5-2024

A Post-Quantum Mercurial Signature Scheme

Madison Mabe
mnmabe@g.clemson.edu

A Post-Quantum Mercurial Signature Scheme

A Master's Project
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master's in Mathematics
Mathematics

---

by
Madison Nicole Mabe
May 2024

Abstract

This paper introduces the first post-quantum mercurial signature scheme. We also discuss how this can be used to construct a credential scheme, as well as some practical applications for the constructions.

Dedication

I would like to dedicate this work to my family and friends who have supported me while I have been at Clemson. Specifically, I would like to dedicate this work to my dad, my sister and most importantly my mom. My mom always dreamed of going back to school to get her Masters but had me and my sister and never did. She has been my biggest supporter, a listening ear, and a shoulder to cry on when things have been rough. So mom, this one's for you!

Acknowledgments

I would like to start by thanking my advisor, Dr. Ryann Cartor. She took me on as a student during the beginning of my second year and has taught me so much about not only math, but about life and how to be a professional in the cryptography field. She has been gracious and understanding during times of stress and has also pushed me to be the best I can be. I have had the time of my life doing research with her and cannot wait to continue on in the future.

I would also like to thank my committee members, Dr. Felice Manganiello, Dr. Shuhong Gao, and Dr. Rafael D'Oliveira for agreeing to be on my committee and for taking time out of their busy schedule to review my paper and presentation.

Finally, I would like to thank all of my friends who have supported me and encouraged me during this crazy time of my life. I would like to especially thank those who are also graduate students who have been there through the ups and downs of the reality of grad student life. Their support has meant everything to me.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Why Study Cryptography?

There have always been groups of people that wanted to securely send information back and forth without the possibility for anyone else to see their messages. This has been especially important in wars or even in letters that are sent between empires or countries. If a war general is sending an attack strategy to his army on the battle field, he wants to encode his message somehow just in case the opposing army gets a hold of the message. Some of these tactics first used like shifting letters around (Shift Cipher) and shifting blocks around (Block Cipher) are what some first think of when they think of Cryptography. Since then though, there has been a lot of development in technology and thus, more security is needed [19].

The shift ciphers mentioned above are considered to be classical cryptographic schemes. As mentioned, these shift ciphers are not secure and are easy to decipher computationally.

An example of a scheme that is still classically secure is RSA which was introduced in 1977 by by Ron Rivest, Adi Shamir and Leonard Adleman [16]. This scheme remains secure by relying on the hardness of factoring large integers.

An algorithm was introduced by Peter Shor in 1994 that factors large numbers in polynomial time [18]. This algorithm is based in quantum computing, but does require a large amount of qubits. Peter Shor also introduced an algorithm to help solve the factoring in the Discrete Log Problem. The pairing of widespread quantum computing and Shor's algorithm would render classical public key cryptography (based on discrete log and factoring) insecure.

A lot of cryptographic schemes are based on the hardness of these above problems. Some of these schemes are classical, meaning they are secure with classical computing power. With the rise in possibility for quantum computing, there has been a shift to study post-quantum cryptography (PQC), instead. The hope is to create schemes whose security will be able to withstand that computing power.

With this new interest in PQC, the National Institute of Standards and Technology (NIST) put out a call for proposals[1] of PQ-schemes that could be tested for security, and if found secure, could be standardized for use. This call for proposals happened in 2016 with multiple rounds ever since.

1.2 Why Look at Mercurial Signatures?

The concept of a Mercurial Signature was introduced by Elizabeth Crites and Anna Lysyanskaya in 2019 [8]. In this paper the authors also introduce a Mercurial Signature scheme whose security relies on the hardness of the discrete log problem. This means the proposed scheme will be vulnerable to Shor's Algorithm on quantum computers. This proposed scheme creates a message-signature pair, and with converters is able to convert both of the preexisting public and private keys as well as change the representation of the current message-signature pair. There has yet to be a quantum secure Mecurial signature scheme. So, there is a question of if there is some quantum secure signature scheme that fits this defintion of a Mercurical Signature. If such a scheme exists, there is also the question of whether it could still be used for a Delegatable Anonymous Credential, leading to more applications. This paper will introduce the first quantum secure Mercurial Signature scheme and its corresponding Delegatable Anonynous Credential.

1.3 Brief Overview

A signature scheme takes as input some message and a secret key into an algorithm, and outputs some message-signature pair, this signature now being linked to this message.These conventional signature schemes already allow delegatable credentials. A delegatable credential is a credential that is able to be passed down from a root of authority to others in a chain also needing

[1]`https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/call-for-proposals`

said credential or other credentials added along the way. Each user in the chain can use their chain of credentials to verify that the possess any credential in their chain. For example, consider Alice, who has a public signing key pk_A and a certification chain of length ℓ, can sign Bob's public key pk_B, which gives Bob a certification chain of length $\ell + 1$. So, Alice can verify that she has any credentials in her chain of length ℓ, and Bob can verify any of those credentials as well as the new credential that Alice gave to him [8].

The Mercurial Signature scheme also looks at anonymous credentials which allow a user to prove possession of a set of credentials, issued by some trusted issuer or issuers, that allow access to a resource. What makes these credentials anonymous is the fact that an issuer need not know the user's identity in order to issue a credential. Then, if it also delegatable, the credentials allow for much more privacy where even the users themselves do not know the actual identities of the links on their certification chains. They only know what they need to.

A concrete example of this is a discount program for senior citizens. Where the government possess some credential to allow senior citizens a discount on groceries that they give to some government official Alice to distribute to grocers. Say that Bob is one of these grocers, so when Carol, a senior citizen, comes in to get her groceries, she can provide some sort of information that will give her the discount. Note that this information can be scanning some sort of card or verifying a birthday, so that most of Carol's information if not all is kept a secret from everyone and she gets the discount without needing to know where it came from. This chain of giving a credential can be seen in the following diagram from [8]:

government → **Alice** (government official) → **Bob** (grocer) → **Carol** (senior citizen)

Chapter 2

Background

2.1 Anonymous Credentials

One application of Mercurial Signatures is to create a specific type of anonymous credeantials called Delegatable Anonymous Credentials. An anonymous credential system consists of two parties: the users and organizations. A pseudonym is linked to a user and that pseudonym is how the organization identifies the user. So the only thing the organization knows is this pseudonym, not the actual identity of the user. The organization can then issue a credential to this pseudonym and then the user can verify or prove possession of said credential to another organization that knows the user by a different pseudonym, without revealing any personal information and losing anonymity. There are anonymous credentials that can be used more than once, called multi-show credentials, or ones that are only used one time, called one-show credentials. It is important to note here that with a multi-show credential, each demonstration of the credential cannot be linked to a previous one.

Some basic properties that these credentials possess are as follows: each pseudonym and credential must be linked to a well-defined user, an organization can never find out anything about a user, and the interaction between organizations and users must be efficient. For the user to be well-defined, it cannot be possible that two users combine together some credentials and are able to receive a credential that one of them alone would not receive. Anonymity between the organization and user is also mentioned as being important as this is the anonymous part of the anonymous credential. This means, that an organization should only know some credentials that go with a pseudonym and if a user has two different pseudonyms, they should not be linked in any way. Finally, as in really anything with cryptography, we desire that the issuing and verification of a credential between user and organization is efficient, meaning if we are using multi-show credentials, the user should be able to verify possession without getting reissued a credential.

Definition 2.1.1 (Anonymous Credentials)**.** An anonymous credential scheme allows Issuers' to issue credentials to 'Provers', who can present said credentials as proof of authenticity to 'Verifiers' with the following functions:

- $\texttt{Issue}(nym, attr, auth, isk) \to (attr, \sigma)$
- $\texttt{Receive}(attr, \sigma, usk) \to (cred)$
- $\texttt{Prove}(cred, attr, usk) \to (\phi, ipk)$
- $\texttt{Verify}(\phi, ipk) \to \{0, 1\}$

An anonymous credential must also have correctness (Definition 3.1.2), unforgeability (Definition 3.1.3), issuer unlinkability (Definition 3.2.1), and multi-show unlinkability (Definition 3.2.2). For a more comprehensive and formal definition of anonymous credentials, refer to [10].

2.2 Relevant Multivariate Signature Schemes

One area of Post-Quantum cryptography is multivariate cryptogrpahy. These schemes are based on the MQ-Problem, which is the problem of solving multivariate polynomial equations over a finite field, and is proven to be NP-Complete [17]. Note that for a problem to be NP-Complete, this means that the problem is both NP and NP-hard. When a problem is NP, that means that the problem is working under nondeterministic polynomial time, meaning that the problem is solvable and verifiable in polynomial time. A problem M is NP-Hard if every L in M can be reduced to M in polynomial time. NP-complete problems are the hardest problems in NP. So, since the MQ-Problem is NP-Complete, this problem is a good candidate for the basis of post-quantum signature schemes. The general public key structure for these schemes is

$$y = T \circ F \circ S(x),$$

where T and S are linear maps and F is a quadratic map often referred to as the central map. Two multivariate schemes will be used throughout this paper are C^* and the Unbalanced Oil and Vinegar (UOV).

C^* was first introduced at Eurocrypt in 1988 [13], and is the first multivariate public key cryptosystem. C^* was broken in [15] using linearization equations. The construction of C^* is as follows: Let $\mathbb{F}_q$ be a finite field and $\mathbb{K}$ be a degree d extension. Now, use this set up, to get a multivariate representation of a univariate map. Let $f(x)$ be our central map such that $f : \mathbb{K} \to \mathbb{K}$ defined by $f(x) = x^{q^\theta}$, where $\gcd(q^\theta + 1, q^d - 1) = 1$ guaranteeing invertibility. Let U and T be affine invertible maps such that $U, T : \mathbb{F}_q^d \to \mathbb{F}_q^d$ and $\phi : \mathbb{F}_q^d \to \mathbb{K}$. Then, the public key is given by $P(x) = T \circ \phi^{-1} \circ f \circ \phi \circ U(x)$.

The Balanced Oil and Vinegar Scheme was first proposed in 1997 by Jacques Patarin [14]. It was then broken a year later in 1998 by Adi Shamir and Aviad Kipnis [12]. The Balanced Oil and Vinegar Scheme was then modifed by Patarin and a few other authors to be the Unbalanced Oil and Vinegar scheme (UOV). UOV was first proposed in 1999 [11], and is still considered to be secure against post-quantum attacks. The signature size is small and the signing and verification are relatively fast and efficient compared to other multivariate signature schemes that have been submitted to NIST. One down side is its larger public key size. The secret key for this scheme is comprised of two linear invertible maps U and T and a quadratic map often referred to as the central map. Define $U, T : \mathbb{F}_q^n \to \mathbb{F}_q^n$ and $F : \mathbb{F}_q^n \to \mathbb{F}_q^m$, where n is the total number of variables, o is the number of oil variables, v is the number of vinegar variables, and $m = o = n - v$. It is important to note here that the central map F can be written as a vector of polynomial equations as follows: $F = (f_1, \ldots, f_m)$ where

$$f_k = \sum_{i=1}^{v} \sum_{j=1}^{v} \alpha_{ij_k} x_i x_j + \sum_{i=1}^{v} \sum_{j=v+1}^{n} \beta_{ij_k} x_i x_j + \sum_{i=1}^{n} \gamma_{ij_k} x_i$$

The associated public key is a set of multivariate quadratic polynomials: $P(x) = T \circ F \circ U(x)$. In this public key, it is the invertible maps U and T that are used to hide the structure of the main part of our secret key, the central map. For the purposes of seeing an example and how this scheme works, there is an example listed in section 2.3, where U and T are both the identity to make computations simpler.

2.3 Example of UOV

For this example, let $n = 5$, $m = 2$, $q = 3$, $v = 3$, and $o = 2$. For ease in this example, let $T = U = I$, so that $P(x) = F(x)$ and $P(x) = y$.

F: $\mathbb{F}_3^5 \to \mathbb{F}_3^2$

Now, let $x = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ o_1 \\ o_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 1 \\ x_4 \\ x_5 \end{bmatrix}$

and $y = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$

$$\begin{aligned} f_1 =& x_1^2 + x_2^2 + x_3^2 + 2x_1x_2 + 2x_1x_3 + 2x_2x_3 + x_1x_4 + x_1x_5 + x_2x_4 + x_3x_4 \\ & + x_3x_5 + x_1 + x_2 + x_3 + x_4 + x_5 \end{aligned}$$

$$\begin{aligned} f_2 =& x_1^2 + 2x_2^2 + x_3^2 + x_2x_3 + x_1x_4 + x_1x_5 + x_2x_4 + 2x_2x_5 + 2x_3x_4 + x_3x_5 \\ & + 2x_1 + x_2 + x_3 + x_4 + 2x_5 \end{aligned}$$

$w = U^{-1}y = Iy = y$ and $F(z) = w = y$

So, we have $\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, and therefore, $z = \begin{bmatrix} x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$.

Now, since $x = T^{-1}z = Iz = z$, we have that $x = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 0 \\ 2 \end{bmatrix}$

Note that we usually do verification with P, but Since $P = F$, then there is no need to look at this because we would just be plugging in the same numbers as our F that we used to solve.

Chapter 3

Mercurial Signatures and Delegatable Anonymous Credentials

The following section describes the definitions relevant to mercurial signatures, anonymous credentials, and delegatable anonymous credentials. These definitions are abridged, but for the unabridged versions refer to [8] and [10]. Note here that vectors are notated by bold lowercase letters.

3.1 Mercurial Signatures

Definition 3.1.1 (Mercurial Signature [8])**.** A mercurial signature scheme for parameterized equivalence relations $\mathcal{R}_{\mathrm{M}}, \mathcal{R}_{\mathtt{pk}}, \mathcal{R}_{\mathtt{sk}}$ is a tuple of the following polynomial-time algorithms, which are deterministic algorithms unless otherwise stated:

$$\mathcal{R}_M = \{(M, M') \in (\mathbb{G}_1^*)^\ell \times (\mathbb{G}_1^*)^\ell \mid \exists r \in \mathbb{Z}_p^* \text{ such that } M' = M^r\}$$

$$\mathcal{R}_{\mathtt{sk}} = \{(\mathtt{sk}, \tilde{\mathtt{sk}}) \in (\mathbb{Z}_p^*)^\ell \times (\mathbb{Z}_p^*)^\ell \mid \exists r \in \mathbb{Z}_p^* \text{ such that } \tilde{\mathtt{sk}} = \mathtt{sk}^r\}$$

$$\mathcal{R}_{\mathtt{pk}} = \{(\mathtt{pk}, \tilde{\mathtt{pk}}) \in (\mathbb{G}_2^*)^\ell \times (\mathbb{G}_2^*)^\ell \mid \exists r \in \mathbb{Z}_p^* \text{ such that } \tilde{\mathtt{pk}} = \mathtt{pk}^r\}$$

$\mathtt{PPGen}(1^k)(1^k) \to PP$: Compute $\mathtt{BG} \leftarrow \mathtt{BGGen}(1^k)$. Output $PP = \mathtt{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, \hat{P}, e)$. Now that $\mathtt{BG}$ is well-defined, the relations $\mathcal{R}_M, \mathcal{R}_{\mathtt{pk}}, \mathcal{R}_{\mathtt{sk}}$ are also well-defined. $\mathtt{sample}_\rho$ and $\mathtt{sample}_\mu$ are the same algorithm, namely the one that samples a random element of $\mathbb{Z}_p^*$.

$\mathtt{KeyGen}(PP, \ell) \to (\mathtt{pk}, \mathtt{sk})$: For $1 \leq i \leq \ell$, pick $x_i \leftarrow \mathbb{Z}_p^*$ and set secret key $\mathtt{sk} = (x_1, \ldots, x_\ell)$. Compute public key $\mathtt{pk} = (\hat{X}_1, \ldots, \hat{X}_\ell)$, where $\hat{X}_i = \hat{P}^{x_i}$ for $1 \leq i \leq \ell$. Output $(\mathtt{pk}, \mathtt{sk})$.

$\mathtt{Sign}(\mathtt{sk}, M) \to \sigma$: On input $\mathtt{sk} = (x_1, \ldots, x_\ell)$ and $M = (M_1, \ldots, M_\ell) \in (\mathbb{G}_1^*)^\ell$, pick a random $y \to \mathbb{Z}_p^*$ and output $\sigma = (Z, Y, \hat{Y})$, where $Z \leftarrow (\prod_{i=1}^\ell M_i^{x_i})^y$, $Y \leftarrow P^{\frac{1}{y}}$ and $\hat{Y} \leftarrow \hat{P}^{\frac{1}{y}}$.

$\mathtt{Verify}(\mathtt{pk}, M, \sigma) \to 0/1$: On input $\mathtt{pk} = (\hat{X}_1, \ldots, \hat{X}_\ell)$, $M = (M_1, \ldots, M_\ell)$, and $\sigma = (Z, Y, \hat{Y})$, check whether $\prod_{i=1}^\ell e(M_i, \hat{X}_i) = e(Z, \hat{Y}) \wedge e(Y, \hat{P}) = e(P, \hat{Y})$. If it holds, output 1; otherwise output 0.

$\mathtt{ConvertSK}(\mathtt{sk}, \rho)(\mathtt{sk}, \rho) \to \tilde{\mathtt{sk}}$: On input $\mathtt{sk} = (x_1, \ldots, x_\ell)$ and a key converter $\rho \in \mathbb{Z}_p^*$, output the new secret key $\tilde{\mathtt{sk}} = \mathtt{sk}^\rho$.

$\mathtt{ConvertPK}(\mathtt{pk}, \rho)(\mathtt{pk}, \rho) \to \tilde{\mathtt{pk}}$: On input $\mathtt{pk} = (\hat{X}_1, \ldots, \hat{X}_\ell)$ and a key converter $\rho \in \mathbb{Z}_p^*$, output the new public key $\tilde{\mathtt{pk}} = \mathtt{pk}^\rho$.

$\mathtt{ConvertSig}(\mathtt{pk}, M, \sigma, \rho)(\mathtt{pk}, M, \sigma, \rho) \to \tilde{\sigma}$: On input $\mathtt{pk}$, message M, signature $\sigma = (Z, Y, \hat{Y})$ and key converter $\rho \in \mathbb{Z}_p^*$, sample $\psi \leftarrow \mathbb{Z}_p^*$. Output $\tilde{\sigma} = (Z^{\psi\rho}, Y^{\frac{1}{\psi}}, \hat{Y}^{\frac{1}{\psi}})$

$\mathtt{ChangeRep}(\mathtt{pk}, M, \sigma, \mu)(\mathtt{pk}, M, \sigma, \rho) \to (M', \sigma')$: On input $\mathtt{pk}$, M, $\sigma = (Z, Y, \hat{Y})$, $\mu \in \mathbb{Z}_p^*$, sample $\psi \leftarrow \mathbb{Z}_p^*$. Compute $M' = M^\mu$, $\sigma' = (Z^{\psi\mu}, Y^{\frac{1}{\psi}}, \hat{Y}^{\frac{1}{\psi}})$. Output (M', σ').

To write out in simpler terms to explain what the algorithm is doing:

Generate Public Parameters $\texttt{PPGen}(1^k) \to PP$: On input the security parameter 1^k, this probabilistic algorithm outputs the public parameters PP. This includes parameters for the parameterized equivalence relations $\mathcal{R}_M, \mathcal{R}_{\texttt{pk}}, \mathcal{R}_{\texttt{sk}}$ so that they are well-defined. It also includes parameters for the algorithms $\texttt{sample}_\rho$ and $\texttt{sample}_\mu$ which sample key and message converters, respectively.

Key Generation: $\texttt{KeyGen}(PP, \ell) \to (\texttt{pk}, \texttt{sk})$: On input the public parameters PP and a length parameter ℓ, this probabilistic algorithm outputs a key pair $(\texttt{pk}, \texttt{sk})$. The message space $\mathcal{M}$ is well-defined from PP and ℓ. This algorithm also defines a correspondence between public and secret keys: we write $(\texttt{pk}, \texttt{sk}) \in \texttt{KeyGen}(PP, \ell)$ if there exists a set of random choices that KeyGen would result in $(\texttt{pk}, \texttt{sk})$ as the output.

Signature Algorithm: $\texttt{Sign}(\texttt{sk}, M) \to \sigma$: On input the signing key sk and a message $M \in \mathcal{M}$, this probabilistic algorithm outputs a signature σ.

Signature Verification: $\texttt{Verify}(\texttt{pk}, M, \sigma) \to 0/1$: On input the public key pk, a message $M \in \mathcal{M}$, and a purported signature σ, output 0 or 1.

Secret Key Conversion: $\texttt{ConvertSK}(\texttt{sk}, \rho) \to \tilde{\texttt{sk}}$: On input sk and a key converter $\rho \in \text{sample}_\rho$ output a new secret key $\tilde{\texttt{sk}} \in [\texttt{sk}]_{\mathcal{R}_{\texttt{sk}}}$.

Public Key Conversion: $\texttt{ConvertPK}(\texttt{pk}, \rho) \to \tilde{\texttt{pk}}$: On input pk and a key converter $\rho \in \text{sample}_\rho$ output a new public key $\tilde{\texttt{pk}} \in [\texttt{pk}]_{\mathcal{R}_{\texttt{pk}}}$ (correctness of this operation, defined below, will guarantee that if pk corresponds to sk, then $\tilde{\texttt{pk}}$ corresponds to $\tilde{\texttt{sk}} = \texttt{ConvertSK}(\texttt{sk}, \rho)$).

Signature Conversion: $\texttt{ConvertSig}(\texttt{pk}, M, \sigma, \rho) \to \tilde{\sigma}$: On input pk, a message $M \in \mathcal{M}$, a signature σ, and key converter $\rho \in \text{sample}_\rho$, this probabilistic algorithm returns a new signature $\tilde{\sigma}$ (correctness of this will require that whenever $\texttt{Verify}(\texttt{pk}, M, \sigma) = 1$, it will also be the case that $\texttt{Verify}(\tilde{\text{pk}}, M, \tilde{\sigma}) = 1$.)

Change Message Representative: $\texttt{ChangeRep}(\texttt{pk}, M, \sigma, \mu) \to (M', \sigma')$: On input pk, a message $M \in \mathcal{M}$, a signature σ, and a message converter $\mu \in \text{sample}_\mu$, this probabilistic algorithm computes a new message $M' \in [M]_{\mathcal{R}_M}$ and a new signature σ' and outputs (M', σ') (correctness of this will require that $\texttt{Verify}(\texttt{pk}, M, \sigma) = 1 \Rightarrow \texttt{Verify}(\texttt{pk}, M', \sigma') = 1$).

A mercurial signature must be correct and unforgeable.

Definition 3.1.2 (Correctness [8])**.** A mercurial signature scheme ($\texttt{PPGen}$, $\texttt{KeyGen}$, $\texttt{Sign}$, $\texttt{Verify}$, $\texttt{ConvertSK}$, $\texttt{ConvertPK}$, $\texttt{ConvertSig}$, $\texttt{ChangeRep}$) for parameterized equivalence relations, $\mathcal{R}_M, \mathcal{R}_{\texttt{pk}}, \mathcal{R}_{\texttt{sk}}$ is correct if it satisfies the following conditions for all k, for all $PP \in \texttt{PPGen}(1^k)$, *for all* $\ell > 1$*, for all* $(\texttt{pk}, \texttt{sk}) \in \texttt{KeyGen}(PP, \ell)$:

Verification For all $M \in \mathcal{M}$, for all $\sigma \in \texttt{Sign}(\texttt{sk}, M)$, $\texttt{Verify}(\texttt{pk}, M, \sigma)$= 1.

Key Conversion For all $\rho \in \text{sample}_\rho$, $(\texttt{pk}, \texttt{sk}) \in \texttt{KeyGen}(PP, \ell)$. Moreover, $\tilde{\texttt{pk}} \in [\texttt{sk}]_{\mathcal{R}_{\texttt{sk}}}$ and $\tilde{\texttt{pk}} \in [\texttt{pk}]_{\mathcal{R}_{\texttt{pk}}}$.

Signature Conversion For all $M \in \mathcal{M}$, for all σ such that $\texttt{Verify}(\texttt{pk}, M, \sigma) = 1$, for all $\rho \in \text{sample}_\rho$, for all $\tilde{\sigma} \in \texttt{ConvertSig}(\texttt{pk}, M, \sigma, \rho)$, $\texttt{Verify}(\tilde{\texttt{pk}},\ M,\ \tilde{\sigma}) = 1$.

Change of Message Representative For all $M \in \mathcal{M}$, for all σ such that $\texttt{Verify}(\texttt{pk}, M, \sigma)$= 1, for all $\mu \in \text{sample}_\mu$, $\texttt{Verify}(\texttt{pk}, M', \sigma') = 1$, where $(M', \sigma') = \texttt{ChangeRep}(\texttt{pk}, M, \sigma, \mu)$. Moreover, $M' \in [M]_{\mathcal{R}_M}$.

Definition 3.1.3 (Unforgeability [8])**.** A mercurial signature scheme ($\texttt{PPGen}$, $\texttt{KeyGen}$, $\texttt{Sign}$, $\texttt{Verify}$, $\texttt{ConvertSK}$, $\texttt{ConvertPK}$, $\texttt{ConvertSig}$, $\texttt{ChangeRep}$) *for parameterized equivalence relations* $\mathcal{R}_M, \mathcal{R}_{\texttt{pk}}, \mathcal{R}_{\texttt{sk}}$ is unforgeable if for all polynomial-length parameters $\ell(k)$ and all probabilistic, polynomial-time (PPT) algorithms $\mathcal{A}$ having access to a signing oracle, there exists a negligible function ν such that:

$$\begin{aligned}\Pr[PP \leftarrow \texttt{PPGen}(1^k); (\texttt{pk}, \texttt{sk}) \leftarrow \texttt{KeyGen}(PP, \ell(k)); (Q, \texttt{pk}^*, M^*, \sigma^*) \leftarrow \\ \mathcal{A}^{\text{Sign}(\texttt{sk},\cdot)}(\texttt{pk}) : \forall M \in Q, [M^*]_{\mathcal{R}_M} \neq [M]_{\mathcal{R}_M} \wedge [\texttt{pk}^*]_{\mathcal{R}_{\texttt{pk}}} = [\texttt{pk}]_{\mathcal{R}_{\texttt{pk}}} \\ \wedge \text{Verify}(\texttt{pk}^*, M^*, \sigma^*) = 1] \leq \nu(k)\end{aligned}$$

where Q is the set of queries that $\mathcal{A}$ has issued to the signing oracle.

In informal terms, suppose Alice generates the parameters, public key, and secret key in the described manner. Eve then has access to a list, Q, of valid message-signature pairs generated by Alice. Assume Eve tries to create a valid σ^* under $\texttt{pk}$ or any equivalent $\texttt{pk}^*$, for a message M^*, not equivalent to any of the message signature pairs in Q. The probability that Eve can do so is negligible.

3.1.1 Properties For a DAC From a Mercurial Signature

While only correctness and unforgeability are necessary to create a mercurial signature scheme, there are two more attributes that are beneficial when using a mercurial signature to create a DAC: Class-hiding and Origin-hiding. Crites and Lysyanskaya introduce these hiding properties and they give a thorough explanation of how to use a mercurial signature with these properties to create a DAC in [8].

The class hiding property deals with message class hiding and public key class hiding. The message class hiding property states that if an adversary is given two distinct messages, M_1 and M_2, they should have a negligible advantage of determining whether $M_2 \in [M_1]_{\mathcal{R}}$ or $M_2 \notin [M_1]_{\mathcal{R}}$. Public key class hiding is the property such that if an adversary is given two distinct public keys, P_1 and P_2 and oracles access to the signing algorithm, they should have a negligible advantage of determining whether $P_2 \in [P_1]_{\mathcal{R}}$ or $P_2 \notin [P_1]_{\mathcal{R}}$.

A mercurial signature has origin-hiding of `ChangeRep` if two different message signature pairs are distributed in the same manner, even with an adversarial `pk`, where M_0 and M_1 are in the same equivalence class, the resulting message- signature pair hides whether it came from M_0 or M_1. A mercurial signature has origin-hiding of `ConvertSig` if given a signature on a message M, the output of `ConvertSig` hides whether `ConvertSig` was given one `pk` or another `pk` in the same equivalence class. Note that this applies even for an adversarially generated `pk`.

3.2 Delegatable Anonymous Credentials

One of the applications for a mercurial signature is to create an anonymous credential, specifically a delegatable anonymous credential. An anonymous credential is a credential that is issued and verified anonymously so that the users information is not revealed. This can then be transformed into a delegatable form by having an anonymous credential chain within the scheme. A delegatable anonymous credential scheme must uphold certain properties pertaining to the anonymity conditions of any user. While the exact definitions and conditions that are necessary for delegatable anonymous credentials will often vary, this paper will be referencing [10] for the essential components and their respective definitions.

Definition 3.2.1 (Issuer Unlinkability [10])**.** A witness to a valid `Prove/Verify` protocol transcript should have negligible advantage in determining which `Issue/Receive` protocol run generated the prover's credential, except for any advantage gained from attributes disclosed by the prover, even if the prover and verifier collude.

Definition 3.2.2 (Multi-show Unlinkability [10]). A witness to multiple `Prove/Verify` protocol runs linked to the same issuer should have negligible advantage in determining which, if any, protocol runs involved the same credential, except for any advantage gained from attributes disclosed by the prover(s), even if the issuer and verifier collude.

Definition 3.2.3 (Delegatable Anonymous Credential). A delegatable anonymous credential holds the same properties as an anonymous credential throughout a multi-user chain of credentials. The credential chain must allow for anonymous delegation of credentials among different parties. This means that no party should have knowledge of the identity of any specific link in the chain nor the number of times a credential has been used. This type of scheme consists of algorithms `(Setup, KeyGen, NymGen)` and protocols for issuing/receiving a credential and proving/verifying possession of a credential [8].

It must also be correct and secure. There are various manners to define secure, see [1], [8], [5], [7], [2], [6] for more detailed explanations. In this paper, we choose to leave out a definitive definition, and we operate under the assumption that UOV is considered secure.

3.3 Example

3.3.1 Mercurial Signature Example

Let G_1 and G_2 be multiplicative groups of prime order p, and $\mathbb{Z}_p^*$ be the multiplicative group without the identity. The length of the message M, will be denoted by ℓ.

Let $G_1 = \mathbb{Z}_7$ and $G_2 = \mathbb{Z}_{7^2} = \mathbb{Z}_{49}$. Thus, $G_1 = \{0, 1, 2, 3, 4, 5, 6\}$ and $G_2 = \{0, 1, 2 \ldots 47, 48\}$. Now note that the base prime of both G_1 and G_2 is $p = 7$ and thus, $\mathbb{Z}_p^* = \mathbb{Z}_7^* = \{0, 2, 3, 4, 5, 6\}$. Now, let the length of the message be $l = 3$ here.

Key Generation

Pick some $x_i \in \mathbb{Z}_7^*$ and set the secret key as $Sk = (2, 3, 4)$. Now, note that P is the multiplicative generator for G_1 and $\hat{P}$ is the multiplicative generator of for G_2. Therefore, for this example, $P = 2$ and $\hat{P} = 2$. Now we want to generate our public key. In order to do this, we note that our public key (pk) is $pk = (\hat{x_1}, \hat{x_2}, \hat{x_3})$, where each $\hat{x_i} = \hat{P}^{x_i}$. Therefore we have the following:

$$\hat{x_i} = \hat{P}^{x_i} = \begin{cases} \hat{x_1} = \hat{P}^{x_1} = 2^2 \equiv 4 \mod 7 \\ \hat{x_2} = \hat{P}^{x_2} = 2^3 = 8 \equiv 1 \mod 7 \\ \hat{x_3} = \hat{P}^{x_3} = 2^4 = 16 \equiv 2 \mod 7 \end{cases} \tag{3.3.0.1}$$

So our public key is $pk = (4, 1, 2)$

Sign: (sk, M) $\rightarrow \sigma$

Input the secret key $sk = (2, 3, 4)$ and the message $M = (3, 5, 6) \in (\mathbb{Z}_7^*)^3$.

Now, pick a random $y \leftarrow \mathbb{Z}_7^*$, so let y=2.

$Z = (\prod_{i=1}^{3} m_i^{x_i})^y = (3^2 \cdot 5^3 \cdot 6^4)^2 \equiv 4 \mod 7$

$Y = P^{\frac{1}{y}} = 2^{\frac{1}{2}} \equiv 3 \mod 7$ and $\hat{Y} = \hat{P}^{\frac{1}{y}} = 2^{\frac{1}{2}} \equiv 10 \mod 49$

So our signature is $\sigma = (Z, Y, \hat{Y}) = (4, 3, 10)$.

Verify$(\mathbf{pk}, \mathbf{M}, \sigma) \rightarrow \mathbf{0/1}$

Input $pk = (4, 1, 2)$, $M = (3, 5, 6)$, and $\sigma = (4, 3, 10)$.

1.) Check that $\prod_{i=1}^{3} m_i \cdot \hat{x}_i = e(Z, \hat{Y})$

So we want to check that $e(3, 4) \cdot e(5, 1) \cdot e(6, 2) = e(4, 10)$.

- $e(3, 4) = e(5 \cdot 2, 2 \cdot 2) = e(2, 2)^{5 \cdot 2} = 2^{10} \equiv 2 \mod 7$
- $e(5, 1) = e(6 \cdot 2, 25 \cdot 2) = e(2, 2)^{6 \cdot 25} = 2^{150} \equiv 1 \mod 7$
- $e(6, 2) = e(3 \cdot 2, 1 \cdot 2) = e(2, 2)^{3 \cdot 1} = 2^3 \equiv 1 \mod 7$
- $e(4, 10) = e(2 \cdot 2, 5 \cdot 2) = e(2, 2)^{2 \cdot 5} = 2^{10} \equiv 2 \mod 7$

Therefore, we have $e(3, 4) \cdot e(5, 1) \cdot e(6, 2) = 2 \cdot 1 \cdot 1 = 2 = e(4, 10)$

2.) Check that $e(Y, \hat{P}) = e(P, \hat{Y})$.

So we want to check that $e(3, 2) = e(2, 10)$

- $e(3,2) = e(5 \cdot 2, 1 \cdot 2) = e(2,2)^{5 \cdot 1} = 2^5 \equiv 4 \mod 7$
- $e(2,10) = e(1 \cdot 2, 5 \cdot 2) = e(2,2)^{1 \cdot 5} = 2^5 \equiv 4 \mod 7$

Therefore, we have $e(3,2) = 4 = e(2,10)$

So, overall, both conditions hold, and thus the message signature pair verifies.

Chapter 4

Post Quantum MS

4.1 UOV*-MS

This section introduces the first post-quantum mercurial signature scheme. This scheme uses UOV as a signature scheme, and uses C^* in the `ConvertSK`, `ConvertPK`, `ConvertSig`, and `ChangeRep`. This scheme has the properties of unforgeablilty, correctness, origin-hiding of `CovertSig`, and origin-hiding of `ChangeRep`. It also has public key class-hiding up to the security of C^*.

This mercurial signature scheme lacks message class-hiding, but the construction to a Semi-DAC implementation will have message class hiding up to the security of C^*. The possible modification to UOV*-MS to introduce both message class-hiding and public key class-hiding are left for future research.

4.1.0.1 UOV*-MS Algorithms

The message space for UOV*-MS is the set of all vectors $M = (M^{(1)}, \ldots, M^{(\ell)})$, where $M^{(i)} \in \mathbb{F}_q^n$ for every $1 \leq i \leq \ell$. Let Π represent a C^* public key, $\Pi(\mathbf{x}) = W \circ F \circ V(\mathbf{x})$.

$$\mathcal{R}_{\mathtt{sk}} = \{(\mathtt{sk}, \tilde{\mathtt{sk}}) \in (V_{sec}^{\ell}) \times (V_{sec}^{\ell}) \mid \exists \Pi \text{ such that } \tilde{\mathtt{sk}} = \mathtt{sk} \circ \Pi^{-1}\}$$

$$\mathcal{R}_{\mathtt{pk}} = \{(\mathtt{pk}, \tilde{\mathtt{pk}}) \in (V_{pub}^{\ell}) \times (V_{pub}^{\ell}) \mid \exists \Pi \text{ such that } \tilde{\mathtt{pk}} = \mathtt{pk} \circ \Pi^{-1}\}$$

$$\mathcal{R}_{M} = \{(M, M') \in (V_{mes}^{\ell}) \times (V_{mes}^{\ell}) \mid \exists a \in \mathbb{F}_q \text{ such that } M' = a^2 M\}$$

$\texttt{PPGen}(1^k)\rightarrow PP$: UOV and C^* parameter selection restricting the UOV central polynomials to those with the form $\sum_{i=1}^{v}\sum_{j=1}^{v}\alpha_{ij_k}x_ix_j+\sum_{i=1}^{v}\sum_{j=v+1}^{n}\beta_{ij_k}x_ix_j$.

$\texttt{KeyGen}(PP,\ell)\rightarrow(\texttt{pk},\texttt{sk})$: Choose random invertible matrices $\mathbf{T}^{(i)}\in\mathbb{F}_q^{m\times m}$ and $\mathbf{U}^{(i)}\in\mathbb{F}_q^{n\times n}$. Generate a public-secret key pair using the UOV key generation algorithm. Let $P^{(i)}=\mathbf{U}^{(i)}\circ F^{(i)}\circ\mathbf{T}^{(i)}$. $\texttt{sk}=(\mathbf{T}^{(i)},F^{(i)},\mathbf{U}^{(i)})$ for $1\le i\le\ell$ and $\texttt{pk}=(P^{(1)},\dots,P^{(\ell)})$.

$\texttt{Sign}(\texttt{sk},M)\rightarrow\sigma$: For a message $M=(M^{(1)},\dots,M^{(\ell)})$ and public key $P=(P^{(1)},\dots,P^{(\ell)})$, create the signature $\sigma=(\mathbf{x}^{(1)},\dots,\mathbf{x}^{(\ell)})$ such that $\mathbf{x}^{(i)}=\left(\mathbf{U}^{(i)}\right)^{-1}\circ F^{(i)}\circ(\mathbf{T}^{(i)})^{-1}M^{(i)}$.

$\texttt{Verify}(\texttt{pk},M,\sigma)\rightarrow 0/1$: If $\left(P^{(1)}(\mathbf{x}^{(1)}),\dots,P^{(\ell)}(\mathbf{x}^{(\ell)})\right)=(M^{(1)},\dots,M^{(\ell)})$, output 1. Otherwise, output 0.

$\texttt{ConvertSK}(\texttt{sk},\Pi)\rightarrow\tilde{\texttt{sk}}$: On input $\texttt{sk}=\{(\mathbf{T}^{(i)},F^{(i)},\mathbf{U}^{(i)}):1\le i\le\ell\}$, and a key converter $\Pi=W\circ F\circ V$, we get $\tilde{\texttt{sk}}=\{(\mathbf{T}^{(i)},F^{(i)},\mathbf{U}^{(i)}\circ\Pi):1\le i\le\ell\}$.

$\texttt{ConvertPK}(\texttt{pk},\Pi^{-1})\rightarrow\tilde{\texttt{pk}}$: On input $\texttt{pk}=(P^{(1)},\dots,P^{(\ell)})$ and a key converter $\Pi=W\circ F\circ V$, output the new public key $\tilde{\texttt{pk}}_i=P^i\circ\Pi^{-1}$.

$\texttt{ConvertSig}(\texttt{pk},M,\sigma,P_c)\rightarrow\tilde{\sigma}$: On input $\texttt{pk}$, message M, signature σ, and key converter $\Pi=W\circ F\circ V$, output $\tilde{\sigma}=\Pi(\mathbf{x}^{(i)})$.

$\texttt{ChangeRep}(\texttt{pk},M,\sigma)\rightarrow(M',\sigma')$: On input $\texttt{pk},M,\sigma=(x^{(1)},\dots,x^{(\ell)})$, sample $a\in\mathbb{F}_q$. Compute $M'=a^2M$, and $\sigma'=a\mathbf{x}$. Output (M',σ').

4.1.1 Security Proofs of UOV*-MS

4.1.1.1 Correctness

Lemma 1 (Correctness for Signature Conversion)**.** *If* $\texttt{Verify}(\texttt{pk},M,\sigma)\rightarrow 1$, *then* $\texttt{Verify}(\tilde{\texttt{pk}},M,\tilde{\sigma})\rightarrow 1$.

Proof. Let $\texttt{Verify}(\texttt{pk},M,\sigma)=1$, then $P(\mathbf{x})=M$. Examine $\texttt{Verify}(\tilde{\texttt{pk}},M,\tilde{\sigma})$ and by the public key conversion and signature conversion algorithms, $\tilde{\texttt{pk}}=P\circ\Pi^{-1}$ and $\tilde{\sigma}=\Pi(\mathbf{x})$. Thus,

$$\begin{aligned}\tilde{P}(\tilde{\sigma})&=P\circ\Pi^{-1}(\Pi(\mathbf{x}))\\&=P(\mathbf{x})\\&=M.\end{aligned}$$

Therefore, $\texttt{Verify}(\tilde{\texttt{pk}}, M, \tilde{\sigma}) = 1$. ■

Lemma 2 (Scaling the Generating Function)**.** For $a \in \mathbb{F}_q$ and $F = (f_1, \ldots, f_n)$ as in as in the UOV* parameter selection, $F(a\mathbf{x}) = a^2 F(x)$.

Proof. Let $F(\mathbf{x}) = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}$ where $f_k = \sum_{i=1}^{v}\sum_{j=i}^{v} \alpha_{ijk} x_i x_j + \sum_{i=1}^{v}\sum_{j=v+1}^{n} \beta_{ijk} x_i x_j$.

Then $F(a\mathbf{x})$ will be $f_k(a\mathbf{x})$ for all k.

$$\begin{aligned} f_k(a\mathbf{x}) &= \sum_{i=1}^{v}\sum_{j=i}^{v} \alpha_{ijk}(ax_i)(ax_j) + \sum_{i=1}^{v}\sum_{j=v+1}^{n} \beta_{ijk}(ax_i)(ax_j) \\ &= \sum_{i=1}^{v}\sum_{j=i}^{v} \alpha_{ijk}(a^2)x_i x_j + \sum_{i=1}^{v}\sum_{j=v+1}^{n} \beta_{ijk}(a^2)x_i x_j \\ &= a^2 \left(\sum_{i=1}^{v}\sum_{j=i}^{v} \alpha_{ijk} x_i x_j + \sum_{i=1}^{v}\sum_{j=v+1}^{n} \beta_{ijk} x_i x_j \right) \\ &= a^2 f_k. \end{aligned}$$

Thus, $f_k(a\mathbf{x}) \in F(a\mathbf{x})$ for all k, and so it follows that $F(a\mathbf{x}) = a^2 F(\mathbf{x})$. ■

Lemma 3 (Correctness for Changing Message Representative)**.** If $\texttt{Verify}(\texttt{pk}, M, \sigma) = 1$, *then* $\texttt{Verify}(\texttt{pk}, M', \sigma') = 1$

Proof. Let $\texttt{Verify}(\texttt{pk}, M, \sigma) = 1$, thus $P(\mathbf{x}) = M$. Observe $\texttt{Verify}(\texttt{pk}, M', \sigma')$; by the change message representative algorithm, $M' = a^2 M$ and $\sigma' = a\mathbf{x}$. Thus,

$$\begin{aligned} P(\sigma') &= \mathbf{U} \circ F \circ \mathbf{T}(a\mathbf{x}) \\ &= \mathbf{U} \circ F \circ a\mathbf{T}(\mathbf{x}) \text{ because } \mathbf{T} \text{ is linear} \\ &= \mathbf{U} \circ a^2 F \circ \mathbf{T}(\mathbf{x}) \text{ by Lemma 2} \\ &= a^2 \mathbf{U} \circ F \circ \mathbf{T}(\mathbf{x}) \text{ because } \mathbf{U} \text{ is linear} \\ &= a^2 P(\mathbf{x}) \\ &= a^2 M \\ &= M'. \end{aligned}$$

Therefore, $\texttt{Verify}(\texttt{pk}, M', \sigma')$= 1. ■

4.1.1.2 Unforgeability

The unforgeability of UOV*-MS is based on the unforgeability of UOV. Let the parameters, `pk`, and `sk` be generated correctly as described in our scheme, and let $Q = \{(\mathbf{x}, \mathbf{y}) : P(\mathbf{x}) = \mathbf{y}\}$. We claim UOV*-MS is forgeable only-if UOV is forgeable. This implies UOV*-MS is no more forgeable than UOV.

Suppose UOV*-MS is forgeable. An adversary would then be able to create a valid message-signature pair $(\mathbf{x}^*, \mathbf{y}^*)$ where $(\mathbf{x}^*, \mathbf{y}^*) \notin Q$ and $\mathbf{y}^* \notin [\mathbf{y}]_{\mathcal{R}}$ so that $\texttt{Verify}(\texttt{pk}^*, M^*, \sigma) = 1$. In our algorithms, this implies $P^*(\mathbf{x}^*) = \mathbf{y}^*$. This means $P^* \circ \mathbf{S}(\mathbf{S}^{-1}\mathbf{x}^*) = \mathbf{T} \circ F \circ \mathbf{U} \circ \mathbf{S}(\mathbf{S}^{-1}\mathbf{x}^*) = \mathbf{T} \circ F \circ \mathbf{U}(\mathbf{x}^*) = \mathbf{y}^*$. If $\mathbf{T} \circ F \circ \mathbf{U}(\mathbf{x}^*) = \mathbf{y}^*$, then a valid forged signature has been created. However, this then contradicts the unforgeability of UOV. So, since the unforgeability of UOV*-MS is based on the unforgeability of UOV, then UOV*-MS is no more forgeable than UOV.

4.2 Delegatable Credential with Message-Origin Hiding

In this section we will construct a semi-anonymous delegatable credential (Definition 4.2.1) based on UOV*-MS. This scheme inherits the origin-hiding and public key class-hiding properties from UOV*-MS. It also introduces message class-hiding by introducing a randomization algorithm. However, it is important to note that both the public key class-hiding and message class-hiding are only guaranteed up to the security of C^*. It is important to note here that C^* does take some computational power to break and thus we will have public key class-hiding and message class-hiding as long as there isn't strong computational power.

4.2.1 Relevant Definitions

Definition 4.2.1 (Semi-Anonymous Delegatable Credential)**.** A *semi-anonymous delegatable credential (Semi-DAC)* scheme is equivalent to a delegatable anonymous credential in all respects except that anonymity is only obfuscated from an attacker, rather than secured.

Definition 4.2.2 (Obfuscated)**.** Let $\mu(x)$ be an algorithm of complexity $\geq O(n)$ where $x \sim n$. Let ω be a function such that $\omega(y, x) \to 1$ if $\mu(x) = y$, otherwise $\omega(y, x) \to 0$. μ is *obfuscated* from ω if the complexity of $\omega(y, x) > O(n^2)$.

4.2.2 Semi-DAC from UOV*-MS Algorithms

The Delegatable Credential that we introduce in this section has the properties for message origin-hiding, but it does not have public key class-hiding, which is why it is labeled as a semi-anonymous credential. The following construction is based on the construction given by Crites and Lysyanskaya in [8].

Before giving the construction, it is important to mention that at each step of this algorithm, when a credential is being issued and a credential chain is being passed along, there are verifications being done. These verifications are done through a zero knowledge interactive protocol between the issuer and receiver of a credential. This is meant to serve as a proof that the receiver is who they say they are and they do have information about the secret key.

4.2.2.1 Randomization

Let a list of public keys be represented by $\mathcal{P} = (P^{(0)}, \dots, P^{(\ell-1)})$, a list of randomized messages be represented by $\mathcal{M} = (M^{(1)}, \dots, M^{(\ell)})$, and a list of randomized signatures is represented be $\mathcal{S} = (\sigma^{(1)}, \dots, \sigma^{(\ell)})$. A level ℓ user has a credential chain $(\mathcal{P}_\ell, \mathcal{M}_\ell, \mathcal{S}_\ell)$. such that $P^{(i-1)}(\sigma^{(i)}) = M^{(i)}$. Let $\texttt{sk}_C$ and $\texttt{sk}_U$ represent a user's C^* and UOV secret keys respectively. We will define `Randomize` as the following function:

$\texttt{Randomize}(\mathcal{P}, \mathcal{M}, \mathcal{S}, \texttt{sk}_C) \rightarrow (\mathcal{P}', \mathcal{M}', \mathcal{S}')$: On input of a credential chain $(\mathcal{P}, \mathcal{M}, \mathcal{S})$ and secret key $\texttt{sk}_C$, compute $P'^{(i)} = \Pi_1 \circ P^{(i)} \circ \Pi_2$, $M'^{(i)} = a^2(M^{(i)})$, and $\sigma'^{(i)} = a(\mathbf{x}^{(i)})$. Output $(\mathcal{P}', \mathcal{M}', \mathcal{S}')$.

4.2.2.2 Issuing a Credential

Let "||" denote appending an element to a list. Let

$$(\mathcal{P}_\ell, \mathcal{M}_{\ell+1}, \mathcal{S}_{\ell+1}) := \texttt{Randomize}(\mathcal{P}||P^{(\ell)}, \mathcal{M}||P^{(\ell+1)}, \mathcal{S}||\sigma^{(\ell+1)}, \texttt{sk}_C).$$

The following algorithm is employed for the `Issue/Receive` transaction:

$\texttt{Issue}(\mathcal{P}, \mathcal{M}, \mathcal{S}, \texttt{pk}, \texttt{sk}_C, \texttt{sk}_U) \leftrightarrow \texttt{Receive}(\texttt{nym}) \rightarrow (\mathcal{P}_\ell, \mathcal{M}_{\ell+1}, \mathcal{S}_{\ell+1})$: On input credential chain $(\mathcal{P}, \mathcal{M}, \mathcal{S})$, public key $\texttt{pk}$, C^* secret key $\texttt{sk}_C$, UOV secret key $\texttt{sk}_U$, and receiver's public key $\texttt{nym} = M^{(\ell+1)}$, the issuer computes $\texttt{Sign}(\texttt{sk}_U, \texttt{nym}) \rightarrow \sigma^{(\ell+1)}$. Output $(\mathcal{P}_\ell, \mathcal{M}_{\ell+1}, \mathcal{S}_{\ell+1})$.

4.2.2.3 Proof of Possession of a Credential

The following algorithm is employed for the `Prove/Verify` transaction:

`Prove`$(\mathcal{P}, \mathcal{M}, \mathcal{S}, \mathtt{sk}_C) \leftrightarrow$ `Verify` $\rightarrow 0/1$: On input of credential chain $(\mathcal{P}, \mathcal{M}, \mathcal{S})$, the prover calculates `Randomize`$(\mathcal{P}, \mathcal{M}, \mathcal{S}, \mathtt{sk}_C) \rightarrow (\mathcal{P}', \mathcal{M}', \mathcal{S}')$. The verifier outputs `Verify`$(\mathcal{P}', \mathcal{M}', \mathcal{S}') \rightarrow 0/1$. On result 1, the credential is valid. On result 0, the credential is invalid.

4.3 Parameters

The parameters for this signature scheme are based on the parameters of UOV for the compact and expanded case for the public key, secret key, and signature size. Note here that our public key and secret key are the only parameters that are compact or expanded. [4]

	NIST S.L.	n	m	q	ℓ	$\lvert\mathtt{epk}\rvert$ (bytes)	$\lvert\mathtt{esk}\rvert$ (bytes)	$\lvert\mathtt{cpk}\rvert$ (bytes)	$\lvert\mathtt{csk}\rvert$ (bytes)	$\lvert\sigma\rvert$ (bytes)
uov-IP	1	112	44	256	3	835,296	713,688	130,728	144	128
uov-IS	1	160	64	16	3	1,236,480	1,046,112	199,728	144	96
uov-III	3	184	72	256	3	3,676,320	3,132,960	567,696	144	200
uov-S	5	244	96	256	3	8,608,320	7,310,112	1,340,976	144	260

Table 4.3.1: This table shows public key, secret key, and signature size in bytes for a credential with chain length 3

	NIST S.L.	n	m	q	ℓ	$\lvert\mathtt{epk}\rvert$ (bytes)	$\lvert\mathtt{esk}\rvert$ (bytes)	$\lvert\mathtt{cpk}\rvert$ (bytes)	$\lvert\mathtt{csk}\rvert$ (bytes)	$\lvert\sigma\rvert$ (bytes)
uov-IP	1	112	44	256	4	1,113,728	951,584	174,304	192	128
uov-IS	1	160	64	16	4	1,648,640	1,394,816	266,304	192	96
uov-III	3	184	72	256	4	4,901,760	4,177,280	756,928	192	200
uov-S	5	244	96	256	4	11,477,760	9,746,816	1,787,968	192	260

Table 4.3.2: This table shows public key, secret key, and signature size in bytes for a credential with chain length 4

	NIST S.L.	n	m	q	ℓ	$\lvert\texttt{epk}\rvert$ (bytes)	$\lvert\texttt{esk}\rvert$ (bytes)	$\lvert\texttt{cpk}\rvert$ (bytes)	$\lvert\texttt{csk}\rvert$ (bytes)	$\lvert\sigma\rvert$ (bytes)
uov-IP	1	112	44	256	5	1,392,160	1,189,480	217,880	240	128
uov-IS	1	160	64	16	5	2,060,800	1,743,520	332,880	240	96
uov-III	3	184	72	256	5	6,127,200	5,221,600	946,160	240	200
uov-S	5	244	96	256	5	14,347,200	12,183,520	2,234,960	240	260

Table 4.3.3: This table shows public key, secret key, and signature size in bytes for a credential with chain length 5

Chapter 5

Conclusion and Future Work

5.1 Efficiency

In this current scheme, there is an issue with efficiency because at every level of the DAC, there is a new credential created and added to the credential chain. This credential gets passed down to each user thereafter and helps in verification at each of the levels. This means that the more levels there are in delegating a credential, the larger the credential chain is. A future goal here would be to find a way to verify a credential with minimal information, so that the chain does not continue to grow with the level and only keeps the level above it and the credential they were given to verify the credential is correct and create their own. This would minimize the length of information being sent to the lower levels of the credential levels and would minimize the operations that are needed to verify and create a new credential.

5.2 Possibility to get rid of C*

C^* is broken was broken in [15], but was chosen to "scramble" the message and key representatives because it is a cryptosystem whose plaintext and ciphertext messages are the same length. A future goal would be to use another modifier with similar properties that is not broken in order to gain full anonymity for the Delegatable Anonymous Credentials.

5.3 General Multivariate Cryptography - MAYO

MAYO was first introduced by Ward Beullens in 2021[3]. This scheme has been submitted to the NIST competition. Much like the UOV* Mercurial Signature, it relies on UOV as well as what is called a whipped modifier. There have been a few attacks on MAYO, but none of them have been successful at completely breaking the security. My future plans involve looking at new possible attacks for MAYO and the complexities of these attacks. The focus will be on the MinRank and Rectangular MinRank attack. Some of these attacks have been explored before under certain parameters in [9], so I will be looking at possible ways to modify these attacks in order to decrease the complexity of these attacks. These attacks focus on using the public key, which is similar to that of UOV.

Appendices

Appendix A Clemson ID Application for Semi-DAC

One practical application of the Semi-DAC with UOV*-MS is assigning credentials to enter buildings at a university. The university in question starts as our root of authority having the overall credential allowing anyone working at or attending the university to enter buildings. The credential next gets delegated to smaller programs focusing on undergraduate students, graduate students, faculty, and staff. The undergraduate authority can then be delegated again by dorm, major/classes being taken that year or semester, special labs, and etc. The graduate student authority can be delegated by department and office space. The faculty and staff authority can be delegated by department and buildings they work in. All of these then go through another split giving credentials for times of day that everyone can access each building. There can then be another layer of delegations for specific situations for different universities such as building access for graduate students that proctor in a different building than they teach/take classes. The chains are then sent to campus security who then assigns a chain of credentials to each student and employee. These credentials are paired with each university ID card whether it be a physical card or a digital card on their phone. The credentials are used when a student or employee scans their ID to get into each building on campus.

Clemson University ID's:

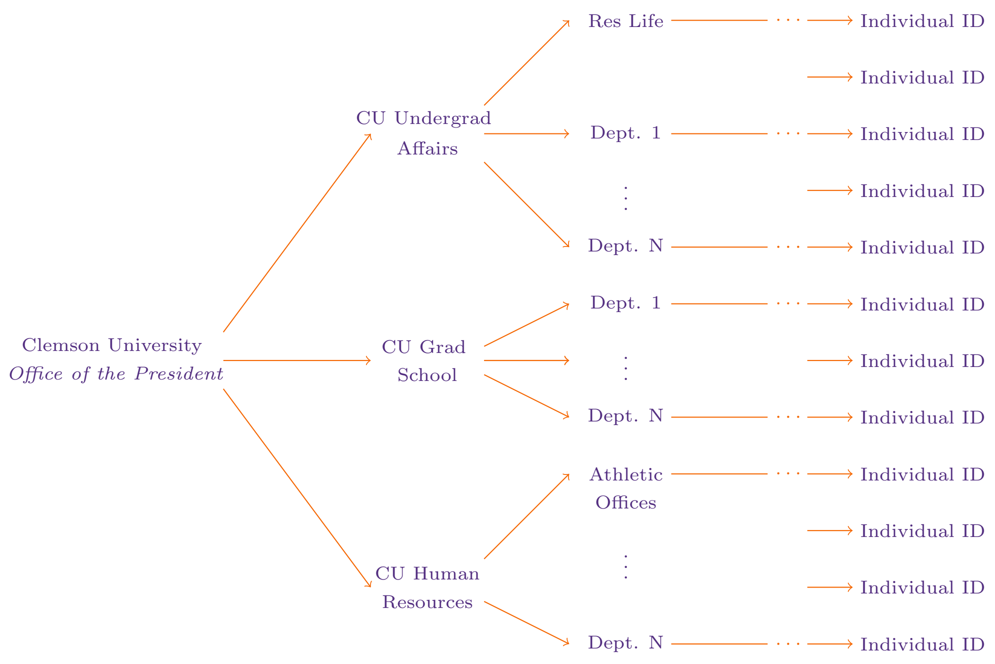

Figure A.1: This figure shows a visual representation of the multiple levels that our credential chain for a CU ID passes through.

Appendix B Possible Attacks

Identity Fraud:

It is possible on a college campus for students to lose their ID card, whether they drop it, misplace it, forget it somewhere, or someone just out right takes it. This ID card is connected to all credentials involving that student: student access to buildings, student access to restricted labs/classrooms, student access to their dorm, their meal plan to buy food, and etc. If a student looses their ID for any of the above reasons or another, it is possible for another student, worker at the university, or even community member to find and use this card and its credentials as if they were said student. So, in this instance, the adversary would be trying to gain some sort of credential for Clemson University and has done it by pretending to be a student.

Grand Theft Mobile:

Because technology is advancing, at most schools, it is possible that ID cards can be found on some sort of mobile device. In this case the adversary steals the mobile device and then has access to the ID. There might be some protocols that can be put on phones in order to have to open the phone to use the ID card, but in most cases the ID is added to a wallet and doesn't require the phone to be unlocked in order to scan and use its credentials like a credit/debit card.

Digging For Credentials:

There are also non-physical attacks such as this digging for credentials attack that is more cyber security esque. This attack involves an adversary on their computer who has hacked into the University's data frame and is searching for credentials. In this case the adversary is interested in discovering what credentials are assigned to what students vs professors and faculty and staff.

Credential Giver/Killer:

This is also a non-physical attack, where similar to the above attack, the adversary is accessing the university data frame. But, the goal here of the adversary is different, instead of focusing on who already has what credentials, this adversary is trying to decide which credential is which and give access/restrict access to this credential to different people. A practical example of this would be some student Jess who wants to be able to stay and study inside her math building and classrooms until midnight. If she doesn't have this credential she might pay a friend Eve who is good at cracking code to give her this credential so that she can stay and study after hours.

Appendix C Voting Scheme Application for Semi-DAC

One application of a Delegatable Credential with Origin Hiding is to distribute voting credentials. This example refers to credentials that are delegated down a chain, starting with the US government, to grant a voting ID to a US citizen. This credential chain involves verifying things such as US citizenship, state residency, and county occupancy. For the sake of simplicity, we will only consider these 3 credentials.

Government official Alice receives a credential directly from the government that identifies potential voters as occupants of the US. Alice then gives a credential to a state official Bob that identifies these potential voters as a US citizen. Bob then gives a credential to a county representative Carol, that specifies a voter as a resident of the state. Carol can then send this chain of credentials as well as one of her own specifying county to some other third party government official Dave who sends a voting ID out to Emma. Because of pseudonyms attached to messages and message origin hiding, Dave does not know what state or county Emma is from, just that she can vote. Emma can now use her voting ID (which is some credential chain) as she votes in her next local or federal election.

It is important to note that in this example each person is known to the government under a pseudonym that corresponds to all of the messages that are describing some credential that they possess and the voting ID is the signature. Because we have message origin hiding here, just as Dave who was issuing the voting ID could not access any information about Emma, when a voting ID is verified by some third party, they will also not be able to track down any of Emma's personal information about where she's from. Thus, we have this partial anonymity in our voting scheme.

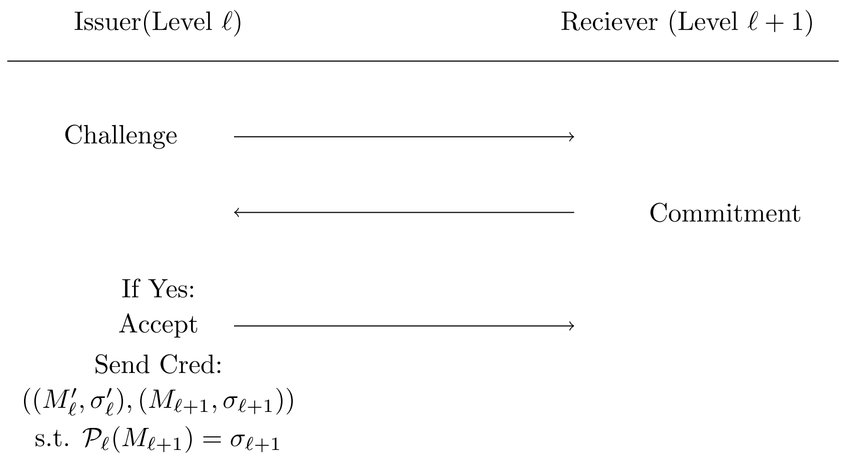
Issuer(Level ℓ)
Reciever (Level $\ell + 1$)
Challenge
Commitment
If Yes:
Accept
Send Cred:
$((M'_\ell, \sigma'_\ell), (M_{\ell+1}, \sigma_{\ell+1}))$
s.t. $\mathcal{P}_\ell(M_{\ell+1}) = \sigma_{\ell+1}$

Figure C.1: This figure shows a visual representation of the zero knowledge protocol scheme used by the issuer and receiver when checking to see if the receiver is who they say they are before sending a credential

Voting ID Scheme:

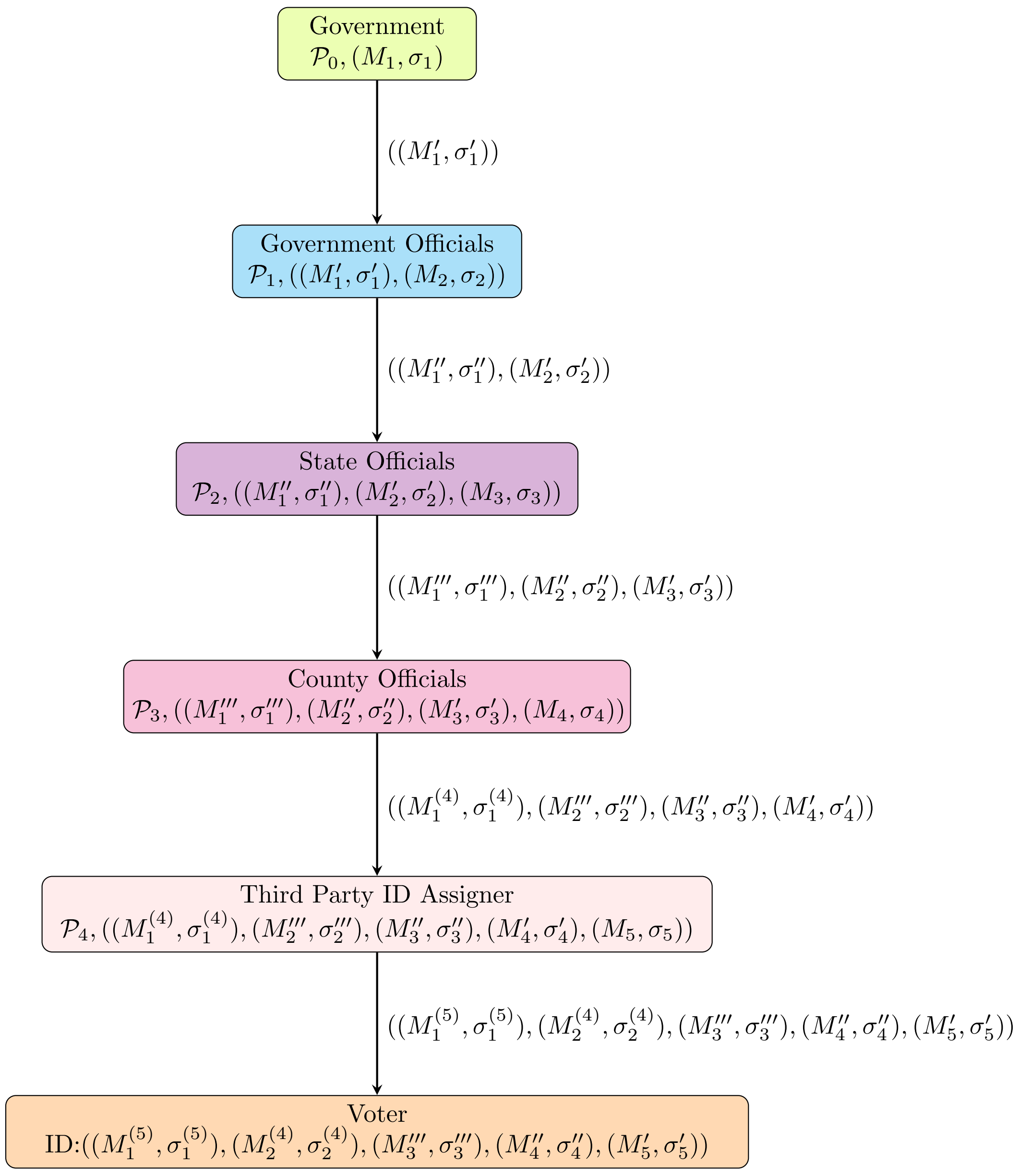
Government
$\mathcal{P}_0, (M_1, \sigma_1)$
$((M_1', \sigma_1'))$
Government Officials
$\mathcal{P}_1, ((M_1', \sigma_1'), (M_2, \sigma_2))$
$((M_1'', \sigma_1''), (M_2', \sigma_2'))$
State Officials
$\mathcal{P}_2, ((M_1'', \sigma_1''), (M_2', \sigma_2'), (M_3, \sigma_3))$
$((M_1''', \sigma_1'''), (M_2'', \sigma_2''), (M_3', \sigma_3'))$
County Officials
$\mathcal{P}_3, ((M_1''', \sigma_1'''), (M_2'', \sigma_2''), (M_3', \sigma_3'), (M_4, \sigma_4))$
$((M_1^{(4)}, \sigma_1^{(4)}), (M_2''', \sigma_2'''), (M_3'', \sigma_3''), (M_4', \sigma_4'))$
Third Party ID Assigner
$\mathcal{P}_4, ((M_1^{(4)}, \sigma_1^{(4)}), (M_2''', \sigma_2'''), (M_3'', \sigma_3''), (M_4', \sigma_4'), (M_5, \sigma_5))$
$((M_1^{(5)}, \sigma_1^{(5)}), (M_2^{(4)}, \sigma_2^{(4)}), (M_3''', \sigma_3'''), (M_4'', \sigma_4''), (M_5', \sigma_5'))$
Voter
ID:$((M_1^{(5)}, \sigma_1^{(5)}), (M_2^{(4)}, \sigma_2^{(4)}), (M_3''', \sigma_3'''), (M_4'', \sigma_4''), (M_5', \sigma_5'))$

Figure C.2: This figure shows a visual representation of the above voting scheme example and its corresponding credential chain.

Bibliography

[1] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 108–125, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[2] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In *Advances in Cryptology-CRYPTO 2009: 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 108–125. Springer, 2009.

[3] Ward Beullens. Mayo: Practical post-quantum signatures from oil-and-vinegar maps. Cryptology ePrint Archive, Paper 2021/1144, 2021. `https://eprint.iacr.org/2021/1144`.

[4] Ward Beullens, Ming-Shing Chen, Jintai Ding, Boru Gong, Matthias J. Kannwischer, Jacques Patarin, Bo-Yuan Peng, Dieter Schmidt, Cheng-Jhih Shih, Chengdong Tao, and Bo-Yin Yang. UOV: Unbalanced oil and vinegar - algorithm specifications and supporting documentation version 1.0, 2023.

[5] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable signatures: Complex unary transformations and delegatable anonymous credentials. *IACR Cryptol. ePrint Arch.*, page 179, 2013.

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

[7] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *Advances in Cryptology-CRYPTO 2006: 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006. Proceedings 26*, pages 78–96. Springer, 2006.

[8] Elizabeth C Crites and Anna Lysyanskaya. Delegatable anonymous credentials from mercurial signatures. In *Topics in Cryptology–CT-RSA 2019: The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4–8, 2019, Proceedings*, pages 535–555. Springer, 2019.

[9] Hiroki Furue and Yasuhiko Ikematsu. A new security analysis against mayo and qr-uov using rectangular minrank attack. In Junji Shikata and Hiroki Kuzuno, editors, *Advances in Information and Computer Security*, pages 101–116, Cham, 2023. Springer Nature Switzerland.

[10] Saqib A. Kakvi, Keith M. Martin, Colin Putman, and Elizabeth A. Quaglia. Sok: Anonymous credentials. In Felix Günther and Julia Hesse, editors, *Security Standardisation Research*, pages 129–151, Cham, 2023. Springer Nature Switzerland.

[11] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced oil and vinegar signature schemes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 206–222, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[12] Aviad Kipnis and Adi Shamir. Cryptanalysis of the oil and vinegar signature scheme. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, pages 257–266, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[13] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In D. Barstow, W. Brauer, P. Brinch Hansen, D. Gries, D. Luckham, C. Moler, A. Pnueli, G. Seegmüller, J. Stoer, N. Wirth, and Christoph G. Günther, editors, *Advances in Cryptology — EUROCRYPT '88*, pages 419–453, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[14] J. PATARIN. The oil and vinegar signature scheme. *Presented at the Dagstuhl Workshop on Cryptography, September 1997. transparencies*, 1997.

[15] Jacques Patarin. Cryptanalysis of the matsumoto and imai public key scheme of eurocrypt'88. In Don Coppersmith, editor, *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, volume 963 of *Lecture Notes in Computer Science*, pages 248–261. Springer, 1995.

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

[17] Eligijus Sakalauskas. The multivariate quadratic power problem over zn is np-complete. *Inf. Technol. Control.*, 41(1):33–39, 2012.

[18] Peter W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[19] Douglas R. Stinson and Maura B. Paterson. *Cryptography: Theory and practice.* CRC Press, 4th edition, 2019.