a \cdot x'_1)} \cdot g_2^{r_2} \cdots r_l \cdot g_l^{(a \cdot x_l)} \neq h_{show},$$

means that the disclosed attributes are invalid.

This analysis also holds in the case that Alice forges a new credential by multiplying two (or more) existing credentials (which are issued to herself or other users by the same authority).

3.5.6 Attacks on Basic RSA signatures

The hospital signs the original credential by applying the basic RSA signature algorithm:

$$sig = h^d,$$

where d is the hospital's secret key.

Based on the properties of discrete logarithm, it is infeasible to figure out d with the knowledge of h and sig. Thus, it is infeasible for Alice to forge a valid signature on a fake credential public key.

However, there are basically two kinds of attacks against the basic RSA signatures, which can usually be avoided by "padding" or "hashing" the message (while neither of them is used in our issuing process).

1. No message attack:

   Alice chooses arbitrary $sig' \in \mathbb{Z}_n^*$ and computes $h' = sig'^e$, then she has a valid set of showing credential public key and signature (h', sig'). However, because of the discrete logarithm problem, it is infeasible for $h_0^{r_0} \cdot r_1 g_1^{(a \cdot x_1)} \cdot g_2^{r_2} \cdots g_{l-1}^{r_{l-1}} \cdot r_l g_l^{(a \cdot x_l)} \equiv h_{show} \equiv h'$ to hold true by constructing the tuple $(r_0, r_1, \cdots r_l)$ using the challenge from Bob and the value of attribute(s) that Alice wants to show.

2. Forging fake showing signature sig' on fake showing credential public key h' by manipulating existing valid signatures and credential public keys.

   Alice has two valid pairs of credential public key and signature, (h_1, sig_1) and (h_2, sig_2). She can do computation either $h_3 = h_1/h_2$ or $h_3 = h_1 \cdot h_2$ to create the fake credential public key h_3 and then forges a signature sig_3 on h_3. Since the forging processes are similar, we just consider $h_3 = h_1 \cdot h_2$ here:

   $$h_1, h_2, sig_1, sig_2 \in \mathbb{Z}_n^*,$$

   $$h_1 \equiv sig_1^e, h_2 \equiv sig_2^e.$$

   Let

   $$h_3 = h_1 \cdot h_2 \bmod n.$$

   Compute

   $$sig_3 = sig_1 \cdot sig_2 \bmod n.$$

   Then

   $$sig_3^e \equiv (sig_1 \cdot sig_2)^e = sig_1^e \cdot sig_2^e \equiv h_1 \cdot h_2 \equiv h_3.$$

   We can see that sig_3 is a valid signature on the credential public key h_3. In our case, the pairs of Digital Credential public key and signature, (h_1, sig_1) and (h_2, sig_2), might be issued from the same authority (only the hospital) or different authorities (the hospital and the bank):

   - Issued from the same authority:

     If $h_3 = h_1 \cdot h_2$, the fixed attribute (the hospital's serial number) should be found to have been changed. Thus, this kind of attack cannot be applied without detection.

   - Issued from different authorities:

     The authority key pair (d, e) obtained from different authorities will be different. The hospital issues (h_1, sig_1) with public key e_1 while the bank issues (h_2, sig_2) with e_2:

     $$h_1 \equiv sig_1^{e_1}, h_2 \equiv sig_2^{e_2}.$$

Let

$$h_3 = h_1 \cdot h_2 \bmod n.$$

Compute

$$sig_3 = sig_1 \cdot sig_2 \bmod n.$$

Then

$$sig_3^e \equiv (sig_1 \cdot sig_2)^e = sig_1^e \cdot sig_2^e \neq sig_1^{e_1} \cdot sig_2^{e_2} \equiv h_1 \cdot h_2 \equiv h_3.$$

So the forged signature sig_3 will be found being invalid.

3.5.7 Response Construction

From the construction of $\{r_i\}$ $i \in [0, l]$:

$$r_i = \begin{cases} a \cdot \alpha + b \cdot w_0 & \text{if } i = 0 \\ a \cdot x_i + b \cdot w_i & \text{if } i \in [1, l-1] \text{ and } i \neq j \\ g_i^{(b \cdot w_i)} & \text{if } i \in \{j, l\} \end{cases},$$

we can see that r_j and r_l are calculated differently from the rest of $\{r_i\}$.

If r_j and r_l are calculated the same way as the other values in $\{r_i\}$:

$$r_i = b \cdot w_i, \ i \in \{j, l\},$$

then Alice's secret parameter b will be easily figured out (if Bob colludes with the authority) as $b = r_i / w_i$, where w_i is available to the authority.

3.5.8 Conclusion

Under the assumption of the intractability of the DLP, we have analyzed (Section 3.5.3 - 3.5.7) and found that our protocol can successfully prevent Alice from forging her issued credential public keys, issued signatures, showing credential public keys and showing

signatures. As well, others cannot illegally use Alice's Digital Credential public keys and signatures (also known as replay attacks), or trace back to Alice's original credentials (or Alice herself) by using her showing credentials (multi-show unlinkability).

In summary:

- The fixed attribute prevents Alice from forging her issued credentials.
- The parameter a prevents Alice from forging her signatures, especially the showing signatures.
- The parameter b avoids others from linking Alice's showing credentials to her original ones.
- The parameter α ensures that no one knows the Alice's entire attribute tuple, so that only she can legally use her credentials, including the original and one-time showing ones, and denying her showing activities later has also been prevented.
- The RSA attacks against the basic RSA signing algorithm are prevented because of the intractability of the DLP and the fixed attribute.
- Replay attacks are not possible by Bob or Eve (an attacker performs a man-in-the-middle attack), because both Alice and Bob contribute randomness in the showing protocol (*i.e.,* challenge a from Bob and Alice's secret value b) and only Alice knows α. (Note that Brands allows Alice to sign a nonce [12] sent by Bob to avoid replay attacks in his Digital Credential system. In this thesis, the parameter a also performs the function of a nonce.)
- The issues of *linkability* and *malleability* in self-blindable credentials, referred to in [43] (which have been formally described in the Appendix of [56]), can be prevented as follows:
  - The unlinkability (which is termed as multi-show unlinkability in this thesis) comes from the randomization of the one-time credentials which will be used in

the showing protocol, as the one-time credentials would not be linkable to each other if they have been completely randomized. Both Alice and Bob contribute to the randomization (*i.e.,* the parameters a and b) of the showing credentials and we only need one of them to be honest to achieve unlinkability.

– The unmalleability (unforgeability) also comes from the randomization as well as the fixed attribute encoded in Alice's original issued credentials. Note that our system just takes the malleability of the basic RSA digital signature scheme to transform the valid signature on a Digital Credential (the original issued one) into another valid signature on the same Digital Credential of its showing version, but the Digital Credentials by applying our technique is unmalleable.

3.6 Possible Revocation Method

We can encode another serial number as an attribute in Alice's Digital Credential. Before every showing transaction, Alice needs to do a zero-knowledge proof of knowledge to Bob that this serial number is in Bob's whitelist but not in the blacklist (which means this credential has not been revoked). Alice should not reveal the value of this serial number (*i.e.,* Bob should not learn the value); otherwise, her one-time showing credentials would be linked to her original credential, because this serial number may be unique, which refers to Alice's original credential.

3.7 Chapter Summery

This chapter started with analyzing the limitations for Brands' Digital Credential system. Our proposed technique was presented in both high-level overview and low-level detailed view. The detailed descriptions of our proposed technique provide explanations of the related mathematics followed by an easy-to-understand example. Using the specific authorities (like the hospital) instead of the normal CAs could be a reasonable way to simplify

the Digital Credential system. Finally, we analyzed the security and privacy issues related to our approaches and provided a possible solution for revocation of our Digital Credentials.

Chapter 4

Proof of Concept

4.1 Architecture

To prove that our proposed protocol could work just as described in the last chapter, we have a Python script for the proof of concept.

The demo script contains these parts:

- Subgroup generation.
- Authority key pairs generation.
- Original credential public key h creation.
- D-H Exchange for obtaining the parameter w.
- Getting signatures for h and w from the hospital.
- One-time showing credential public key h_{show} and signature sig_{show} creation.
- Verification of the credentials and the disclosed attributes.

Basically, the program can be roughly separated into two parts:

1. Authority side: the hospital finds a desired subgroup, and generates system parameters and its authority key pairs.

2. User side: Alice creates her multi-show Digital Credentials based on her issued original credential and then shows her attributes to verifiers. Verifiers (*e.g.*, Bob) receive the revealed attribute(s) and verify them (we consider the verifier as another user in this proof of concept implementation for the sake of convenience).

Note that, since this implementation is just a proof of concept, all the programming (both the authority side and the user side) will run on a single (*i.e.*, the same) computer. The computer environment is shown in Table 4.1), but only 1 CPU core is occupied (by default) in our Python implementation.

The parameters generated by the authority side will be stored in a file, which will be kept secretly or be obtained (by Alice or Bob) for later use. Thus, there is no real data communication between different sides, and any time consumption related to it (including any data transfer between Alice and the hospital, Alice and Bob, and the hospital and Bob) has not been considered in this implementation for proof of concept.

Table 4.1: Computer Environment

CPU	Intel i5-6200U 2.3 GHz-2.4GHz
Number of CPU Cores	4
Operating System	Windows 10 Home 64bits
RAM	8GB

4.2 Authority Side

We have generated a subgroup for the authority side to use in later protocols (some useful parameters of this Subgroup have been shown in Table 4.2 and 4.3). After successfully generating a desired subgroup, the authority needs to safely store these useful parameters

of this group for further use. Every parameter (except n) in Table 4.2 and 4.3 should be kept as secret.

However, the authority will make the system parameters (*i.e.*, $h_0, g_1, ..., g_l$) available to all the users (*e.g.*, Alice and Bob) in this system. Each $g_i(h_0)$ can be generated as $g_i = gen^r \bmod n$, where r is a random positive integer from $\mathbb{Z}_s$. In our implementation, gen is raised to a random exponent less than 100 for the purpose of simplifying calculations (the system parameters are shown in Table 4.4). The current subgroup, system parameters and the issuing key pairs would be revoked by the authority when they are not secure anymore. The user's issued credential would also be revoked when s/he no longer possesses the issued attributes.

The authority will also generate a pair of authority keys (e, d), where e is the public key and d is the secret key (see Section 3.4.1.3). It will use its secret key to sign Alice's h and w. The keys and the time consumption for key generation are shown in Table 4.5. The time consumption on issuing the credential (*i.e.*, signing h and w) is shown in Table 4.9 corresponding to a varying number of attributes contained in h.

Table 4.2: Subgroup 1 Infomation (1)

p'	247391482843335719862072005127333994336729439769844947375102071003704386271616921735018423138135336696881559377936863770967758625817204774996723911384529853250334792810742429998754291238916626134111281978507428621739352326458774756719989876185652137508354827012452579216521070753526887358434332692285967273 1
q'	145725812610997024523887559130205832529820106216227084316077576750607382180781813255308487573954879053642807955870665671504410964831281359205391007174990377925306613113027569491945852370268848961418697148482620309460201934113099454985810201524524683505141746058571551230800643316971465899988703236597670129 9
n	849369213946261799758903770141623763002306367958846244963853426082849816408759629659604400198706790821294265552066771151450699499694176923840815074511267183706067480178525242255091835845113074265270339768165873893219215010458472173340082458268149702677184502468382678768167877585841544605507842936382480037167975754197077310269001713954778793129446465171532882919532774127395758770303445027223062596323985955532721211786824196695812604534749910004170717563222306594044234175976578004715056413949483118265328872605544629770130158610831947578294800849627151026219111737270252680648612069316432503628409117384977579688 1
$4xy$	2356

Table 4.3: Subgroup 1 Infomation (2)

$phi(n)$	84936921394626179975890377014162376300230636795884624496385342608284 98164087596296596044001987067908212942655520667711514506994996941769 23840815074511267183706067480178525242255091835845113074265270339768 16587389321921501045847217334008245826814970267718450246838267876816 78775858415446055078429363824800371495398774671487373937619490938673 39727801899874872695691682085670717334396773529579463236287489640659 72536786216124210176576429693415668428356096648576076873916250203616 06563562617504999139826895611409342163398829575022502903863406138149 53060051839423582840055165618494111171495030179898208604874409055275 2564
s	42468460697313089987945188507081188150115318397942312248192671304142 49082043798148298022000993533954106471327760333855757253497498470884 61920407537255633591853033740089262621127545917922556537132635169884 08293694660960750522923608667004122913407485133859225123419133938408 39387929207723027539214681912400185747699387335743686968809745469336 69863900949937436347845841042835358667198386764789731618143744820329 86268393108062105088288214846707834214178048324288038436958125101808 03281781308752499569913447805704671081699414787511251451931703069074 76530025919711791420027582809247055585747515089949104302437204527637 6282
gen	2

Table 4.4: System Parameters

h_0	18889465931478580854784
g_1	72057594037927936
g_2	137438953472
g_3	4294967296
g_4	262144
g_5	2251799813685248
g_6	147573952589676412928
g_7	32
g_8	1152921504606846976
g_9	4398046511104

Table 4.5: The Authority's Key Pair

Private Key d	626921235122488396474319118559909154946222224655893150550627611576753500604863352999436099467660155216114065452341820969973063870770807640522937388030174214221181961619784267568023519162238960303857700926278099666397501986677871651753662949994738934763621251169615260388064524504629644890706484678298208749821241321585034192501158837962031292623296173194488867490024528589187933789693241166781743578966635452231490907370275786024922049156903143092431674196495196091055551993457122387680999388625342701712636684786410869559229997608936302462915158221674590360208320826228336106560259070892700187925062843733909559942 9
Public Key e	65537
Time (secs)	0.008534

4.2.1 Results and Analysis

Table 4.6: Time Consumed for Subgroup Generation

Subgroup	Attempts	Size of p, q (bits)	Size of n (bits)	gen	Time for Generating p and q (secs)	Whole Time (secs)
1	1	1024	2048	2	6.067157	6.703348
2	1	1024	2047	5	14.156274	16.634017
3	1	1024	2047	7	10.410402	13.969227
4	1	1024	2048	2	9.586145	10.195111
5	1	1024	2048	2	8.644857	9.284206
6	1	1024	2047	2	8.158881	8.771338
7	1	1024	2047	2	11.12562	11.759679
8	1	1024	2047	5	68.742346	70.874102
9	2	1024	2048	7	24.313717	35.934925
10	1	1024	2047	2	11.84783	12.472403
Average	1.1	N/A	N/A	N/A	17.3053229	19.6598356

We choose $|p'| = |q'| = 1018$ bits, $|x| = |y| = 6$ bits where all of them are primes, and keep strictly $|p| = |q| = 1024$ bits. Thus $|n|$ is 2047 or 2048 bits. Every calculation has been performed modulo n to keep the number size smaller and eventually the computation faster.

Our implementation can find a desired subgroup $\mathbb{G}_s < \mathbb{Z}_n^*$, where $|n| \approx 2048$ bits and $s = \phi(n)/2$, in around 20 seconds on average (details for ten generated subgroups with the chosen limit of [2,19] for gen are shown in Table 4.6), which is acceptable. Because this process only happens once at the beginning (before the authority issues a credential to Alice), and there is no need to do this with every transaction.

We can see from the Table 4.6 that the time consumed for generating primes dominates the whole time consumed for DSG. As we discussed in Section 3.4.1.2, during the process of the DSG, we need to randomly choose a prime number and check if a given number is a prime. For the former one, there is no straightforward way to generate a prime number with a given number of bits (for example, $k = 1018$ bits). We need to start by

randomly generating a k-bit odd number and then test whether it is a prime or not. Our implementation uses the Miller-Rabin algorithm for the primality testing.

In this case, the time consumed for the DSG depends on how fast it is to randomly choose a prime number with a given size (*e.g.*, p' and q'). As we discussed in Section 2.1.1.1, if we want to find a prime with 1018 bits, we need to test, on average, $0.5\ln(2^{1018}) \approx 353$ numbers randomly generated. Therefore, the speed of choosing a proper prime number depends on how lucky we are in choosing the random k-bit odd number for the primality testing later, which is not under our control.

The "Attempts" shown in the table above represents how many times the DSG algorithm performs (goes through from Step 1 to 8 described in 3.4.1.2). The DSG can be separated into two main parts: one is for generating the prime numbers p and q (Step 1 to 2), and another one is for finding a desired *gen* (Step 5 to 8). The "Time for Generating p and q" shown in Table 4.6 and 4.7 are the sum of time consumed in each attempt. The same is true for the "Time for Finding a Desired *gen*" in Table 4.7.

Furthermore, we run the DSG algorithm for 50 times for each "Chosen Limit for *gen*" (in Step 5). The average time consumption is shown in Table 4.7 and Figure 4.1. We can see that the way we choose the limit for the generator has significant influence on the time consumed for finding the desired generator (and consequently, the whole time consumption), but has little influence on generating the prime numbers.

Specifically, if the desired *gen* has not been found in the first several attempts (except the last one), the time required would be more when the upper bound for choosing *gen* is too big (*e.g.*, 199, 999) (*i.e.*, more *gen* that need to be checked whether their order is large enough or not). However, if the upper bound is too small (*e.g.*, 5), it would take more attempts to find the desired *gen*, which will increase the time consumed for generating prime numbers (*i.e.*, more pairs of p and q are generated).

Besides, the actual desired *gen* are usually small numbers less then 10. So it is reasonable to choose the upper bound for *gen* to be just a little bit bigger than 10 (*e.g.*, 19).

Table 4.7: Average Time Consumption with Different Chosen Limit for *gen*

Chosen Limit for *gen*	Average Attempts	Actual Limit of *gen*	Average Time for Generating p and q (secs)	Average Time for Finding a Desired *gen* (secs)
[2,5]	1.56	[2,5]	29.39007858	2.46552778
[2,11]	1.38	[2,11]	24.31211326	3.36911468
[2,19]	1.5	[2,10]	18.43386952	5.39648638
[2,199]	1.26	[2,10]	20.58670354	26.08188316
[2,999]	1.26	[2,11]	20.036508	82.86357576

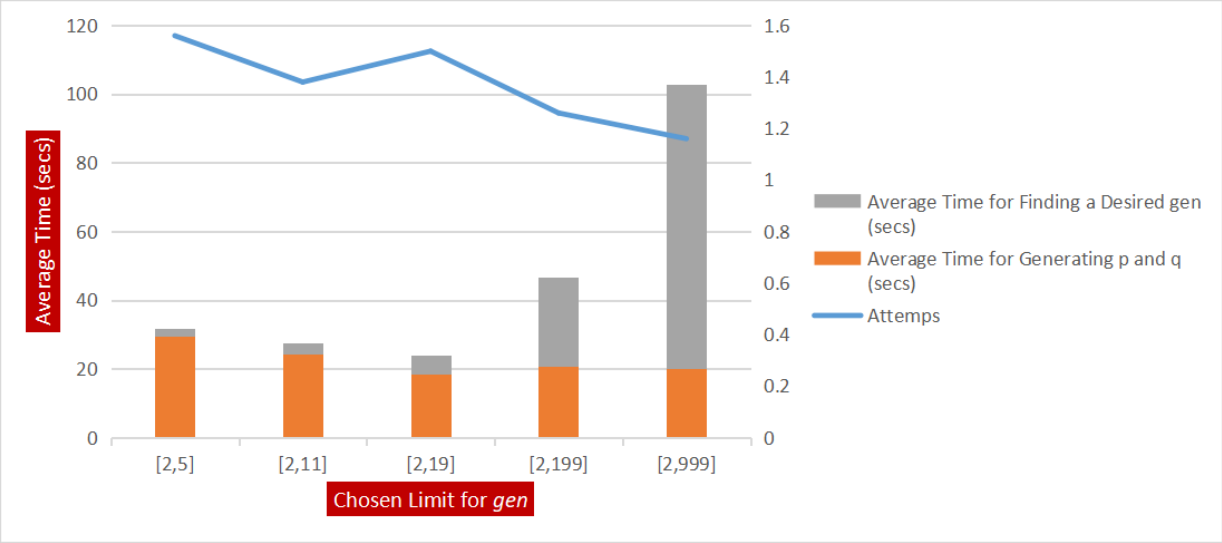

Figure 4.1: Average Time Consumption with Different Chosen Limit for *gen*

4.3 User Side

We separate the "issuing protocol" in Chapter 3 into three parts in this chapter, "Original Credential Creation", "D-H Exchange for w", and "Issuance" (as shown in Table 4.9).

Original Credential Creation : The user (Alice) needs to input the number of attributes, the value to each of them and the fixed attribute (the authority's serial number), along with system parameters (n for $\mathbb{Z}_n^*$ and $g_1 \cdots g_l, h_0$), to create her original credential

public key h. As we mentioned in the description of the issuing protocol in Section 3.2, the authority and Alice need to work together on the creation of h, or the authority can supervise Alice during this process. Then Alice will obtain her other important parameter, w, by D-H Exchange with the authority.

In our implementation, we randomly choose nine numbers less than 100 as Alice's attributes and secret value α, and choose 123 as the authority's serial number (shown in Table 4.8). In practice, α can be randomly chosen by Alice from $\mathbb{Z}_n^*$, the values of attributes map to meaningful information (see Section 2.2.1), and the authority chooses its own serial number. Here we choose smaller numbers for the sake of convenience of computation and explanation.

Showing Credential Creation : Alice inputs the position of the disclosed attribute, chooses her secret b and receives the challenge a from Bob. As we described in Section 3.2, Alice constructs her one-time showing credential public key h_{show}, signature sig_{show}, the tuple $(r_0, r_1, ..., r_l)$ by using a, b, her secret value α, her remaining hidden attributes $(x_2, ..., x_{l-1})$, and the the tuple $(w_0, ..., w_l)$. She needs to disclose the value of the showing attribute(s) along with the serial number during every showing process for security purposes. The serial number would be considered as the last attribute with fixed value in Alice's Digital Credential.

Note that the position of the revealed attribute is necessary for the construction of the tuple $(r_0, r_1, ..., r_l)$.

Verification : Bob, another user, receives Alice's showing attribute and its position, credential public key, signature and other assistant parameters. Then he will need to obtain the public key e and the serial number x_l from the appropriate authority to verify whether the attribute is valid or not.

4.3.1 Results and Analysis

We construct eight different original credentials, varying the number of attributes contained, for Alice by using the system parameters from Table 4.4 and the randomly chosen

Table 4.8: Alice's Attributes in her Original Credential

α	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9 serial number
55	7	77	39	61	3	38	9	15	123

attributes from Table 4.8.

When there is only one attribute (x_1) contained, Alice's original credential public key will be $h = h_0^{\alpha} g_1^{x_1} g_9^{x_9}$. And when there are two attributes (x_1, x_2) contained, Alice's original credential public key will be $h = h_0^{\alpha} g_1^{x_1} g_2^{x_2} g_9^{x_9}$. And so on and so forth. We use these eight different credentials to simulate Alice's different credentials with different numbers of attributes contained.

In the showing process of every one-time showing credential, Alice discloses one of her attributes (let us say, x_1) and the serial number (x_9) at the same time (x_9 is actually not one of Alice's attributes, and she discloses x_1 along with x_9 for security reasons mentioned in Section 3.5.5). She can continue to disclose another attribute if she wants. In our results (Table 4.9), we just consider the situation of disclosing one single attribute at each showing process in order to compare time consumed for the credentials with different numbers of attributes. Note that the computing complexity for disclosing different numbers of attributes (*e.g.,* disclosing only x_1 or x_1 and x_2 at a single showing process) is the same, since different attributes disclosure only affect the construction of r_i (see Section 3.4.1.5), so the time consumed should be similar.

Table 4.9: Time Consumed for the User Side (secs)

Number of Attributes	Original Credential Creation	D-H Exchange for w	Issuance	Showing	Verification	Whole Process
1	0.000077	0.378504	0.071522	0.257839	0.158962	0.866904
2	0.000099	0.490506	0.079770	0.275672	0.208298	1.054345
3	0.000100	0.608315	0.071142	0.281935	0.263979	1.225471
4	0.000128	0.722182	0.071866	0.277791	0.337685	1.409652
5	0.000252	0.857160	0.070922	0.275358	0.380963	1.584656
6	0.000121	0.947632	0.072453	0.262987	0.486090	1.769282
7	0.000163	1.092301	0.073249	0.260507	0.501812	1.928032
8	0.000164	1.181961	0.069095	0.275238	0.555113	2.081572

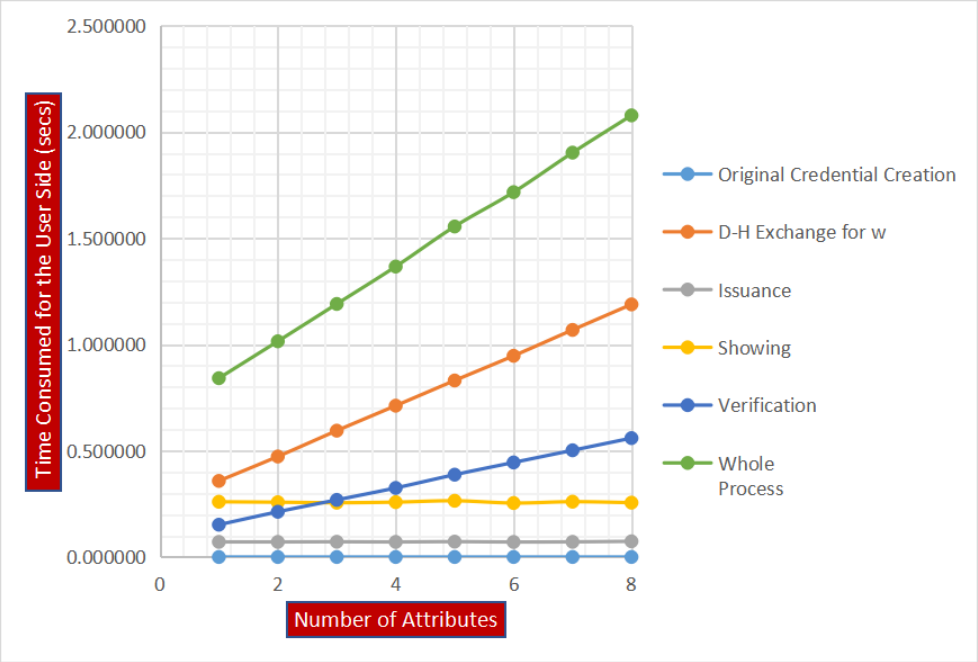

Figure 4.2: Average Time Consumed for Each Credential for 50 Runs

The results of time consumed for each process on the user side are shown in Table

4.9 (the program for each original credential has been run only once). We then run the program for each original credential for 50 times, and the average time consumed is graphed in Figure 4.2.

We can see that the whole process takes around 2 seconds when the credential contains 8 attributes. However, the process of D-H Exchange for w consumes more than half of the total time, which is actually not necessary in our scheme if Alice and the hospital trust each other. It is also obvious that when more attributes are contained in the credential, more time will be required for the executions which are related to the number of attributes contained.

To be more specific, based on the detailed description of protocols from Section 3.4.1, the time required for D-H Exchange for w and Verification increases significantly because of the increase of the amount of multiplication and exponentiation, when more attributes are contained in the credential, while that for Issuance (*i.e.*, getting signatures for h and w from the hospital) clearly stays the same because there are no attributes involved in this process. There is no obvious increases in the time required for Original Credential Creation and Showing processes because the exponents (*e.g.*, $a, b, \alpha, x_1, ..., x_9, r_0, r_2, ..., r_8$) used for calculations are too (relatively) small to influence the execution time.

Furthermore, the time consumed for each process on the user side can be graphed as six box plots shown in Figure 4.3, which specify the minimum, maximum, and median of our timing results. Around 90% of the resulting data are within the boxes, and most of the results are close to the minimum. The outliers beyond the upper bound indicate that some unexpected background applications running on the laptop might influence the execution time for our program.

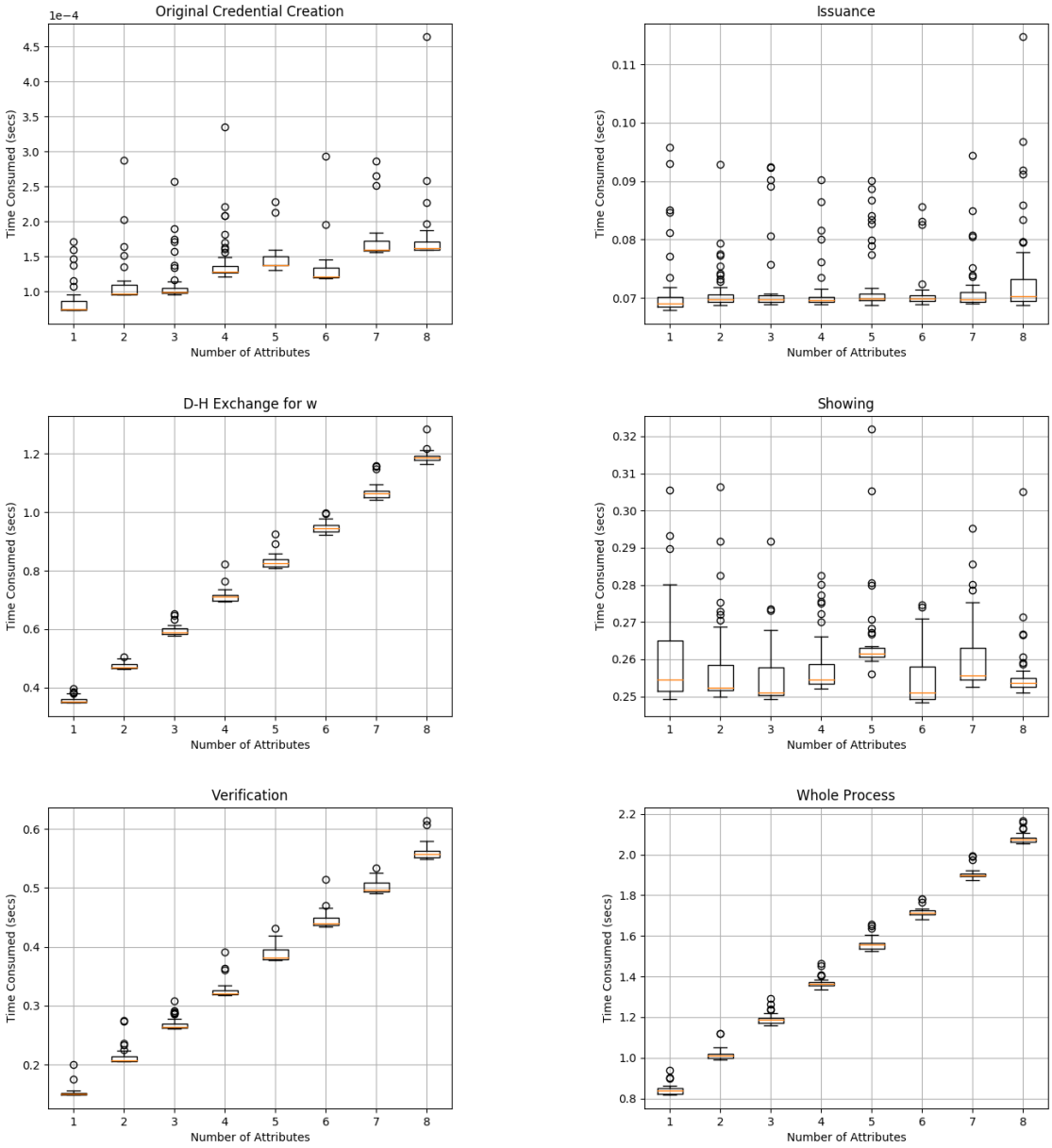

Figure 4.3: Time Consumed for Each Process

4.4 Python

Python (https://www.python.org) is an interpreted, high-level, general-purpose programming language, created by Guido van Rossum. In our implementation, there are 114 code lines on the authority side and 132 on the user side.

Here are the reasons why we chose Python (version 3.5.2.) for our proof of concept:

- It contains various built-in functions (libraries) and supports a large number of third-party packages.
- It is user-friendly and easy to learn. Complex implementations are achievable with simple and logical syntax using fewer lines.
- It is developed under an open source license, which makes it freely usable.
- It supports calculations with numbers of unlimited size, which is beneficial to our implementation. Its execution speed, when dealing with large numbers, is adequate for our proof of concept implementation, so there is no need to use any other third-party big number libraries.

4.4.1 Primality Test

To perform the primality test, we use a Python extension module, called *gmpy2* (https://gmpy2.readthedocs.io/en/latest/), which is the only third-party package used in our implementation. *gmpy2* supports multiple-precision arithmetic (including Miller-Rabin algorithm) and it is a C-coded extension module, which can speed up the primary testing.

By applying the function `gmpy2.is_prime(x[, n=25])`, we can check if x is probably prime for small divisors and up to `n` Miller-Rabin tests. In our implementation, we set `n` (which is referred to as t repetitions for primality testing in Section 2.1.1.1) as its default 25, so the probability that the pseudo-prime we found is actually nonprime is much less than the tolerable error probability $(1/2)^{80}$.

4.5 Chapter Summary

This chapter has further explained more details to do with the implementation of our proposal. The related parameters, results, and analysis for the timing results are also provided.

Chapter 5

Conclusions and Future Work

5.1 Conclusion

This thesis analyzes the limitations of Brands' Digital Credentials, and then makes the use of malleable signatures to randomize the issued credentials from the CA in Brands' Digital Credentials, which allows the blinding process to be moved from the issuing protocol (*i.e.*, Alice's interaction with the CA during the creation of the credential) to the showing protocols (*i.e.*, Alice's interaction with a verifier during the use of her credential).

This proposal, and its instantiation using the malleability of the RSA digital signature scheme, provides an efficient way to allow Brands' single-show credentials to be converted to multi-show without sacrificing any of their original security or privacy properties. With randomization of the credential and its associated signature in the showing protocol, Alice is able to prevent the linking attacks, including linking any of her transactions to her other transactions or to herself by the CA as well as by all verifiers (or their collusion).

We also propose an idea of replacing the CA by actual authorities (*e.g.*, the hospital). With this replacement, Alice can directly interact with the corresponding authority to obtain her issued credential. By applying our issuing and showing protocol, the replacement will not cause any security or privacy problems.

To prove the concept of our proposal, we provide the specific steps for conducting each

protocol, explain the mathematics, implement a Python script to demonstrate the whole technique, and provide some program timing results.

5.2 Future Work

The technique presented in this thesis has achieved Multi-show Unlinkability, but we can seek efficient ways to combine our protocol with additional biometric techniques to make our multi-show Digital Credentials also non-transferable. This will ensure that, for instance, Alice cannot lend her credential to several friends so that they all enjoy access to a subscription service for the price of a single user.

This work can be extended with other (additional) effective ways to avoid replay attacks. In particular, we are considering the best places and ways to add nonces or timestamps, for example, so that credentials cannot be maliciously replayed by any party. In the meantime, credentials cannot be illegally used by any unauthenticated parties.

The approach presented here allows Brands' Digital Credentials to be used in multi-show fashion. It probably can also be applied to improve Brands' other work (*e.g.*, Payment and Withdrawal Protocol, Smartcard Integration). Furthermore, this technique may be used to address privacy problems in Vehicular Ad Hoc Networks (VANETs).

This work needs to be developed with a proper (efficient and unlinkable) revocation extension. Several revocation extensions on Microsoft's U-Prove have been proposed [1, 48, 49, 68], some of which have been found to be tractable or with other disadvantages [40, 41]. Furthermore, the revocation techniques that complement U-Prove (or Brands' Digital Credential system) should not be directly applied on the scheme in this thesis because of the self-blindable property of our scheme. We have provided a possible direction in Section 3.6. Also, the revocation solution proposed by Brands et al. [13] in 2007 could be a good start.

The primality test used in this thesis applies a probabilistic algorithm. With the development of prime number theory, it may be possible to make a more efficient true primality

test available. Such a test would increase the security of the work in this thesis (as well as many other algorithms and protocols) because provable primes could be used throughout.

The implementation for this thesis was coded in Python. Other programming languages and big number libraries could certainly be helpful to the performance. Furthermore, it would be worth trying to further develop this proof of concept to create a real application (*e.g.,* including real data transfer between the authorities and the users) and even plug it into real credential environments.

U-Prove (Brands' work) can be implemented over elliptic curves, which makes it more efficient. However, our extension makes use of the malleability of the RSA signature scheme, and, thus, our scheme cannot be implemented over elliptic curves. A future direction could be replacing the RSA signature by a (discrete-logarithm-based) malleable signature which can be used over elliptic curves. Note, however, that our current scheme should be efficient enough for users who are using their PCs instead of mobile devices. Another direction could be replacing the basic RSA signature scheme by a post-quantum malleable signature scheme (if there is any) to make our system to be quantum-resistant.

We also could replace the CA by applying blockchain techniques. One obvious benefit of this is to ensure that all attributes issued cannot be modified by any parties. However, since all the transaction packages between users, issuers, and verifiers would be stored on the blockchain, the biggest challenge would also be to keep the users unlinkable.

Glossary

Blum integer: A composite integer n, which is the product of distinct prime integers p and q, where $p \equiv q \equiv 3 \pmod 4$.

Credential Authority (CA): The CA plays the role of issuer in Brands' Digital Credential system.

Desired subgroup $\mathbb{G}_s$: A Subgroup of $\mathbb{Z}_n^*$ with the desired property (i.e., its order is large enough so that computing discrete logarithm is intractable).

Desired Subgroup Generation (DSG) Algorithm: A algorithm for finding a desired generator which can generate $\mathbb{G}_s$ with a large enough order s (see Section 3.4.1.2).

Diffie-Hellman key agreement (D-H Exchange): A practical solution that allows two parties to establish a shared secret key over an open (unsafe) channel without meeting in advance or sharing any keying material (see Section 2.1.4).

Digital Credential: A data structure that allows its holder to determine for herself when, how, and to what extent she is willing to reveal her attributes to others, and to what extent others can link or trace this information.

Discrete Logarithm Problem (DLP): Given a prime p, a generator α of $\mathbb{Z}_p^*$, and an element $\beta \in \mathbb{Z}_p^*$, find the integer x, $0 \leq x \leq p-2$, such that $\alpha^x \equiv \beta \pmod p$.

$\gcd(a, b)$: The *greatest common divisor* of integers a and b is the largest positive integer that divides both a and b.

gen: A generator that generates the subgroup $\mathbb{G}_s$.

Issuer: The issuer is generally an organization, which issues a credential to an individual user. It may verify the validity of the user's attributes during the issuing process. In some systems, the issuer is able to revoke a previously issued credential.

Issuer-linking attack: A kind of linking attack which allows the issuer to collude with the verifiers so that the verifiers learn Alice's unrevealed attributes.

Issuing protocol: The issuer collaborates with the user to encode the user's personal attributes into a credential and cryptographically signs the credential.

Malleable Signature: Generally, a signature scheme is *malleable* if, for a given message and its signature, it is possible to efficiently modify the signature to be a valid signature on a related message without using the private signing key.

Multi-show Unlinkability of a credential: A credential can be shown to different verifiers without being linked to each other even if the issuers, the verifiers, and any other parties collude.

One-time Showing Credential: A single-use Digital Credential which is constructed by the user during the showing protocol.

Original Credential: The user's original Digital Credential issued by the authorities.

Showing protocol: An individual user, possessing a particular credential, shows it to a verifying organization to claim a privilege. Then the verifying organization checks the validity of the credential and the claimed privilege.

User: The user is assumed to be capable of communicating with the issuer to encode his/her attributes into the credentials and constructing the corresponding multi-show credential.

User-unlinkability of a credential: A credential can be shown to different verifiers without being traced back to the user (the credential holder) even if the issuers, the verifiers, and any other parties collude.

Valid: A credential or attribute being valid means it has not been illegally altered.

Verification: At the end of the showing protocol, the verifying organization checks the validity of the credential and the claimed privilege.

Verifier: The verifier is the organization with whom the user interacts in the showing process. It makes sure that the shown credentials are properly signed, currently valid and rightfully encoded with the claimed attributes.

Verifier-linking attack: A kind of linking attack, which allows the verifiers to collude with each other to learn Alice's unrevealed attributes.

$\mathbb{Z}_n^*$: A *multiplicative group* used in this thesis, such that $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n | \gcd(a, n) = 1\}$.

References

[1] Tolga Acar, Sherman SM Chow, and Lan Nguyen. Accumulators and u-prove revocation. In *International Conference on Financial Cryptography and Data Security*, pages 189–196. Springer, 2013.

[2] Carlisle Adams. Achieving non-transferability in credential systems using hidden biometrics. *Security and Communication Networks*, 4(2):195–206, 2011.

[3] Giuseppe Ateniese, Daniel H Chou, Breno De Medeiros, and Gene Tsudik. Sanitizable signatures. In *European Symposium on Research in Computer Security*, pages 159–177. Springer, 2005.

[4] Eric Bach. *Discrete logarithms and factoring*. Computer Science Division, University of California Berkeley, 1984.

[5] Michael Backes, Sebastian Meiser, and Dominique Schröder. Delegatable functional signatures. In *Public-Key Cryptography–PKC 2016*, pages 357–386. Springer, 2016.

[6] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1087–1098. ACM, 2013.

[7] Amira Barki, Solenn Brunet, Nicolas Desmoulins, and Jacques Traoré. Improved algebraic macs and practical keyed-verification anonymous credentials. In *International Conference on Selected Areas in Cryptography*, pages 360–380. Springer, 2016.

[8] Raghav Bhaskar, K Chandrasekaran, Satyanarayana V Lokam, PL Montgomery, Ramarathnam Venkatesan, and Yacov Yacobi. Vulnerabilities in anonymous credential systems. *Electronic Notes in Theoretical Computer Science*, 197(2):141–148, 2008.

[9] David Bissessar. Cryptographic credentials with privacy-preserving biometric bindings. Master's thesis, Université d'Ottawa/University of Ottawa, 2013.

[10] Johannes Blömer and Jan Bobolz. Delegatable attribute-based anonymous credentials from dynamically malleable signatures. In *International Conference on Applied Cryptography and Network Security*, pages 221–239. Springer, 2018.

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

[12] Stefan Brands. A technical overview of digital credentials. *Available from* http://www.cypherspace.org/credlib/brands-technical.pdf*, Feb*, 20:145–8, 2002.

[13] Stefan Brands, Liesje Demuynck, and Bart De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In *Australasian Conference on Information Security and Privacy*, pages 400–415. Springer, 2007.

[14] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[15] Christina Brzuska, Henrich C Pöhls, and Kai Samelin. Efficient and perfectly unlinkable sanitizable signatures without group signatures. In *European Public Key Infrastructure Workshop*, pages 12–30. Springer, 2013.

[16] Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. Composable and modular anonymous credentials: definitions and practical constructions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 262–288. Springer, 2015.

[17] 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, pages 93–118. Springer, 2001.

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

[19] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Annual International Cryptology Conference*, pages 56–72. Springer, 2004.

[20] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable signatures: Complex unary transformations and delegatable anonymous credentials. *IACR Cryptology ePrint Archive*, 2013:179, 2013.

[21] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable signatures: New definitions and delegatable anonymous credentials. In *Computer Security Foundations Symposium (CSF), 2014 IEEE 27th*, pages 199–213. IEEE, 2014.

[22] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.

[23] David Chaum. Blind signature system. In *Advances in cryptology*, pages 153–153. Springer, 1984.

[24] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.

[25] David Chaum. One-show blind signature systems, April 3 1990. US Patent 4,914,698.

[26] David Chaum. Zero-knowledge undeniable signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 458–464. Springer, 1990.

[27] David Chaum and Jan-Hendrik Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 118–167. Springer, 1986.

[28] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *Conference on the Theory and Application of Cryptography*, pages 319–327. Springer, 1988.

[29] David Chaum and Hans Van Antwerpen. Undeniable signatures. In *Conference on the Theory and Application of Cryptology*, pages 212–216. Springer, 1989.

[30] David Chaum and Eugène Van Heyst. Group signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 257–265. Springer, 1991.

[31] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[32] Elizabeth C Crites and Anna Lysyanskaya. Delegatable anonymous credentials from mercurial signatures. In *Cryptographers' Track at the RSA Conference*, pages 535–555. Springer, 2019.

[33] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[34] Jinnan Fan and Carlisle Adams. Using malleable signatures to allow multi-show capability in digital credentials. *International Journal of Sensor Networks and Data Communications*, 07, 2018.

[35] Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. *IET Information Security*, 12(3):166–183, 2018.

[36] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, pages 1–49, 2018.

[37] Christina Garman, Matthew Green, and Ian Miers. Decentralized anonymous credentials. In *NDSS*. Citeseer, 2014.

[38] Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 491–511. Springer, 2014.

[39] Lucjan Hanzlik and Kamil Kluczniak. A short paper on how to improve u-prove using self-blindable certificates. In *International Conference on Financial Cryptography and Data Security*, pages 273–282. Springer, 2014.

[40] Lucjan Hanzlik, Kamil Kluczniak, and Miroslaw Kutylowski. Attack on a u-prove revocation scheme from fc'13-exploiting the weakness of the underlying accumulator scheme (short paper). *Financial Cryptography, LNCS*, 2014.

[41] Lucjan Hanzlik, Przemyslaw Kubiak, and Miroslaw Kutylowski. Tracing attacks on u-prove with revocation mechanism. *IACR Cryptology ePrint Archive*, 2015:108, 2015.

[42] J Håstad, AW Schrift, and A Shamir. The discrete logarithm modulo a composite hides o (n) bits. In *Journal of Computer and System Sciences*. Citeseer, 1993.

[43] Jaap-Henk Hoepman, Wouter Lueks, and Sietse Ringers. On linkability and malleability in self-blindable credentials. In *IFIP International Conference on Information Security Theory and Practice*, pages 203–218. Springer, 2015.

[44] Robert Johnson, David Molnar, Dawn Song, and David Wagner. Homomorphic signature schemes. In *Cryptographers' Track at the RSA Conference*, pages 244–262. Springer, 2002.

[45] Jinhua Ma, Jianghua Liu, Xinyi Huang, Yang Xiang, and Wei Wu. Authenticated data redaction with fine-grained control. *IEEE Transactions on Emerging Topics in Computing*, 2017.

[46] Alfred J Menezes, Paul C. Oorschot, and Scott A. Vanstone. Handbook of applied cryptography, 1997.

[47] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[48] Lan Nguyen and Christian Paquin. U-prove designated-verifier accumulator revocation extension. https://www.microsoft.com/en-us/research/publication/u-prove-designated-verifier-accumulator-revocation-extension/, September 2013.

[49] Christian Paquin and Lan Nguyen. U-prove designated-verifier accumulator revocation extension. https://www.microsoft.com/en-us/research/publication/u-prove-designated-verifier-accumulator-revocation-extension-2/, May 2015.

[50] Christian Paquin and Greg Zaverucha. U-prove cryptographic specification v1.1 (revision 3). https://www.microsoft.com/en-us/research/publication/u-prove-cryptographic-specification-v1-1-revision-3/, December 2013. Released under the Open Specification Promise.

[51] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual International Cryptology Conference*, pages 129–140. Springer, 1991.

[52] Torben Pryds Pedersen. Distributed provers and verifiable secret sharing based on the discrete logarithm problem. *DAIMI Report Series*, 21(388), 1992.

[53] Giuseppe Persiano and Ivan Visconti. An efficient and usable multi-show non-transferable anonymous credential system. In *International Conference on Financial Cryptography*, pages 196–211. Springer, 2004.

[54] Henrich C Pöhls, Stefan Peters, Kai Samelin, Joachim Posegga, and Hermann de Meer. Malleable signatures for resource constrained platforms. In *IFIP International Workshop on Information Security Theory and Practices*, pages 18–33. Springer, 2013.

[55] Harsha Sandaruwan Gardiyawasam Pussewalage and Vladimir A Oleshchuk. An efficient multi-show unlinkable attribute based credential scheme for a collaborative e-health environment. In *2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC)*, pages 421–428. IEEE, 2017.

[56] Sietse Ringers, Eric Verheul, and Jaap-Henk Hoepman. An efficient self-blindable attribute-based credential scheme. In *International Conference on Financial Cryptography and Data Security*, pages 3–20. Springer, 2017.

[57] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[58] Ronald L Rivest and Robert D Silverman. Are 'strong' primes needed for rsa? In *The 1997 RSA Laboratories Seminar Series, Seminars Proceedings*, 1997.

[59] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.

[60] Zahava Shmuely. Composite diffie-hellman public-key generating systems are hard to break. In *Technical Report no. 356*. Computer Science Department, Technion-Israel Institute of Technology, 1985.

[61] Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge university press, 2009.

[62] William Stallings. *Cryptography and Network Security, 4/E*. Pearson Education India, 2006.

[63] Ron Steinfeld, Laurence Bull, and Yuliang Zheng. Content extraction signatures. In *International Conference on Information Security and Cryptology*, pages 285–304. Springer, 2001.

[64] IBM Research Zürich Security Team. Specification of the identity mixer cryptographic library, version 2.3.0. https://tinyurl.com/idemix-spec, February 2010.

[65] Giulia Traverso, Denise Demirel, and Johannes Buchmann. *Homomorphic Signature Schemes: A Survey*, volume 1. Springer, 2016.

[66] Eugène Van Heyst and Torben Pryds Pedersen. How to make efficient fail-stop signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 366–377. Springer, 1992.

[67] Eric R Verheul. Self-blindable credential certificates from the weil pairing. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 533–551. Springer, 2001.

[68] Eric R Verheul. Practical backward unlinkable revocation in fido, german e-id, idemix and u-prove. *IACR Cryptology ePrint Archive*, 2016:217, 2016.

[69] Eric R Verheul, Sietse Ringers, and Jaap-Henk Hoepman. The self-blindable u-prove scheme by hanzlik and kluczniak is forgeable. *IACR Cryptology ePrint Archive*, 2015:725, 2015.

[70] Lei Wei, Scott E Coull, and Michael K Reiter. Bounded vector signatures and their applications. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 277–285. ACM, 2011.

[71] Andrew C Yao. Theory and application of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, pages 80–91. IEEE, 1982.

[72] Zuoxia Yu, Man Ho Au, and Rupeng Yang. Accountable anonymous credentials. In *Advances in Cyber Security: Principles, Techniques, and Applications*, pages 49–68. Springer, 2019.