Advanced Isogeny-based Cryptosystems

Yi-Fu Lai

University of Auckland

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in mathematics, the University of Auckland, 2023.

October 2023

Abstract

Cryptography has a rich history, spanning thousands of years and evolving from ancient techniques such as the scytale and Caesar's cipher to modern systems like RSA, DHKE, and ECDH. However, quantum computing poses a substantial threat to the security of these modern cryptosystems. To address this challenge, post-quantum cryptography has emerged, with various branches including hash-based cryptography, code-based cryptography, multivariate cryptography, lattice-based cryptography, and isogeny-based cryptography.

Isogeny-based cryptography, a promising and relatively new area of research in post-quantum cryptography, employs algebraic mappings between elliptic curves to create cryptographic systems. Despite recent advancements challenging the hardness of the SIDH problem, an isogeny problem with additional information, several cryptosystems remain secure and continue to flourish. Isogeny-based cryptography is a vibrant and active research field.

This thesis delves into the fascinating world of advanced isogeny-based cryptosystems, discussing their primitives, challenges, and innovative approaches to their development. Topics covered include oblivious transfers, ring signatures, group signatures, blind signatures, verifiable random functions, and the application of generic proof systems to isogenies.

Concretely and first, we present the first efficient UC-secure oblivious transfer using only a constant number of isogeny computations based on the group action inverse problem (GAIP). To prove this, we propose a new assumption, the reciprocal CDH assumption, and show the equivalence to the GAIP. Second, we present the first post-quantum accountable ring signature, which immediately implies the first efficient isogeny-based group signature with proof size logarithmic in the number of members. Here, we also show how to use the Katz-Wang method to obtain a tight-secure variant, which is a less explored feature in the post-quantum group/ring signature literature. Third, we present the first provably secure blind signatures from isogenies based on the GAIP. Here, we present a novel approach to optimize the result by proposing a new assumption, the ring-GAIP. We also give a thorough analysis of it and show the equivalence to the GAIP for a few cases. Fourth, we present the first provably secure verifiable random functions from isogenies based on the standard DDH assumption. To prove this, we propose a generalized DDH assumption, the master DDH assumption, and show the equivalence to the DDH problem. Here, we also give a new use of the quadratic twist and relax the assumption to optimize the performance. Finally, we present the first practical application of generic proof systems to the isogeny construction. Here, we consider the identification scheme for an isogeny problem with a smooth degree.

All presented constructions have advantages over previously published schemes in terms of the security notions achieved or the performance or both. By offering a thorough analysis of these cryptosystems, this thesis lays a solid foundation for those new to the subject. It equips readers with a comprehensive understanding of the principles and potential applications of isogeny-based cryptosystems, fostering further research and development in this exciting area of post-quantum cryptography.

Delicated to the people and the homeland I love.

Acknowledgements

I am deeply grateful to many individuals who have helped me throughout my PhD journey. First and foremost, I would like to express my sincere appreciation to my supervisor Steven Galbraith. His endless patience, optimism, and kindness have been invaluable to me, and I appreciate the latitude he has afforded me to explore my whimsical research interests. Despite the difficulties posed by the pandemic, I consider myself fortunate to be offered the chance by Steven and to have crossed paths with so many remarkable individuals, to have witnessed awe-inspiring landscapes and to complete my supersingular journey in this wonderful land of Aotearoa.

Also, many thanks to previous supervisors Jiun-Ming Chen and Chen-Mou Cheng, who invited me to the group when I was an inexperienced lad during my master program. I would like to express my gratitude to Bo-Yin Yang who introduced me to isogeny cryptography. Without their introduction, I would never encounter the beauty of cryptography.

I am indebted to the academics and co-authors whom I had the privilege of meeting during my PhD journey. Without the contributions of each of them, this thesis would be incomplete. I would like to extend special thanks to Dr. Shuichi Katsumata and Dr. Christophe Petit, whose insights and innovative approaches have served as a constant source of inspiration to me. Your rigorous approach to academic writing has also inspired me to strive for excellence in my own work. Special thanks to Dr. Fehr, Dr. Petit, Dr. Castryck, and Dr. de Feo, Dr. Matsuda (and also Steven's MBIE funding) for hosting my visits to Europe and Japan, and to Mingjie, Yu-Hsuan, Simon, Keitaro for their company during those visits. The trips and your hospitality have made a significant difference in my academic journey.

My profound gratitude extends to Jasmine, Rolland, and Lydia, whose presence rendered my stay in New Zealand an exuberant and indelible tapestry of memories. During the challenging lockdown periods, their positive attitudes and infectious laughter lifted my spirits and kept me going. Together, we explored many new places, from the rugged coastlines of the South Island to the vibrant cityscape of Auckland. I will always cherish the moments we spent together exploring new places, trying new restaurants and snacks, and playing Switch games at night. Beyond just the experiences we shared, they enriched my life with their warmth, humor, generosity and a cup of yogost. Time is a fleeting muse. I hold the hope that our paths shall converge once more on a voyage yet to be charted.

I am also grateful for the wonderful colleagues I met in the math department during my PhD, including Zhaochen, Ling, Felix, Kelong, Susan, Annabelle, Trey, and Samuel, who made my time in the department so enjoyable. Being in your company made me want to stay in the department every day, even when the work was challenging. May the stars align once more in the weave of our futures, allowing our paths to intertwine anew.

Finally, I would like to offer a special thank you to my mom. I appreciate all the love and support you have shown me over the years. Without your belief in me, I would not have been able to pursue my dreams and achieve my goals.

This thesis represents my love for isogeny cryptography and the passion that drove me in pursuing them. It is an account of my exciting journey into the world of cryptosystems through isogenies. I dedicate this thesis to the people and my homeland that hold a special place in my heart and it is my way of expressing gratitude to those who have supported and inspired me along the way.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Military tactics are rooted in guile.

Sun Tzu, "The Art of War"

While the origins of human awareness of privacy remain uncertain, the need for information security can be traced back thousands of years through historical records. One of the earliest known ciphers, the scytale, was used by Spartans in ancient Greece to encrypt military messages. The scytale involved wrapping a parchment strip around a cylinder of a specific diameter and writing the message lengthwise on the parchment. When unwrapped, the message appeared scrambled and could only be deciphered by someone with a cylinder of the same diameter.

In the information age, cryptography has become an indispensable cornerstone due to the rapid development of the internet and communication technology. Most real-world cryptosystems are derived from RSA [RSA78], DHKE [DH76], or ECDH [Mil86, Kob87]. Cryptography is used in various applications beyond military use, such as securing online transactions in our daily lives and protecting sensitive government communications. As technology becomes increasingly essential in our lives, the need for robust and reliable cryptographic methods will continue to grow.

These cryptosystems are able to protect our private information effectively because it is difficult to factorize large numbers or compute logarithm functions in discrete spaces. However, these mathematical problems can theoretically be efficiently solved by a fully operational quantum computer [Sho99]. While we are still some distance away from having a quantum computer powerful enough to break these cryptosystems in practice, the threat is becoming more tangible with the quantum computing advancements [IBM, Goo23].

To address the growing threat of quantum computing to information security, new cryptography studies, known as *post-quantum cryptography*, are being developed. This field of study mainly consists of five branches: hash-based cryptography, code-based cryptography, multivariate cryptography, lattice-based cryptography, and isogeny-based cryptography. Each branch has its own distinct features and hardness assumptions. One of the most significant activities in post-quantum cryptography is the request from NIST for the standardization of post-quantum public key encryption and digital signature schemes, which began in 2016. After eight years, in 2022, NIST announced four algorithms that have been chosen for standardization. These algorithms are CRYSTALS-KYBER [SAB+22], CRYSTALS-DILITHIUM [LDK+22], FALCON [SAB+22] from lattice-based cryptography, and SPHINCS+ [HBD+22] from hash-based cryptography. Another round of selection for post-quantum public key encryption is currently ongoing, and there is also a call for more post-quantum signature schemes due to rapid advancements in the field.

Isogeny-based cryptography is a relatively new and active area of research within post-quantum cryptography. Isogenies refers to a specific algebraic mapping between elliptic curves, which was initially introduced in elliptic curve cryptography as a tool for cryptanalysis [Gal99, GHS02]. However, isogeny has since evolved into a crucial component of cryptographic systems itself. It was first introduced with the CRS key exchange [Cou06, RS06] and the CGL hash function [CLG09]. The core assumption in isogeny-based cryptography is that it is hard to recover an isogeny between two given isogenous elliptic curves. This is known as the isogeny problem. In fact, the study of the isogeny problem can be traced back to 1999 [Gal99]. One of the most well-known isogeny-based cryptosystems is SIDH [JD11], which is a public key encryption that survived until the fourth round in the NIST standardization program [JAC+22]. Although recent advancements [CD23, Rob23, MMP+23] have falsified the hardness of the SIDH problem, which is a relaxed isogeny problem, its 11-year longevity and iconic status have spurred significant research into isogeny-based cryptography [CLM+18, DG19, BKV19, DG19, MCR19, Pei20, BS20, DKL+20, EKP20, DM20, LGD21, BDK+22, AEK+22, Ler22, BKL+22, DLLW23, DFK+23, CLL23, Lai23]. As the field continues to evolve, it is expected that more advancements and innovations will emerge.

Despite the vulnerability of SIDH, the original isogeny problem is still considered hard, and several cryptosystems continue to be based on the original assumption [CLG09, DKL+20, CLL23]. There is also a group action version of isogeny-based cryptography, called CSIDH, proposed by [CLM+18]. While it offers limited operations as the evaluation of the action is restricted to generating sets with small cardinality, it still resulted in the first secure and practical post-quantum non-interactive key exchange. With optimization advancements [BKV19, DFK$^+$23], the CSIDH group action is becoming more flexible. Despite a known subexponential vulnerability [Reg04, Kup05, Kup11, Pei20, BS20], recent research continues to demonstrate the competitiveness of isogeny-based cryptography as a post-quantum branch, including signature schemes [BKV19, EKP20, DG19], UC-secure oblivious transfers [LGD21, BMM$^+$22], threshold signatures [DM20], (linkable/accountable) ring and group signatures [BKP20, BDK$^+$22], and PAKE [AEK$^+$22].

In this thesis, we present six advanced isogeny-based cryptosystems: oblivious transfers, ring signatures, group signatures, blind signatures, verifiable random functions, and proof of isogeny knowledge. By providing an in-depth analysis of these cryptosystems, this thesis establishes a strong foundation for beginners in the subject. It offers readers a comprehensive understanding of the principles and potential applications of isogeny-based cryptosystems, paving the way for further research and development in this exciting area of post-quantum cryptography.

Chapter 2

Mathematical Preliminaries

Notation.

We let $\mathbb{N}$ represent the set of natural numbers, $\mathbb{Z}$ be the ring of integer and $\mathbb{Z}_N := \mathbb{Z}/N\mathbb{Z}$. Let O be the point at infinity of an elliptic curve. Let $\mathbb{F}_p$ denote a finite field of order p. For a finite field $\mathbb{F}$, $\bar{\mathbb{F}}$ represents its algebraic closure. We let $[N]$ denote $\{1, 2, \cdots, N\} \subset \mathbb{N}$. For a set S, $s \leftarrow S$ means uniformly sampling an element, s, from S. When displaying pseudocodes, we will mark the sampling as $\overset{\$}{\leftarrow}$ to distinguish the use of $\leftarrow$ of a subroutine or assigning values.

Let λ be the security parameter. Two probability ensembles X_λ, Y_λ are said to be *computationally indistinguishable*, denoted by $X_\lambda \approx_c Y_\lambda$, if for every probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such $|\Pr[\mathcal{A}(X_\lambda) = 1] - \Pr[\mathcal{A}(Y_\lambda) = 1]| \leq \mathsf{negl}(\lambda)$. Also, X_λ, Y_λ, defined over the same set, are said to be *statistically indistinguishable*, denoted by $X_\lambda \approx_s Y_\lambda$, if there exists a negligible function $\mathsf{negl}(\lambda)$ such $\sum_a |\Pr[X_\lambda = a] - \Pr[Y_\lambda = a]| \leq \mathsf{negl}(\lambda)$.

2.1 Elliptic Curves over Finite Field and Isogenies

An elliptic curve defined a finite field $\mathbb{F}$, denoted by $E/\mathbb{F}$, is given by a non-singular (i.e. with non-zero discriminant) affine Weierstrass equation

$$E : y^2 + a_1xy + a_3y = x_3 + a_2x^2 + a_4x + a_6,$$

where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}$.

When $\mathsf{char}(\mathbb{F}) \neq 2, 3$, we can always transform it as the short Weierstrass equation $y^2 = x^3 + a'_4x + a'_6$ through a linear transform where $a'_4, a'_6 \in \mathbb{F}$ and $4a'^3_4 + 27a'^2_6 \neq 0$. Throughout this thesis, we consider only non-singular curves.

Let E defined over a field $\mathbb{F}$. (E, O) has an additive group structure where O represents the point at infinity and is the identity element of the group. Let $\mathbb{F}'$ be a field containing $\mathbb{F}$, $E(\mathbb{F}')$ is the set of point of E over $\mathbb{F}'$. For every $N \in \mathbb{Z}$, there is an endomorphism $[N] : E \to E$ defined by $P \mapsto NP$. For every $N \in \mathbb{N}$, the N-torsion subgroup of E is defined to be $E[N] = \{P \in E(\bar{\mathbb{F}}) | [N]P = \mathsf{O}\}$.

Theorem 2.1.1 (III.6. [Sil09])**.** *Given an elliptic curve $E/\mathbb{F}$ and $N \in \mathbb{N}$, we have the following results.*

1. *If $N \neq 0$ over $\mathbb{F}$, then $E[N] \cong \mathbb{Z}_N \times \mathbb{Z}_N$.*
2. *If $\mathsf{char}(\mathbb{F}) = p > 0$, then one of the following is true:*
   - *$E[p^i] = \{\mathsf{O}\}$ for all $i \in \mathbb{N}$.*
   - *$E[p^i] \cong \mathbb{Z}_p^i$ for all $i \in \mathbb{N}$.*

Definition 2.1.2 (Supersingularity)**.** *Let $E/\mathbb{F}$ be an elliptic curve and $\mathsf{char}(\mathbb{F}) = p > 0$. E is said to be* supersingular *if $E[p^i] = \{\mathsf{O}\}$ for all $i \in \mathbb{N}$.*

Definition 2.1.3 (j-invariant)**.** *Suppose $\mathsf{char}(\mathbb{F}) \neq 2, 3$ and let $a_4, a_6 \in \mathbb{F}$ be such that $4a_4^3 + 27a_6^2 \neq 0$. The j-invariant of a short Weierstrass equation $E : y^2 = x^3 + a_4x + a_6$ is $j(E) = 1728\frac{4a_4^3}{4a_4^3+27a_6^2}$.*

Theorem 2.1.4 (Chapter V.3. [Sil09])**.** *Let E be an elliptic curve defined over a finite field $\mathbb{F}$ and $\mathsf{char}(\mathbb{F}) = p$. The following are equivalent.*

1. *E is supersingular.*
2. *The map $[p] : E \to E$ is purely inseparable and $j(E) \in \mathbb{F}_{p^2}$.*
3. *The endomorphism ring $\mathsf{End}(E)$ is an order in a quaternion algebra.*
4. *$|E(\mathbb{F})| = |\mathbb{F}| + 1 - t$ where $p \mid t$.*

Theorem 2.1.5 (Chapter III.1. [Sil09])**.** *Let $\mathbb{F}$ be a field and E_1, E_2 be non-singular elliptic curves defined over $\mathbb{F}$. There is an $\bar{\mathbb{F}}$-isomorphism from E_1 to E_2 if and only if $j(E_1) = j(E_2)$.*

We know $y^2 = x^3 + 1$ and $y^2 = x^3 + x$ are of j-invariant 0 and 1728 respectively. For every $J \in \mathbb{F}$ with $J \neq 0, 1728$,

$$y^2 = x^3 + \frac{3J}{1728 - J}x + \frac{2J}{1728 - J}$$

is an elliptic curve defined $\mathbb{F}$ of j-invariant J.

Definition 2.1.6 (Isogeny). *Let E_1, E_2 be elliptic curves defined over $\mathbb{F}$. An* isogeny *over $\mathbb{F}$ is a morphism $\phi : E_1 \to E_2$ defined over $\mathbb{F}$ preserving the point at infinity.*

For an isogeny **defined over** $\mathbb{F}$, $\phi : E_1 \to E_2$, we may write $\phi(x, y) = \left(\frac{f_1(x)}{f_2(x)}, y\frac{f_3(x)}{f_4(x)}\right)$ for some $f_1, f_2, f_3, f_4 \in \mathbb{F}[x]$ and f_1, f_2 have no common factors (see Sec. 2.9 [Was08]). We define the **degree** of ϕ, denoted by $\deg(\phi)$, to be the maximal degree among $f_1(x), f_2(x)$. That is, $\deg(\phi) = \max(\deg(f_1(x)), \deg(f_2(x)))$. If $N = \deg(\phi)$, then we say E_1 and E_2 are N**-isogenous**. There exists the dual isogeny, denoted by $\hat{\phi} : E_2 \to E_1$, of degree N such that $\hat{\phi} \circ \phi = [N]$, which represents the multiplication by N over E_1. Hence, E_2 and E_1 are also N-isogenous.

Theorem 2.1.7 (Hasse's Bound). *Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$. Then $||E(\mathbb{F}_q)| - q - 1| \leq 2\sqrt{q}$.*

Proposition 2.1.8 (Supersingularity over a prime field). *Let E be an elliptic curve defined over a prime field $\mathbb{F}_p$. Then, E is supersingular if and only if $|E(\mathbb{F}_p)| = p + 1$.*

Theorem 2.1.9 (Chapter V. [Sil09]). *Let E, E' be elliptic curves defined over a finite field $\mathbb{F}$. Then, E and E' are isogenous over $\mathbb{F}$ if and only if $|E(\mathbb{F})| = |E'(\mathbb{F})|$.*

Theorem 2.1.10 (Modular Polynomial (Sec 10.3.[Was08])). *For every positive integer $N \in \mathbb{N}$, there exists a polynomial $\Phi_N(X, Y) \in \mathbb{Z}[X, Y]$ of univariate-degree at most $N + 1$ such that $\Phi(j_1, j_2) = 0$ if any only if the elliptic curves of j-invariants j_1, j_2 are N-isogenous.*

Examples.

$$\begin{aligned}
\Phi_2(X, Y) = &- X^2Y^2 + X^3 + Y^3 + 2^4 \cdot 3 \cdot 31XY(X + Y) \\
&+ 3^4 \cdot 5^3 \cdot 4027XY - 2^4 \cdot 3^4 \cdot 5^3(X^2 + Y^2) \\
&+ 2^8 \cdot 3^7 \cdot 5^6(X + Y) - 2^{12} \cdot 3^9 \cdot 5^9, \\
\Phi_3(X, Y) = &\, X^4 - X^3Y^3 + 2232X^3Y^2 - 1069956X^3Y + 36864000X^3 \\
&+ 2232X^2Y^3 + 2587918086X^2Y^2 + 8900222976000X^2Y \\
&+ 452984832000000X^2 - 1069956XY^3 + 8900222976000XY^2 \\
&- 770845966336000000XY + 1855425871872000000000X + Y^4 \\
&+ 36864000Y^3 + 452984832000000Y^2 + 1855425871872000000000Y
\end{aligned}$$

2.2 CSIDH: Commutative Supersingular Isogeny Diffie-Hellman

This section is a brief overview of the isogeny group action CSIDH based on [CLM+18]. For a given prime p and a supersingular elliptic curve E defined over $\mathbb{F}_p$, $\mathsf{End}_p(E)$ is the subring of the endomorphism ring $\mathsf{End}(E)$ consisting of the endomorphisms defined over $\mathbb{F}_p$. We say two elliptic curves are in the same $\mathbb{F}_p$-isomorphism class if there exists an isogeny defined over $\mathbb{F}_p$ between them. Let $\mathcal{O}$ be an order in an imaginary quadratic field and $\gamma \in \mathcal{O}$ an element of norm p. Define the set of $\mathbb{F}_p$-isomorphism classes of elliptic curves $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$ where E defined over $\mathbb{F}_p$, $\mathsf{End}_p(E) \cong \mathcal{O}$, and π is the $\mathbb{F}_p$-Frobenius map of E corresponding to $\gamma \in \mathcal{O}$. For an ideal $\mathfrak{a} \in \mathcal{O}$ and $E \in \mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, an action can be defined by $\mathfrak{a} \star E = E'$ such that there exists an isogeny $\phi : E \to E'$ with $\mathsf{ker}(\phi) = \cap_{\alpha \in \mathfrak{a}} \{P \in E(\bar{\mathbb{F}}_p) \mid \alpha(P) = 0\}$. The image curve of $\mathfrak{a} \star E$ is well-defined up to $\mathbb{F}_p$-isomorphism. Moreover, the ideal class group $Cl(\mathcal{O})$ acts freely and transitively on $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$. Castryck et al. [CLM+18] chose the prime to be $p = 4 \times \ell_1 \times \cdots \times \ell_n - 1$ where ℓ_i are small odd primes. For a more comprehensive exploration of isogeny-based group actions, we direct readers to [Onu21]. It is also important to highlight the pioneering contributions by Couveignes, Rostovtsev, and Stolbunov [Cou06, RS06], who introduced isogeny-based key exchange methods using ordinary curves.

In the case of $p = 3 \mod 8$, for every supersingular elliptic curve E defined over $\mathbb{F}_p$, we have $\mathsf{End}_p(E) = \mathbb{Z}[\pi] \cong \mathbb{Z}[\sqrt{-p}]$ if and only if E is $\mathbb{F}_p$-isomorphic to $E_A : y^2 = x^3 + Ax^2 + x$ for some unique $A \in \mathbb{F}_p$. The *quadratic twist* of a given elliptic curve $E : y^2 = f(x)$ is $E^t : dy^2 = f(x)$ where $d \in \mathbb{F}_p$ has Legendre symbol -1. When $p = 3 \mod 4$ let E_0 be such that $j(E_0) = 1728$, then E_0 and E_0^t are $\mathbb{F}_p$-isomorphic. The quadratic twist can be efficiently computed in the CSIDH setting [CLM+18]. Since the prime $p = 3 \mod 4$, $E' : -y^2 = x^3 + Ax^2 + x$ is the quadratic twist of $E_A : y^2 = x^3 + Ax^2 + x$ and E' is $\mathbb{F}_p$-isomorphic to E_{-A} by $(x, y) \mapsto (-x, y)$. Further, $(a \star E_0)^t = a^{-1} \star E_0$. Therefore, for every curve $E \in \mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, we have, by the transitivity of the action,

$$(a \star E)^t = a^{-1} \star E^t.$$

In general, computing $a \star E$ for a random element $a \in \mathcal{C}\ell(\mathcal{O})$ can be computationally infeasible due to the difficulty of computing the kernel. However, the special form of the prime $p = 4 \times \ell_1 \times \cdots \times \ell_n - 1$ is advantageous for evaluation. For each prime ℓ_i, we can factorize the ideal $\ell_i\mathcal{O} = (\ell_i, \pi - 1)(\ell_i, \pi + 1)$ because $x^2 + p = (x+1)(x-1) \pmod{\ell_i}$ and we can restrict the factorization from the ring of integers to $\mathcal{O}$ (Prop 7.20 [Cox22]). The ideal $(\ell_i, \pi - 1)$ induces a kernel by collecting the ℓ_i-torsion points $E(\mathbb{F}_p)$ since $\pi(P) = P$ if and only if $P \in E(\mathbb{F}_p)$. Therefore, evaluating the action with $(\ell_i, \pi - 1)$ is feasible.

Similarly, the ideal $(\ell_i, \pi + 1)$ induces a kernel by collecting the point at infinity and the ℓ_i-torsion points over $E(\mathbb{F}_{p^2}) - E(\mathbb{F}_p)$ with x-coordinates defined over $\mathbb{F}_p$. This is because for a point $P \neq \mathsf{O}$, we have $\pi(P) = -P$ if and only if $P \in E(\mathbb{F}_{p^2}) - E(\mathbb{F}_p)$ and the x-coordinate is defined over $\mathbb{F}_p$. Therefore, both the actions of $\mathfrak{l}_i = (\ell_i, \pi - 1)$ and $\mathfrak{l}_i^{-1} = (\ell_i, \pi + 1)$ are feasible. For any small range of j, we can evaluate the action of $\mathfrak{l}_i^j$.

Under appropriate assumptions as made in [CLM+18], we can sample elements from $\mathcal{C}\ell(\mathcal{O})$ uniformly and heuristically at random and evaluate the action using the generating set $\mathfrak{l}_1, \ldots, \mathfrak{l}_n$ for $\mathcal{C}\ell(\mathcal{O})$.

In [BKV19], Beullens et al. determined the structure of $\mathcal{C}\ell(\mathcal{O})$ (under the parameter CSIDH-512) as a direct sum of cyclic groups, i.e., $\mathcal{C}\ell(\mathcal{O}) = \bigoplus_i \mathbb{Z}_{m_i}$. They also found a generator $\mathfrak{g}$ for the entire group, which is cyclic in their case. Moreover, they computed the reduced relation lattices between $\{\mathfrak{l}_1, \cdots, \mathfrak{l}_n\}$, which enables us to efficiently evaluate the action of a random element using the generator $\mathfrak{g}$. Specifically, we can uniformly sample a random element by selecting small coefficients j_i and computing $\mathfrak{g} = \mathfrak{l}_1^{j_1} \cdots \mathfrak{l}_n^{j_n}$. This way, we can make the evaluation feasible using the reduced relation lattice.

Throughout this thesis, we concentrate on supersingular curves defined over $\mathbb{F}_p$. Denote the ideal class group $Cl(\mathsf{End}_p(E))$ by Cl and the set of elliptic curves $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$ by $\mathcal{E}$.

2.3 Group Action Models

To ensure that the cryptosystems presented in this thesis are easily understandable, we have modified the abstract models by emphasizing the properties of isogeny group actions. This approach enables us to provide clear and intuitive explanations of the protocols we propose. In this section, we provide a concise introduction to the primary ingredient that underpins most of our protocols – group actions. Specifically, we focus on abelian, *free, transitive, and effective* group actions throughout our work.

These properties play a crucial role in the design and analysis of our cryptographic schemes. By adopting this approach, we aim to make our research accessible to a broad audience and to inspire new innovations in the field of cryptography.

Definition 2.3.1 (Group Action)**.** *Let $(G, \odot)$ be a group and $\star$ be a map $\star : G \times \mathcal{E} \to \mathcal{E}$. We denote $\star(g, x)$ as $g \star x$. $(G, \star)$ is said to* act *on a set $\mathcal{E}$ if it satisfies the following requirements.*

1. *Identity: if e is the identity element of G, then for every $E \in \mathcal{E}$, we have $e \star E = E$.*

2. *Compatibility: for every $g, h \in G$ and every $E \in \mathcal{E}$, we have $(g \odot h) \star E = g \star (h \star E)$.*

We say $(G, \mathcal{E}, \star)$ is a group action if $(G, \star)$ acts on $\mathcal{E}$. We may denote the group operation to be the additive notation $(g_1 + g_2)$ or the multiplication notation $(g_1 g_2)$ depending on the nature of the chapter, which we will state in each chapter.

Definition 2.3.2. *A group action $(G, \mathcal{E}, \star)$ is said to be*

1. transitive *if for every $x_1, x_2 \in \mathcal{E}$ there exists $g \in G$ such that $x_2 = g \star x_1$, or*

2. free *if for any $g \in G$, g is the identity element if and only if there exists some $x \in \mathcal{E}$ such that $x = g \star x$.*

A group action is said to be *regular* if it is both transitive and free. To construct efficient and practical schemes, we need to rely on efficient algorithms for various tasks. To this end, we make use of the *effective group action* framework proposed in [ADMP20] with a slight modification.

Definition 2.3.3 ((Modified) Effective Group Action)**.** *Let $(G, \mathcal{E}, \star)$ group action $(G, \mathcal{E}, E_0, \star)$ is* effective *if the following properties are satisfied:*

1. *The abelian group G is finite and there exist PPT algorithms for (i.) membership testing, (ii.) equality testing, (iii.) group operations, (iv.) element inversions (v.) unique string representation, and (vi.) a sampling method over G. The sampling method is required to be statistically indistinguishable from the uniform distribution over G.*

2. *The set $\mathcal{E}$ is finite, and there exist PPT algorithms for membership testing and generating a unique bit-string representation for every element in $\mathcal{E}$.*

3. *$E_0 \in \mathcal{E}$ is a distinguished element and the bit-string representation is publicly known.*

4. *There exists a PPT algorithm that, given any $(g, x) \in G \times \mathcal{E}$, outputs $g \star x$.*

We modify the definition of an effective group action of [ADMP20] by requiring every group element to have a *unique representation* (v.). A weaker model (restricted effective group action) restricts the feasible evaluation of the action to a generating set of small cardinality, which captures the original setting CSIDH where the action relies upon a generating set of several small-norm ideals. To keep this thesis accessible, we use the (modified) effective group action model for the presentations of these cryptosystems. We stress that group-action-based constructions in Chapters 4, 5 and 7 can be translated to the restricted group action setting by using Fiat-Shamir with aborts technique [Lyu09, DG19]. Remark that the translation of the results in EGA to REGA is not always clear in general [GPSV21, MZ22], which should be investigated case by case.

A more convenient and stronger model is the *known-order effective group action model* by assuming the group structure and the lattice relation of the action are known. Formally,

Definition 2.3.4 (Known-order Effective Group Action (KO-EGA))**.** *An EGA $(G, \mathcal{E}, E_0, \star)$ is a* known-order effective group action *if the following properties are satisfied:*

1. *The abelian group G is finite and the structure of $G \cong \oplus_{i=1}^{d} \mathbb{Z}_{m_i}$ and a minimal generating set $\langle g_i \rangle_{i=1}^{d} = G$ are known with an effective isomorphism $(e_1, \cdots, e_d) \in \oplus_{i=1}^{d} \mathbb{Z}_{m_i} \mapsto \Pi_{i=1}^{d} g_i^{e_i} \in G$. Hence, $\oplus_{i=1}^{d} \mathbb{Z}_{m_i}$ can serve as a standard representation for G.*

2. *The set $\mathcal{E}$ is finite and there exist PPT algorithms for the membership testing and generating a unique bit-string representation for every element in $\mathcal{E}$.*

3. *$E_0 \in \mathcal{E}$ is a distinguished element and its bit-string representation is publicly known.*

The isogeny-based threshold signature scheme [DM20] is based on this model. We will only use this structure to build the blind signature Chapter 6. While CSIDH-512 serves as an instantiation for the known-order effective group action, we present a potential concern in Sec. 2.5 regarding the feasibility of using larger parameter sets derived from isogenies.

Quadratic Twists. Aside from the abovementioned PPT algorithms, the last operation to remark is the quadratic twist with the property $(a \star E_0)^t = a^{-1} \star E_0$ when the underlying prime is $p = 3 \pmod 4$. The quadratic twist has been shown to be a useful tool in some cryptosystems [BKV19, EKP20, LGD21, AEK$^+$22, Lai23]. We will use quadratic twists in Chapters 4 and 6 and Sec. 7.6 and the twist is not required in the rest of the sections.

2.4 Standard Assumptions

This section introduces the *standard* computational and decisional assumptions in the literature. We start with the core hard problem in isogeny-based cryptography – the isogeny problem.

Problem 1 (Isogeny Problem)**.** *Given two isogenous supersingular elliptic curves, recover an isogeny between them.*

Generic Attacks on The Isogeny Problem.

To begin, let us first review the state-of-art classical and quantum algorithms that have been developed to tackle the isogeny problem. The most efficient classical algorithm is a meet-in-the-middle-type attack with time and space complexities of $\tilde{O}(\sqrt{p})$ [DG16] based on Galbraith's work [Gal99] on the ordinary case. The size of the graph in this algorithm is approximately $p/12$, which roughly explains the complexity bound. In contrast, the best quantum algorithm against the isogeny problem is a combination of [DG16] with the Grover's algorithm, resulting in a complexity of $\tilde{O}(p^{1/4})$ [BJS14]. This has not been changed for nearly ten years. We also refer [GHS02, GS13] for the ordinary case. Note that the complexity of the problem can vary based on the degree of the isogeny, and may be either easier or harder to solve. Specifically, the complexity of the SIDH problem, a specific instance of the isogeny problem, is $\Theta(p^{1/4})$ using the claw-finding algorithm (accelerated by the Grover's algorithm or quantum random walks) [Tan07, JS19], but this may change for different degrees.

We can translate the isogeny problem to the group action setting, known as the *group action inverse problem*, as follows where we use the multiplication notation for the group G.

Problem 2 (Group Action Inverse Problem (GAIP))**.** *Let $(G, \mathcal{E}, \star, E_0)$ be a group action with a distinguished element $E_0 \in \mathcal{E}$. Given E sampled from the uniform distribution over $\mathcal{E}$, the* GAIP *problem consists in finding an element $g \in G$ such that $g \star E_0 = E$.*

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathsf{GAIP}}_{(G,\mathcal{E},E_0,\star)}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$ where the probability is also taken over the randomness used in the experiment.

At first glance, the GAIP problem may seem identical to the original isogeny problem. However, the nuance lies in the structure of the space where isogenies are determined and sampled. It is worth noting that the best known algorithms against the classical security of both the isogeny problem and GAIP are the Pollard-rho-type algorithms [GHS02, GS13] where GAIP has a smaller isogeny graph due to the restriction to $\mathbb{F}_p$. However, the situation changes when we consider quantum cryptoanalysis.

Generic Quantum Attacks on GAIP.

The best quantum algorithm against GAIP is Kuperberg's algorithm [Kup05, Reg04, Kup11, Pei20, BS20]. Roughly speaking, say in the known-order effective group action model, there is a generator for the cyclic group $\langle \mathfrak{g} \rangle = G$ and given a challenge E to find $a \in \mathbb{Z}$ such that $E_0 = [\mathfrak{g}^a] \star E$, we have a hidden shift problem by defining $f(x) = [\mathfrak{g}^x] \star E_0$ and $g(x) = [\mathfrak{g}^x] \star E$, the permutations f, g over $\mathcal{E}$ are hidden shifted by a. By applying Kuperberg's algorithm, one can solve GAIP in time complexity $2^{O(\sqrt{\log(|G|)})}$.

We remark that there exist weak instances of the group for GAIP [FIM+14, BN18, CDEL21] conditioned on the structure of the group having a high rank cyclic subgroup. In general, this structure is unlikely to occur when the group is sampled from the imaginary ideal class group.

For efficient and versatile cryptosystems, we may require a few different assumptions. As the GAIP is an analogue of the classical discrete logarithm problem, so we have the standard CDH and DDH assumptions for group actions translated from the classical setting.

Problem 3 (Computational Diffie-Hellman (CDH) Problem)**.** *Let $(G, \mathcal{E}, \star, E_0)$ be an effective group action. The computational Diffie-Hellman problem is that given a tuple $(g_1 \star E_0, g_2 \star E_0)$ where g_1, g_2 are sampled uniformly from G, to compute $(g_1 g_2) \star E_0$.*

Notably, CDH is quantum equivalent to the GAIP problem [GPSV21]. A full quantum equivalence is given in [MZ22].

Problem 4 (Decisional Diffie-Hellman (DDH) Problem)**.** *Let $(G, \mathcal{E}, \star, E_0)$ be an effective group action. The decisional Diffie-Hellman problem is that the adversary $\mathcal{A}$ is given one instance of $T_b = (g_1 \star E_0, g_2 \star E_0, h_b \star E_0)$ where $h_0 = g_1 g_2, h_1 = g_3$ and $g_1, g_2, g_3, b \leftarrow G^3 \times \{0, 1\}$ to output b.*

We denote the advantage of the decisional problem adversary $\mathcal{A}$ by

$$\mathsf{Adv}^{\mathsf{DDH}}(\mathcal{A}) = |\Pr[\mathcal{A}(T_0) \to 1] - \Pr[\mathcal{A}(T_1) \to 1]| ,$$

where b is the randomness in the experiment, and the probability is taken over the randomness used by $\mathcal{A}$ and the randomness used in the experiment. The group action $(G, \mathcal{E}, \star, E_0)$ is implicitly parameterized in the experiment. We say the DDH problem is hard, if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl such that $\mathsf{Adv}^{\mathsf{DDH}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$.

Note that when using CSIDH as an instance, we require $p = 3 \pmod 4$ to avoid the attacks presented in [CSV20, CHVW22], exploiting distinct pairings. Both attacks rely on the existence of the nontrivial characters derived from the nontrivial 2-torsion subgroup in the ideal class group. Hence, when $p = 3 \pmod 4$, the group size is odd so the attacks are not applicable. Also, it is not clear when $p = 1 \pmod 4$ if the attack is still applicable to the case where 2-isogenies are not used in the experiment. Therefore, when CSIDH instantiated with $p = 3 \pmod 4$, DDH is believed to be hard.

Naturally, we can relax the standard DDH, which only gives one instance, to give multiple instances.

Problem 5 (Multi-Challenge Decisional Diffie-Hellman (mcDDH) Problem)**.** *Let $(G, \mathcal{E}, \star, E_0)$ be a group action and $b \in \{0, 1\}$. The multi-challenge decisional Diffie-Hellman experiment $\mathsf{Exp}^{\mathsf{mcDDH}}(b)$ on input b proceeds as follows. The adversary $\mathcal{A}$ is given $(g_1 \star E_0)$ where $g_1 \leftarrow G$ together with access to the oracle $\mathcal{O}_b^{\mathsf{mcDDH}}$ defined as follows:*

1. *$\mathcal{O}_0^{\mathsf{mcDDH}}$: $(g_2 \star E_0, (g_1 g_2) \star E_0)$ where g_2 are sampled uniformly from G,*
2. *$\mathcal{O}_1^{\mathsf{mcDDH}}$: $(g_2 \star E_0, g_3 \star E_0)$ where g_2, g_3 are sampled uniformly from G,*

to output $b \in \{0, 1\}$.

We denote the advantage of a multi-challenge decisional Diffie-Hellman problem adversary $\mathcal{A}$ problem by

$$\mathsf{Adv}^{\mathsf{mcDDH}}(\mathcal{A}) = \left|\Pr[\mathcal{A}(\mathsf{Exp}^{\mathsf{mcDDH}}(b = 0)) \to 1] - \Pr[\mathcal{A}(\mathsf{Exp}^{\mathsf{mcDDH}}(b = 1)) \to 1]\right| ,$$

where b is the randomness in the experiment, and the probability is taken over the randomness used by $\mathcal{A}$ and the randomness used in the experiment. The group action

$(G, \mathcal{E}, \star, E_0)$ is implicitly parameterized in the experiment. We say the mcDDH problem is hard, if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl such that $\mathsf{Adv}^{\mathsf{mcDDH}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$. One can use a standard hybrid argument and give a reduction from the DDH problem to the mcDDH problem.

A standard hybrid argument can lead to a reduction looseness that is proportional to the number of queries made. The equivalence is tight in the classical setting (i.e. the group setting) due to the randomizer introduced [Sta96] which can keep regenerating a DH instance or a random instance depending on the input instance. Achieving a tight equivalence for the decisional problem in the group action setting remains an open problem.

Looking ahead, our oblivious transfer (Chapter 4), the unforgeability of our accountable ring signature and group signature (Chapter 5), and one of our blind signature constructions are based on GAIP. The anonymity of the accountable ring signature and group signature and one of our verifiable random functions (Chapter 7) are based on DDH. Chapter 8 presents an identification scheme for the isogeny problem of a given smooth degree (2^k, to be more precise).

2.5 Discussion about Having Larger Known-Order Effective Group Actions

Despite the appealing cryptographic properties provided by the known-order effective group (Def. 2.3.4), one of the well-known technical bottlenecks in CSIDH-based action is that the largest known instantiation is CSIDH-512 [BKV19]. According to [CSCJR22], achieving NIST security level 1 and 2 require the underlying prime to be approximately $2^{2048}, 2^{6144}$ respectively.

In fact, the ideal class group of CSIDH-512 also holds the record for the largest known class number of a quadratic number field. A belief in the isogeny community is that finding such an instantiation of a known-order effective group action is theoretically feasible using a quantum computer, which is called *post-post-quantum cryptography* [DF19]. Here, we express a concern regarding this belief, based on an argument using the rule-of-thumb in lattice reduction [GN08] and the model of [Laa15].

First, we formalize the concept of employing quantum algorithms to instantiate a larger known-order effective group action based on a given CSIDH parameter $p = 4 \times p_1 \times \cdots \times p_d - 1$ as follows:

1. Use the quantum algorithm [Hal02, Hal05] to compute the class number N.

2. Use the quantum algorithm [Sho99] or [Kit95] to collect the relation lattice L modulo N of dimension d for the representatives $\mathfrak{l}_1, \cdots, \mathfrak{l}_d$.

3. Apply lattice reduction, potentially quantum-optimized, to reduce the lattice L with small coefficients.

4. Solve for the approximate closest vector problem (CVP) for a targeted vector $(e_1, \cdots, e_d)$ over the lattice L to evaluate the action $\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_d^{e_d}$.

It is worth noting that in the second step described above, Shor's algorithm (or its generalization [Kit95]) is sufficient to recover the relation vectors in the lattice. There is a unique representation for ideal classes in imaginary quadratic fields, and it can be efficiently computed using existing classical algorithms [Cox22, Chapter 7].

When the parameter is small, instantiation is possible. Indeed, [BKV19] presents an example for CSIDH-512, where the prime is $p \approx 2^{512}$ and the lattice dimension is 74. The size is so modest that the task of collecting very short vectors becomes attainable. Also, they deployed the approximate Voronoi cell to solve the approximate closest vector problem and find short coefficients in a very fast way. Overall, the group action evaluation only slows down by a factor of 1.15 compared to the original CSIDH. However, as the underlying prime grows larger, the method's complexity increases significantly. Specifically, the preprocessing time required to reduce the lattice basis or generating set, as outlined in Item 3 above, impacts the length of the vector $(e_1, \cdots, e_d)$ along with the time needed for evaluating group actions. From a theoretical standpoint, this length tends to grow exponentially with the dimension of the lattice [GN08, Laa15], as pointed out in the concurrent work [Pan23]. Nonetheless, it is crucial to emphasize that the practicality of obtaining KO-EGA using the method described above is not necessarily ruled out. Actual performance depends on specific implementation details, aligning with the spirit of lattice reduction.

Unfortunately, due to the lack of the corresponding ideal class group instances, we are not able to implement the lattice reduction at the current stage. Here, we provide an estimate of the factor slowdown that can be expected for CSIDH-6144 when employing the method described above to generate KO-EGA. Our approach involves utilizing the BKZ algorithm to find the reduced generating set for the lattice in Item 3.

To simplify our analysis, we introduce the following assumptions, acknowledging that these assumptions may not hold in all cases but are adopted for ease of analysis:

1. **Limited SVP Subroutine Executions:** Recall that the BKZ algorithm will iteratively invoke an SVP subroutine for sub-lattices of dimension β, called the block size, as the main approach to reduce the input lattice. We assume that the SVP subroutine within the BKZ algorithm is executed only $\lceil d/\beta \rceil$ times, where d represents the dimension of the lattice, and β denotes the block size.

2. **Sufficient Generating Set in One Execution:** We assume that a single execution of the BKZ algorithm in Item 3 yields a generating set that is sufficient for the purposes of running Item 4. Furthermore, we assume that the vectors within this resulting set have approximately the same length as the shortest vector in the resulting set.

3. **Expected Length of Output Vector in Approximate CVP Algorithm:** We assume that the expected length of the vector produced by the approximate CVP algorithm used in Item 4 is of average length of the vectors within the generating set.

We emphasize that these assumptions, while simplifying our analysis, should be recognized as *idealized scenarios for the user* and may not always reflect the complexity of practical implementations.

For an underlying prime of $p \approx 2^{6144}$, it takes at most 590 distinct odd prime factors for $p-1$ (i.e., $p = 4 \times p_1 \times \cdots \times p_{590} - 1$ and $d = 590$). (We remark that one can opt for larger prime factors in $p-1$ to reduce the dimension. However, this approach doesn not yield efficiency benefits. This is due to the fact that when the group size (approximately $\sqrt{p}$) is fixed, reducing the number of dimensions results in longer average vector lengths and slower evaluation times.) The coefficient interval for each generator $\mathfrak{l}_i$ is assumed to be of length 37 (e.g., $\{-18, \cdots, 18\}$), such that $37^{590} > 2^{3072}$. We may take the volume of the lattices to be $vol(L) \approx 2^{3072}$.

Given the block size β in the BKZ reduction, we use the root Hermite factor $\delta \approx \left(\frac{(\pi\beta)^{\frac{1}{\beta}}\beta}{2\pi e}\right)^{\frac{1}{2\beta-2}}$ to estimate the length of the shortest vector in the BKZ reduced basis by $\delta^d vol(L)^{1/d}$. We adopt the quantum-optimized cost model from [Laa15] to estimate the precomputation cost in the lattice reduction part. The estimation is presented in Tab. 2.1. Due to the ideal scenarios we assume for the user, the numbers can be viewed as lower bound estimations for the precomputation cost and the isogeny evaluation cost using the folklore method described above. The slowdown factor is given by calculating the expected length in ℓ_1-norm (by dividing $d/\sqrt{d}$) and divide it bythe expected ℓ_1-length of the vector in the REGA model, which is approximately $9.2 * 590$.

We also give the same estimation for CSIDH-2048 by taking $p \approx 2^{2048}$, $d \approx 235$, and each vector is taken from $\{-10, \cdots, 10\}$ in the REGA model such that $21^{235} > 2^{1024}$ in Tab. 2.2

We look forward to further investigations into the study of instantiating isogeny-based known-order effective group actions or of giving a more rigorous estimation and analysis.

Root Hermite Factor	Block Size	Preprocessing Cost	Length in ℓ_2-norm	Slowdown Factor (Estimated)
1.0091	100	$6 * 2^{29}$	8040	36.0
1.0080	130	$5 * 2^{37}$	4104	18.4
1.0076	140	$5 * 2^{40}$	3407	15.2
1.0074	160	$4 * 2^{46}$	2454	11.0

Table 2.1: An *estimation* cost of the lattice reduction part for finding the known-order effective group action of CSIDH-6144 using the folklore method describe above and an *estimated* slowdown factor of evaluation compared to REGA.

Root Hermite Factor	Block Size	Preprocessing Cost	Length in ℓ_2-norm	Slowdown Factor (Estimated)
1.0091	100	$3 * 2^{29}$	174.9	8.04
1.0080	130	$2 * 2^{37}$	133.8	5.71
1.0076	140	$2 * 2^{40}$	124.3	5.33
1.0074	160	$2 * 2^{46}$	109.1	5.01

Table 2.2: An *estimation* cost of the lattice reduction part for finding the known-order effective group action of CSIDH-2048 using the folklore method describe above and an *estimated* slowdown factor of evaluation compared to REGA.

Chapter 3

Cryptographic Preliminaries

3.1 Secret Key Encryption

A symmetric encryption scheme is a tuple of algorithms $(\mathsf{SKE.Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$, ciphertext space $\mathcal{C}$ and key space $\mathcal{K}$. We assume $|\mathcal{K}| \geq 2^\lambda$ to have large enough key space. We recall the standard IND-CPA security notion for a symmetric key encryption scheme.

Definition 3.1.1 (IND-CPA Security)**.** *A symmetric encryption scheme* $\Pi_{\mathsf{SKE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is* IND-CPA secure *if, for any* $\lambda \in \mathbb{N}$*, any PPT adversary* $\mathcal{A}$ *has at most a negligible advantage in the following game played against a challenger.*

(i) The challenger runs $pp \leftarrow \mathsf{SKE.Setup}(1^\lambda)$*,* $k \leftarrow \mathsf{KeyGen}(pp)$ *and samples a bit* $b \in \{0,1\}$*. The challenger provides* pp *to* $\mathcal{A}$*.*

(ii) $\mathcal{A}$ *sends a pair of messages* $(\mathsf{M}_0, \mathsf{M}_1) \in \mathcal{M}^2$ *to the challenger, and the challenger returns* $c_b \leftarrow \mathsf{Enc}_k(\mathsf{M}_b)$ *to* $\mathcal{A}$*.*

(iv) $\mathcal{A}$ *outputs a bit* $b^* \in \{0,1\}$*. We say* $\mathcal{A}$ *wins if* $b^* = b$*.*

The advantage of $\mathcal{A}$ *is defined as* $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{(\mathsf{Enc},\mathsf{Dec})}(\mathcal{A}) = |\Pr[\mathcal{A} \textit{ wins}] - 1/2|$*.*

In this work, we may assume KeyGen simply draws a key uniformly at random from $\mathcal{K}$ and abuse the notation $(\mathsf{Enc}, \mathsf{Dec})$ to represent a symmetric encryption scheme for simplicity.

3.2 Public Key Encryption

We recall the standard multi-challenge IND-CPA security of a public-key encryption (PKE) scheme.

Definition 3.2.1 (Public-Key Encryption)**.** *A public-key encryption* Π_{PKE} *over a message space* $\mathcal{M}$ *consists of four algorithms* $\Pi_{\mathsf{PKE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: *On input the security parameter* 1^λ*, it outputs a public parameter* pp*.*
- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$: *On input a public parameter* pp*, it outputs a public key and a secret key* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{M}) \to \mathsf{ct}$*: On input a public key* pk_i *and a message* $\mathsf{M} \in \mathcal{M}$*, it outputs a ciphertext* ct*.*
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{M}$ *or* $\perp$: *On input a secret key* sk *and a ciphertext* ct*, it outputs either* $\mathsf{M} \in \mathcal{M}$ *or a special symbol* $\perp \notin \mathcal{M}$*.*

We will denote by $\mathcal{R}$ *the set containing the randomness used by the encryption algorithm* Enc*.*

Below, we define the standard IND-CPA security extended to the multi-challenge setting. By using a textbook hybrid argument, it is clear that the multi-challenge definition can be reduced to the standard single-challenge definition with a tightness loss linear in the number of instances. The motivation for introducing the multi-challenge variant is because in some cases, we can show that the two definitions are equally difficult without incurring any reduction loss. Looking ahead, the notion will imply anonymity in our ring and group signature in Chapter 5.

Definition 3.2.2 (Multi-Challenge IND-CPA Security)**.** *A* PKE *scheme* $\Pi_{\mathsf{PKE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is* multi-challenge IND-CPA secure *against* Q *challenges if, for any* $\lambda \in \mathbb{N}$*, any PPT adversary* $\mathcal{A}$ *has at most a negligible advantage in the following game played against a challenger.*

(i) The challenger runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *and samples a bit* $b \in \{0, 1\}$*. The challenger provides* $(\mathsf{pp}, \mathsf{pk})$ *to* $\mathcal{A}$*.*

(ii) $\mathcal{A}$ *can adaptively query the challenge oracle at most* Q *times. In each query,* $\mathcal{A}$ *sends a pair of messages* $(\mathsf{M}_0, \mathsf{M}_1) \in \mathcal{M}^2$*, and the challenger returns* $\mathsf{ct}_b \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{M}_b)$ *to* $\mathcal{A}$*.*

(iv) $\mathcal{A}$ *outputs a bit* $b^* \in \{0, 1\}$*. We say* $\mathcal{A}$ *wins if* $b^* = b$*.*

The advantage of $\mathcal{A}$ *is defined as* $\mathsf{Adv}^{\mathsf{Multi\text{-}CPA}}_{\Pi_{\mathsf{PKE}},Q}(\mathcal{A}) = |\Pr[\mathcal{A} \text{ wins}] - 1/2|$*.*

3.3 Sigma Protocols

Definition 3.3.1 (Sigma Protocol). *A sigma protocol* Π_Σ *is a three-move proof system for a relation* R *that consists of oracle-calling PPT algorithms* $(P = (P_1, P_2), V = (V_1, V_2))$*, where* V_2 *is deterministic. We assume* P_1 *and* P_2 *share states and so do* V_1 *and* V_2*. Let* ChSet *denote the challenge space. Then,* Π_Σ *proceeds as follows.*

- *The prover, on input* $(\mathsf{st}, \mathsf{wt}) \in R$*, runs* $\mathsf{com} \leftarrow P_1^{\mathcal{O}}(\mathsf{st}, \mathsf{wt})$ *and sends a* commitment com *to the verifier.*
- *The verifier runs* $\mathsf{ch} \leftarrow V_1^{\mathcal{O}}(1^\lambda)$*, drawing a random challenge from* ChSet*, and sends it to the prover.*
- *The prover, given* ch*, runs* $\mathsf{resp} \leftarrow P_2^{\mathcal{O}}(\mathsf{st}, \mathsf{wt}, \mathsf{ch})$ *and returns a* response resp *to the verifier.*
- *The verifier runs* $V_2^{\mathcal{O}}(\mathsf{st}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$ *and outputs* $\top$ *(accept) or* $\bot$ *(reject).*

Here, $\mathcal{O}$ *is modeled as a random oracle. For simplicity, we often drop* $\mathcal{O}$ *from the superscript when it is clear from context. We assume the statement* st *is always given as input to both the prover and the verifier. The protocol* transcript $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ *is said to be valid in case* $V_2(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ *outputs* $\top$*.*

Definition 3.3.2 (Correctness). *A sigma protocol* Π_Σ *is said to be correct if for all* $\lambda \in \mathbb{N}$*,* $(\mathsf{st}, \mathsf{wt}) \in R$ *and the prover and the verifier both follow the protocol specification, the verifier always outputs* $\top$*.*

Definition 3.3.3 (High Min-Entropy). *We say a sigma protocol* Π_Σ *has* $\alpha(\lambda)$ *min-entropy if for any* $\lambda \in \mathbb{N}$*,* $(\mathsf{st}, \mathsf{wt}) \in R$*, and a possibly computationally-unbounded adversary* $\mathcal{A}$*, we have*

$$\Pr\left[\mathsf{com} = \mathsf{com}' \middle| \mathsf{com} \leftarrow P_1^{\mathcal{O}}(\mathsf{st}, \mathsf{wt}), \mathsf{com}' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{st}, \mathsf{wt})\right] \leq 2^{-\alpha},$$

where the probability is taken over the randomness used by P_1 *and by the random oracle. We say* Π_Σ *has* high min-entropy *if* $2^{-\alpha}$ *is negligible in* λ*.*

Definition 3.3.4 (Honest Verifier Zero-Knowledge (HVZK)). *We say* Π_Σ *is* honest-verifier-zero-knowledge *for relation* R *if there exists a PPT simulator* $\mathsf{Sim}^{\mathcal{O}}$ *with access to a random oracle* $\mathcal{O}$ *such that any statement-witness pair* $(\mathsf{st}, \mathsf{wt}) \in R$*,* $\mathsf{ch} \in \mathsf{ChSet}$*,* $\lambda \in \mathbb{N}$ *and any computationally-unbounded adversary* $\mathcal{A}$ *that makes at most a polynomial number of queries to* $\mathcal{O}$*, we have*

$$\mathsf{Adv}_{\Pi_\Sigma}^{\mathsf{HVZK}}(\mathcal{A}) := \left|\Pr[\mathcal{A}^{\mathcal{O}}(P^{\mathcal{O}}(\mathsf{st}, \mathsf{wt}, \mathsf{ch})) = 1] - \Pr[\mathcal{A}^{\mathcal{O}}(\mathsf{Sim}^{\mathcal{O}}(\mathsf{st}, \mathsf{ch})) = 1]\right| = \mathsf{negl}(\lambda),$$

where $P = (P_1, P_2)$ is a prover running on $(\mathsf{st}, \mathsf{wt})$ *with a challenge fixed to* ch *and the probability is taken over the randomness used by* (P, V) *and by the random oracle.*

Remark 3.3.5. *Roughly speaking, HVZK requires that there exists a PPT simulator* Sim *such that given any statement* st *(in the language) and challenge* $\mathsf{ch} \in \mathsf{ChSet}$*, it outputs a valid transcript* $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ *that is indistinguishable from a real transcript.* Witness indistinguishability *is a weaker notion compared with HVZK, where we require the interactions between a prover using a witness* wt_1 *or* wt_2 *satisfying* $(\mathsf{st}, \mathsf{wt}_1), (\mathsf{st}, \mathsf{wt}_2) \in R$ *are indistinguishable. Namely, the interaction does not leak which witness is being used. We will use this property in the blind signature construction Chapter 6.*

Definition 3.3.6 (Special Soundness)**.** *We say a sigma protocol* Π_Σ *has special soundness if there exists a polynomial-time extraction algorithm* $\mathsf{Extract}$ *such that, given a statement* st *and any two valid transcripts* $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ *and* $(\mathsf{com}, \mathsf{ch}', \mathsf{resp}')$ *relative to* st *and such that* $\mathsf{ch} \neq \mathsf{ch}'$*, outputs a witness* wt *satisfying* $(\mathsf{st}, \mathsf{wt}) \in R$*.*

In some circumstances, we can relax the relation for special soundness to R' where $R \subseteq R'$. That is, we allow the extractor outputs wt such that $(\mathsf{st}, \mathsf{wt}) \in R'$ but $(\mathsf{st}, \mathsf{wt}) \notin R$. As long as given st to find wt such that $(\mathsf{st}, \mathsf{wt}) \in R'$, the sigma protocol can still serve as a proof system for some applications.

3.4 Proof Systems

We consider non-interactive zero-knowledge proof of knowledge protocols (or simply NIZK (proof system)) in the ROM. Below, we define a variant where the proof is generated with respect to a label. Although syntactically different, such NIZK is analogous to the notion of signature of knowledge [CL06].

Definition 3.4.1 (NIZK Proof System)**.** *Let* L *denote a label space, where checking membership can be done efficiently. A non-interactive zero-knowledge (*NIZK*) proof system* Π_{NIZK} *for the relations* R *and* $\tilde{R}$ *such that* $R \subseteq \tilde{R}$ *(which are implicitly parameterized by* λ*) consists of oracle-calling PPT algorithms* $(\mathsf{Prove}, \mathsf{Verify})$ *defined as follows:*

$\mathsf{Prove}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \mathsf{W}) \to \pi/\bot$: *On input a label* $\mathsf{lbl} \in \mathsf{L}$*, a statement and witness pair* $(\mathsf{X}, \mathsf{W}) \in R$*, it outputs a proof* π *or a special symbol* $\bot$ *denoting abort.*

$\mathsf{Verify}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \pi) \to \top/\bot$: *On input a label* $\mathsf{lbl} \in \mathsf{L}$*, a statement* X*, and a proof* π*, it outputs either* $\top$ *(accept) or* $\bot$ *(reject).*

Definition 3.4.2 (Correctness)**.** *A* NIZK *proof system* Π_{NIZK} *is correct if for all* $\lambda \in \mathbb{N}$*,* $\mathsf{lbl} \in \mathsf{L}$*,* $(\mathsf{X}, \mathsf{W}) \in R$*, we have*

$$\Pr\left[\ \mathsf{Verify}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \pi) = \top \ \middle|\ \begin{array}{c} \pi \leftarrow \mathsf{Prove}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \mathsf{W}), \\ \pi \neq \bot. \end{array}\ \right] = 1,$$

where the probability is taken over the randomness used by $(\mathsf{Prove}, \mathsf{Verify})$ *and by the random oracle.*

Definition 3.4.3 (Zero-Knowledge)**.** *Let* $\mathcal{O}$ *be a random oracle,* Π_{NIZK} *a* NIZK *proof system, and* $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ *a zero-knowledge simulator for* Π_{NIZK}*, consisting of two algorithms* Sim_0 *and* Sim_1 *with a shared state. We say the advantage of an adversary* $\mathcal{A}$ *against* Sim *is*

$$\mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{A}) = \left|\Pr\left[\mathcal{A}^{\mathcal{O}, \mathsf{Prove}}(1^\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{Sim}_0, \mathcal{S}}(1^\lambda) = 1\right]\right|,$$

where Prove *and* $\mathcal{S}$ *are prover oracles that on input* $(\mathsf{lbl}, \mathsf{X}, \mathsf{W})$ *return* $\bot$ *if* $\mathsf{lbl} \notin \mathsf{L} \vee (\mathsf{X}, \mathsf{W}) \notin R$ *and otherwise return* $\mathsf{Prove}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \mathsf{W})$ *or* $\mathsf{Sim}_1(\mathsf{lbl}, \mathsf{X})$*, respectively. Moreover, the probability is taken also over the randomness of sampling* $\mathcal{O}$*.*

We say Π_{NIZK} *for* R *and* $\tilde{R}$ *is zero-knowledge if there exists a PPT simulator* Sim *such that for any (possibly computationally-unbounded) adversary* $\mathcal{A}$ *making at most polynomially many queries to the random oracle and the prover oracle, we have* $\mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$*.*

Statistical soundness, the most widely-used notion for a proof system, guarantees that any adversary cannot generate a proof for an invalid statement except with a negligible probability.

Definition 3.4.4 (Statistical Soundness)**.** *Let* $\mathcal{O}$ *be a random oracle and* Π_{NIZK} *a* NIZK *proof system. We say the advantage of an adversary* $\mathcal{A}$ *against soundness is*

$$\mathsf{Adv}^{\mathsf{soundness}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{A}) = \Pr\left[\begin{array}{c} \nexists \mathsf{W} : (\mathsf{X}, \mathsf{W}) \in \tilde{R} \wedge \\ \mathsf{Verify}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \pi) = \top \end{array} \ \middle|\ (\mathsf{lbl}, \mathsf{X}, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda)) \right],$$

where the probability is taken also over the randomness of sampling $\mathcal{O}$*.*

We say the NIZK *proof system* Π_{NIZK} *for* R *and* $\tilde{R}$ *has (relaxed) statistical soundness if for any (possibly computationally-unbounded) adversary* $\mathcal{A}$ *making at most polynomially many queries to the random oracle, we have* $\mathsf{Adv}^{\mathsf{soundness}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$*.*

We introduce two stronger notions, multi-proof online extractability and online extractability, which will be a useful tool in a security proof. Roughly speaking, online extractability requires the existence of an extraction algorithm which, on input a valid proof π and the list or random-oracle queries made by an adversary, always output a (relaxed) witness except with a negligible probability. It is worth noting that the extraction process does not involve rewinding the adversary.

Conceptually, extractability (not online) is sufficient for certain applications, such as digital signature schemes. Usually, in the chosen-message unforgeability experiment of a digital signature scheme, a reduction outputs the witness by rewinding the adversary *one time* on distinct hash values to respond to the hash query. Employing the forking lemma, one can argue that the extraction is successful with a non-negligible probability.

However, rewinding arguments typically result in a loose reduction loss (quadratic) and complex proof arguments in some scenarios. For instance, certain blind signature schemes exhibit subtle issues when employing the rewinding method, issues that have been overlooked by numerous follow-up works and were only identified and rectified after 20 years [AO00, KLX22a]. Thus, we strive to avoid using an extractor that relies on rewinding. More specifically, we will utilize extractors in Chapter 7 twice in a single proof and in Chapter 5 an arbitrary number polynomial in λ in a proof. An extractor that employs rewinding would incur either a convoluted proof or exponential loss in the reduction. An online extractor offers a solution to these issues.

Definition 3.4.5 (Multi-Proof Online Extractability (mpOE))**.** *A* NIZK *proof system* Π_{NIZK} *is (multi-proof) online extractable if there exists a PPT extractor* $\mathsf{OnlineExtract}$ *such that for any (possibly computationally-unbounded) adversary* $\mathcal{A}$ *making at most polynomially-many queries has at most a negligible advantage in the following game played against a challenger (with access to a random oracle* $\mathcal{O}$*).*

(i) The challenger prepares empty lists $L_{\mathcal{O}}$ *and* L_P*, and sets* flag *to* 0*.*

(ii) $\mathcal{A}$ *can make random-oracle, prove, and extract queries an arbitrary polynomial number of times:*

- $(\mathtt{hash}, x)$*: The challenger updates* $L_{\mathcal{O}} \leftarrow L_{\mathcal{O}} \cup \{(x, \mathcal{O}(x))\}$ *and returns* $\mathcal{O}(x)$*. We assume below that* $\mathcal{A}$ *runs the verification algorithm after receiving a proof from the prover oracle and before submitting a proof to the extract oracle.*[1]
- $(\mathtt{prove}, \mathsf{lbl}, \mathsf{X}, \mathsf{W})$*: The challenger returns* $\perp$ *if* $\mathsf{lbl} \notin \mathsf{L}$ *or* $(\mathsf{X}, \mathsf{W}) \notin R$*. Otherwise, it returns* $\pi \leftarrow \mathsf{Prove}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \mathsf{W})$ *and updates* $L_P \leftarrow L_P \cup \{\mathsf{lbl}, \mathsf{X}, \pi\}$*.*
- $(\mathtt{extract}, \mathsf{lbl}, \mathsf{X}, \pi)$*: The challenger checks if* $\mathsf{Verify}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \pi) = \top$ *and* $(\mathsf{lbl}, \mathsf{X}, \pi) \notin L_P$*, and returns* $\perp$ *if not. Otherwise, it runs* $\mathsf{W} \leftarrow \mathsf{OnlineExtract}^{\mathcal{O}}(\mathsf{lbl}, \mathsf{X}, \pi, L_{\mathcal{O}})$ *and checks if* $(\mathsf{X}, \mathsf{W}) \notin \tilde{R}$*, and returns* $\perp$ *if yes and sets* $\mathsf{flag} = 1$*. Otherwise, if all checks pass, it returns* W*.*

[1]This is w.l.o.g., and guarantees that the list $L_{\mathcal{O}}$ is updated with the input/output required to verify the proof $\mathcal{A}$ receives or sends.

(iii) At some point $\mathcal{A}$ outputs 1 to indicate that it is finished with the game. We say $\mathcal{A}$ wins if $\mathsf{flag} = 1$. *The advantage of $\mathcal{A}$ is defined as* $\mathsf{Adv}^{\mathsf{mpOE}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{A}) = \Pr[\mathcal{A}\ \textit{wins}]$ *where the probability is also taken over the randomness used by the random oracle.*

We introduce the stronger notion *multi-proof* online-extractability mainly for our ring/group signatures constructions in Chapter 5. Note, importantly, that the mpOE experiment is not given access to the queries $\mathsf{Prove}^{\mathcal{O}}$ makes directly to $\mathcal{O}$. Thus, mpOE is not guaranteed to return a valid witness W when called with any output of the Prove oracle. The requirement that $(\mathsf{lbl}, \mathsf{X}, \pi) \notin L_P$ ensures that this does not allow the adversary to trivially win the game, and in particular by extension ensures that modifying the label lbl should invalidate any proof obtained from the Prove oracle.

The mpOE notion provides a strong guarantee that an adversary cannot break the extractability of the proof, even with access to a proving oracle. However, in certain circumstances, online-extractability (OE) alone is sufficient. This notion is somewhat weaker since it removes the proving oracle but relies only on the extractability of the random oracle, rather than the programmability used in mpOE.

Definition 3.4.6 (Online Extractability (OE)). *Let Π_{NIZK} be a NIZK proof system. We say Π_{NIZK} has online-extractability if for any (possibly computationally-unbounded) adversary $\mathcal{A}$, there exists a PPT extractor* $\mathsf{OnlineExtract}$ *with only extractability access to $\mathcal{O}$ such that $\mathcal{A}$ wins the following game with a negligible advantage:*

(i) $\mathcal{A}$ can make polynomial number of queries of the random oracle.

(ii) $\mathcal{A}$ outputs st *and π.*

We say $\mathcal{A}$ wins if $\mathsf{Verify}^{\mathcal{O}}(\mathsf{st}, \pi) = \top$ *and* $(\mathsf{st}, \mathsf{wt}) \notin \widetilde{R}$ *where* $\mathsf{wt} \leftarrow \mathsf{OnlineExtract}(\mathsf{st}, \pi)$. *The advantage of $\mathcal{A}$ is defined as* $\mathsf{Adv}^{\mathsf{OE}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{A}) = \Pr[\mathcal{A}\ \textit{wins}]$ *where the probability is taken over the randomness used by the random oracle.*

Remark 3.4.7 (OE, mpOE implies statistical soundness.). *If a* NIZK *proof system Π_{NIZK} is (multi-proof) online extractable, it is statistically sound—that is, online extractability implies statistical soundness. This is clear, because if an adversary is able to generate an accepting tuple* $(\mathsf{lbl}, \mathsf{X}, \pi)$ *for which* $\nexists \mathsf{W} : (\mathsf{X}, \mathsf{W}) \in \tilde{R}$ *in the soundness game, then clearly* $(\mathtt{extract}, \mathsf{lbl}, \mathsf{X}, \pi)$ *will allow the adversary to win the online extractability game.*

Remark 3.4.8 (NIZKs with Labels). *If the label space of the* NIZK *is* $\mathsf{L} = \{\perp\}$*, we say the* NIZK *is without labels (or a plain/unlabelled* NIZK*). In this case, we omit the* lbl *argument from the* Prove *and* Verify *functions for clarity.*

Chapter 4

UC-Secure Oblivious Transfers

This chapter presents the work carried out in [LGD21], which the author of the thesis co-authored. The author contributed to the work by proposing the project and contributing the most ideas (designing, hardness reductions, security proofs etc). The chapter is almost verbatim of the original work. The work were mostly done in 2020. We give a brief overview of the recent advancements in Sec. 4.6.

Abstract. In this chapter, we present the first efficient (using only a constant number isogeny compuation) UC-secure OT from isogenies in record. This scheme builds on the group action inverse problem and leverages quadratic twists in a clever way to achieve efficiency. To demonstrate the security of our protocol, we introduce the computational reciprocal CSDIH problem and establish its equivalence to the group action inverse problem (GAIP).

4.1 Introduction

Oblivious transfer (OT) was first introduced by Rabin [Rab81] in 1981 to establish an exchange of secrets protocol based on the factoring problem. Say the sender has two messages, oblivious transfer allows the receiver to know one of them and keeps the sender oblivious to which message has been received. Meanwhile, the receiver learns no information about the unchosen message.

It has been shown that oblivious transfer can serve as an important and powerful cryptographic building block. Oblivious transfer can be used as one of the components to realize any cryptographic functionality [GMW87, CvdGT95, Ode09]. Several oblivious transfer protocols based on Diffie-Hellman-related problems were proposed [BM89, NP01, PVW08, CO15, BDD$^+$17].

Even though oblivious transfer protocols exist for various hardness assumptions, a cryptographic protocol subordinate to either the discrete logarithm problem or the factoring problem will suffer a polynomial-time quantum attack from Shor's algorithm [Sho99]. Given the fact, several post-quantum OTs have been proposed, including from lattices [PVW08] and from codes [DvdGMN08, DNMQ12, BDD$^+$17]. Notably, a few isogeny-based OTs have been proposed [DOPS20, Vit19], some of which are based on SIDH is no longer secure.

There are various notions of security for OT. Traditional notions guarantee privacy for both parties, including one-sided simulation or the view-based definition for a two-message protocol [NP01, DvdGMN08, HL10]. These notions ensure privacy for both parties in a standalone setting. However, in most cases, these notions guarantee nothing in real-world deployment where OT is executed, as a subroutine, concurrently with others in an enormous and complex construction. To ensure the entire system's security, a powerful notion – universally-composable security (UC security) – introduced by Canetti [Can01] is precisely the one capturing the feature. The notion ensures the security of any system composed arbitrarily of UC-secure components.

Essentially, the adversaries in UC security come with two flavors: semi-honest or malicious ones. The former will follow the protocol specification while the latter does not. To thwart the real-world threats, a cryptosystem being UC secure against malicious adversaries is undoubtedly the best guarantee.

Given the prior works in the isogeny literature, at the time this research was done in 2020, a UC-secure OT against malicious adversaries appears to be elusive. In fact, a few schemes [DOPS20, Vit19] achieve UC-security against an semi-honest adversary. Using folk wisdom in the MPC literature, these schemes can be upgraded to be secure against the malicious ones via applying zero-knowledge proofs. Though subjects vary, the main idea remains the same – restrict the behaviour of the corrupted parties using zero-knowledge proofs. For instance, one can apply the ZKP to the random tape, or to the crucial message to be transmitted [GMW87, Pas03]. By using the trapdoor of the proof system, a simulator extracts the secret input from the malicious adversary. Aside from the ZKP technique, some transformations can also help for specific functionalities (e.g. [DGH$^+$20] for OTs). Using any of these techniques (together with ZKP for the isogeny languages [FG19, BKV19]), we obtain UC-secure isogeny-based OT against malicious adversaries immediately. However, all result in one consequence: they are *inefficient*. The efficiency is bounded below by λ times isogeny evaluations for the security parameter λ.

It is natural to ask:

Can we have an efficient oblivious OT from isogenies UC-secure against malicious adversaries?

4.2 Preliminaries

Notation.

We use multiplication notation for an effective group action (EGA) $(G, \mathcal{E}, E_0, \star)$.

4.2.1 Assumptions

The section starts with a new introduced assumption for our UC-secure construction – the reciprocal CSIDH (rCDH) problem. To show the hardness of the problem, we start from the square and the inverse CDH problem. We will show classical reductions to prove the equivalence of the square CDH problem, the inverse CDH problem and the reciprocal CSIDH problem. This gives a strong guarantee for the hardness of the reciprocal problem because both the square and inverse problems are known to be as hard as GAIP (Dlog).

Problem 6 (**Compuational Square CSIDH Problem**, sCDH)**.** *Given an EGA $(G, \mathcal{E}, E_0, \star)$ and curves E, $s \star E$ in $\mathcal{E}$ where $s \in G$, to find $E' \in \mathcal{E}$ such that $E' = s^2 \star E$.*

Problem 7 (**Computational Inverse CSIDH Problem**, iCDH)**.** *Given an EGA $(G, \mathcal{E}, E_0, \star)$ and curves E, $s \star E$ in $\mathcal{E}$ where $s \in G$, to find $E' \in \mathcal{E}$ such that $E' = s^{-1} \star E$.*

The advantage of an adversary $\mathcal{A}$ against the inverse CSIDH problem is defined as $\mathsf{Adv}^{\mathsf{iCDH}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$. The equivalence between these two problems is not hard to see. Let $(E, s \star E)$ be an "undetermined" instance. With the instance $(s \star E, E)$ the answer to the inverse problem is exactly $s^2 \star E$. Similarly, with the instance $(s \star E, E)$ the answer to the square problem is exactly $s^{-1} \star E$. This proves the equivalence of sCDH and iCDH.

For each of the two problems, a conditional reduction of CDH was given in [Fel19]. The condition for the second reduction is that the group order is given and odd. Therefore, we can say that there is a quantum reduction [Sho99, Hal02] to the computational CSIDH problem when $p = 3 \mod 4$. In fact, there is also an efficient quantum reduction for the case of $p = 1 \mod 4$, see Appendix A.1. Note that the quantum computation is only to compute the group structure of G, and so can be considered as a precomputation; the remainder of the reduction is classical.

Also, it has been shown in [GPSV21, MZ22] that CDH is quantum equivalent to GAIP. In fact, we can have a reduction to prove the equivalence to GAIP where the reduction is as tight as that of CDH. The proof is based on the strategy of [GPSV21] and skips the proof of [Fel19].

Proposition 4.2.1. *Let $(G, \mathcal{E}, E_0, \star)$ be an EGA where G is of order N and cyclic and generated by a public $g \in G$. Given (perfect) access (correct on all inputs) to the* sCDH *oracle, there exists a quantum algorithm to solve* GAIP *problem in polynomial time. The result remains the same if it is given access to the* iCDH *oracle.*

Proof. Given the instance E, to find $s \in G$ such that $s \star E_0 = E$. Say $g^k = s$. To find $s \in G$ such that $s \star E_0 = E$ for a given instance E, we can use the following approach. Firstly, we can compute $s^n \star E_0$ for any $n \in \mathbb{N}$ using the oracle. We can achieve this by making two types of queries to the oracle: for input $(E, s^i \star E)$, the oracle returns $s^{2i} \star E$, and for input $(s^{-1} \star E, s^i \star E)$, the oracle returns $s^{2i+1} \star E$. Using this, we can compute $s^n \star E_0$ with only $\log(n) + 1$ oracle queries.

Next, consider the function $f : \mathbb{Z}_N \times \mathbb{Z}_N \to \mathcal{E}$ defined as $(m, n) \mapsto (g^m s^n) \star E$. The function f hides the abelian group $\langle(-k, 1)\rangle$, and therefore we can apply the quantum period-finding algorithm to f to find a generator (m', n') for $\langle(-k, 1)\rangle$. Since (m', n') is a generator for $\langle(-k, 1)\rangle$, we have n' is invertible modulo N. Then, we know that $g^{-m'/n'} = s$.

The result holds true when the access is to the iCDH oracle because, as described above, the oracle outputs the same as the sCDH oracle by reversing the order of the input entries. □

As Castryck et al. [CLM+18] pointed out both problems contain exceptional cases when E_0 takes part in the problems due to the symmetric structure. That is, $(a \star E_0)^t = a^{-1} \star E_0$, and so Problem 7 is easy in the special case $E = E_0$. The issue can be circumvented if the public curve is generated by a trusted third party.

Problem 8 (**Reciprocal CSIDH Problem**, rCDH)**.** *Let $(G, \mathcal{E}, E_0, \star)$ be an EGA and E in $\mathcal{E}$ be given by the challenger. Firstly, the adversary chooses and commits to $X \in \mathcal{E}$, then receives the challenge $s \star E$ where $s \leftarrow G$ from the challenger. The adversary wins if it outputs the pair $(s \star X, s^{-1} \star X)$ with respect to the committed X.*

The advantage of $\mathcal{A}$ against the reciprocal CSIDH problem is defined as $\mathsf{Adv}^{\mathsf{rCDH}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$. Let $\mathsf{Adv}^{\mathsf{rCDH}}(\mathcal{A}(E; X))$ denote $\mathcal{A}$ wins conditioned on committed X.

Intuitively, rCDH is a hybrid and relaxed version of the square problem or the inverse problem (Probs. 6 and 7). To see this, if one can solve the inverse problem, then, by taking $X = E$, we have $(s \star X, s^{-1} \star X) = (s \star E, s^{-1} \star E)$. Then, one can solve rCDH by invoking the iCDH oracle. Conversely, if an attacker knows the isogeny between X and E, or E^t, then this can be used to solve iCDH. That is, say $X = r \star E$, one can obtain $s^{-1} \star E$ by computing $r^{-1} \star (s^{-1} \star X)$ with the given r. On the other hand, if $X = r \star E^t$, one can obtain $s^{-1} \star E$ by computing $r \star (s \star X)^t$ with the given r. Note that the attacker is not required to know the isogeny between X and E or E^t in the problem.

The reciprocal CSIDH problem appears to be non-standard at first sight but, in fact, it is equivalent to the inverse CSIDH problem.

Proposition 4.2.2. *The reciprocal CSIDH problem is equivalent to the computational inverse CSIDH problem.*

Proof. Given a rCDH adversary $\mathcal{A}$ and an iCDH challenge (E, E_1), we can construct an iCDH algorithm $\mathcal{B}$ as follows.

1. Invoke $\mathcal{A}$ with E.
2. Receive $X \in \mathcal{E}$ from $\mathcal{A}$. Send the challenge $t_1 \star E_1$ to the adversary where $t_1 \xleftarrow{\$} G$.
3. Receive (X_0, X_1) from $\mathcal{A}$. Rewind $\mathcal{A}$ to the time when it outputs X.
4. Send $t_2(t_1)^{-1} \star X_0$ as the new challenge with respect to committed X where $t_2 \xleftarrow{\$} G$.
5. After receiving (X_0', X_1') from the adversary, $\mathcal{B}$ outputs $t_2 \star X_1'$.

By the heavy row lemma ([PS00], Lemma 1), with 1/2 chance with respect to $X \leftarrow \mathcal{A}$, we have

$$\mathsf{Adv}^{\mathsf{rCDH}}(\mathcal{A}) \leq 2\mathsf{Adv}^{\mathsf{rCDH}}(\mathcal{A}(E; X)).$$

Say $\mathcal{A}$ answers correctly twice. Claim $t_2 \star X_1' = s^{-1} \star E$. Write $E_1 = s \star E$ and $X = b \star E$ for some $b \in G$. We have $X_0 = st_1 \star X = sbt_1 \star E$. Thereby, $t_2(t_1)^{-1} \star X_0 = sbt_2 \star E$. The correctness of the second response from $\mathcal{A}$ implies $t_2 \star X_1' = t_2 \star (sbt_2^{-1} \star X) = s^{-1} \star E$. Hence, the result follows.

Precisely, we have

$$(\mathsf{Adv}^{\mathsf{rCDH}}(\mathcal{A}))^2 \leq 8\mathsf{Adv}^{\mathsf{iCDH}}(\mathcal{B}).$$

$\square$

In the proof above, the reduction $\mathcal{B}$ firstly extracts the first curve from the first response, and rewinds the adversary with a new challenge with respect to the extracted curve. Then, $\mathcal{B}$ extracts the second curve from the second response, which will be the solution for the inverse CSIDH problem.

Looking ahead, Prob. 8 depicts a core idea of our protocol. If the receiver commits to a curve X, then $s \star X$ and $s^{-1} \star X$ are two decryption keys, and the receiver can get only one decryption key unless the receiver can solve a hard problem. Furthermore, after committing to X, the receiver can only get the i-th decryption for some $i \in \{0, 1\}$. This captures the main idea of our "proof of ability to decrypt mechanism", which allows us to extract corrupted receiver's input by reading random oracle's queries.

Problem 9 (**Tweaked Reciprocal CSIDH problem**, tReGA)**.** *Given E in $\mathcal{E}$. The adversary $\mathcal{A}$ chooses and commits to a curve $X \in \mathcal{E}$.*

(i) $\mathcal{A}$ receives the first challenge $s \star E$ where $s \leftarrow G$ from the challenger. $\mathcal{A}$ outputs a $C \in \mathcal{E}$.

(ii) The challenger sends s and another challenge $s' \star E$ to $\mathcal{A}$ where $s' \leftarrow G$.

(iii) $\mathcal{A}$ outputs $C' \in \mathcal{E}$.

Write $(C_0, C_1) = (s \star X, s^{-1} \star X)$ and $(C_0', C_1') = (s' \star X, s'^{-1} \star X)$. We say $\mathcal{A}$ wins if $(C, C') = (C_i, C_{1-i}')$ for some $i \in \{0, 1\}$.

The advantage of an adversary $\mathcal{A}$ against the tweaked reciprocal CSIDH problem is defined as $\mathsf{Adv}^{\mathsf{tReGA}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$. Let $\mathsf{Adv}^{\mathsf{tReGA}}(\mathcal{A}(E; X))$ denote the probability that $\mathcal{A}$ wins when the parameter is E and $\mathcal{A}$ returns X. By using the same approach above, one can show the tweaked reciprocal CSIDH problem is as hard as the inverse CSIDH problem.

Proposition 4.2.3. *The tweaked reciprocal CSIDH problem is equivalent to the computational inverse CSIDH problem.*

Due to the similarity, we leave the proof in Appendix A.2.

We end the subsection with the following problem relations. (A full reduction is provided in Appendix A.1.)

$$\begin{aligned}\textbf{GAIP} &=_{quantum} \textbf{Computational Inverse CDH}\\ &=_{classical} \textbf{Computational Square CDH}\\ &=_{classical} \textbf{Computational Reciprocal CDH}\\ &=_{classical} \textbf{Tweaked Reciprocal CDH}\end{aligned}$$

4.2.2 Functionalities

In this subsection, we define the functionalities we need as well as the related security definitions. We refer to [HL10] for more detailed explanations and intuitions.

$\mathcal{F}_{\mathsf{RO}}$-Functionality of Random Oracle

The functionality is a function with the domain $\mathcal{D}$ and the codomain $\mathcal{R}$. It keeps a list L of pairs in $\mathcal{D} \times \mathcal{R}$ where the initial state is empty. It works as follows:

1. Upon receiving a query $C \in \mathcal{D}$, check whether $(C, k') \in L$ for some $k' \in \mathcal{R}$. If so, set $k = k'$; if not, generate $k \leftarrow \mathcal{R}$ and store the pair (C, k) in the list L.

2. Output k.

The functionality of a random oracle $\mathcal{F}_{\mathsf{RO}}$ internally contains an initially empty list. Upon receiving the query from the domain, it will check whether it is a repetition. If so, return the value assigned before; otherwise, it randomly assigns a value from the codomain, stores the pair, and returns the value. Formally speaking, an input of a random oracle can be an arbitrary binary string. For simplicity, we restrict the domain to $\mathcal{E}$. This can be easily and compatibly extend to $\{0,1\}^*$, since supersingularity can be efficiently verified [CLM+18].

$\mathcal{F}_{\mathsf{TSC}}$-Functionality of a trusted setup curve

The functionality is to output an element of $\mathcal{E}$. It generates an ideal class $t \leftarrow G$ and outputs the curve $t * E_0$.

The functionality of trusted setup curves $\mathcal{F}_{TSC}$ serves as a setup for generating a curve for the protocol. This setup hides the relation t between the public curve and the curve E_0. In practice, this can be replaced with a key exchange protocol [BD21]. That is, two parties do a key exchange first and obtain a curve such that the isogeny relation to E_0 remains unknown if the two parties do not share their ideal classes or collude.

Here we define the functionality of oblivious transfer in a simple and classic way. The two-party functionality of the oblivious transfer is characterized by $\mathcal{F}_{\mathsf{OT}} = (f_1, f_2)$ where $f_1 : \{0,1\}^* \times \{0,1\}^* \to \{\perp\}$ and $f_2 : \{0,1\} \to \{0,1\}^*$. Looking ahead, f_1, f_2 are the algorithms executed by the sender and the receiver respectively. The functionality $\mathcal{F}_{\mathsf{OT}} : \{0,1\}^* \times \{0,1\}^* \times \{0,1\} \to \{\perp\} \times \{0,1\}^*$ takes in a message pair $x = (\mathsf{M}_0, \mathsf{M}_1)$ of equal length from one party and a bit $y = \mathsf{i} \in \{0,1\}$ from the other party, and returns $\mathcal{F}_{\mathsf{OT}}(x, y) = (f_1(x, y), f_2(x, y)) = (\perp, \mathsf{M}_i)$ where $\perp$ represents an empty string.

We briefly define the security of OT. We refer [HL10, Lin17] for more details. Intuitively, we say a protocol realizes the functionality securely in the simulation-based definition, if the protocol realizes the function and also whatever the adversary can learn from a real execution of the protocol can be indistinguishably generated by a simulator. Thus, we have to formalize the “view" of a corrupted party and compare the output of the protocol with the ideal functionality. Let π be a protocol computing $\mathcal{F}_{\mathsf{OT}}$. We denote by $view_i^\pi(x, y)$ the transcript that records whatever the i^{th} party sees during an execution of the protocol π taking input (x, y). Precisely, $view_i^\pi(x, y)$ is the tuple $(input, r^i, m_1^i, ..., m_n^i)$ where $input$ is the input of the party, r^i is its internal random tape, and m_j^i is the j^{th} received message. We also write $output_i^\pi(x, y)$ as the output received by the i^{th} party after the execution of the protocol π with the input (x, y), and write $output^\pi(x, y) = (output_1^\pi(x, y), output_2^\pi(x, y))$. In particular, if the protocol π completely realizes the functionality $\mathcal{F}_{\mathsf{OT}}$, then $output^\pi(x, y) = \mathcal{F}_{\mathsf{OT}}(x, y)$.

Definition 4.2.4. *(OT security against semi-honest adversary) We say a protocol π securely (privately) computes $\mathcal{F}_{\mathsf{OT}}$ in the presence of static semi-honest adversaries if there exist probabilistic polynomial-time algorithms S_1, S_2 such that*

$$output^\pi(x, y) = \mathcal{F}_{\mathsf{OT}}(x, y)$$

$$\{S_1(x, f_1(x, y))\}_{x,y} =_c \{view_1^\pi(x, y)\}_{x,y}$$

and

$$\{S_2(y, f_2(x, y))\}_{x,y} =_c \{view_2^\pi(x, y)\}_{x,y}.$$

The notion implies that whatever the semi-honest adversary can learn from running the protocol, it could be generated by themself without the execution. In other words, the semi-honest adversary can learn nothing more than allowed. The idea of ideal execution is implicit here. Since anything apart from the output of the functionality can be self-generated in an indistinguishable manner, the real protocol ideally realizes the functionality as long as the two parties follow the protocol specification (see Section 7.2 of [Ode09] for more details).

However, the semi-honest adversary model is not sufficient. It is inevitable in the real world that malicious users depart from the protocol specification with arbitrary strategies. A relaxation for oblivious transfer protocols or single-output functionalities is one-sided simulation. One-sided simulation requires the indistinguishability for the sender and the simulation for the receiver. Since the sender has no outputs, the notion ensures privacy for both parties in the presence of malicious adversaries. It is also a plausible choice for an efficient construction in the stand-alone model. Here, we consider full-simulation in the presence of malicious adversaries.

Roughly speaking, the standard real/ideal paradigm demonstrates that for any adversary in the real world, there exists a corresponding simulator in the ideal world such that the outputs from the two worlds are indistinguishable. The notion provides an ultimate guarantee that whatever the adversary can do in the real execution is simulatable in the ideal world. Since the execution in the ideal world is secure, the real execution is secure as well. To see this, we need to clarify the definitions of the real and ideal executions.

Ideal Execution. The ideal execution captures a world where a trusted third party exists. The parties do not communicate with each other but instead hand their inputs to the trusted party. Then, the trusted party honestly returns the outcomes to each party, corresponding to the defined functionality. Nevertheless, the ideal execution in the presence of malicious adversaries is slightly different from the previous consideration of the semi-honest adversary. Due to losing the honest majority, fairness is not taken into consideration. Moreover, rational rebelling behaviors of the malicious adversaries, including refusing to participate, aborting the running sessions, or replacing the inputs, are taken into account. These strategies will be taken into account in the definition of the modified ideal functionality.

We define the modified ideal execution before going to the security definition. For more detailed exposition, also see [HL10, Lin17]. The ideal execution in consideration of a malicious adversary of a two-party functionality $\mathcal{F} = (f_1, f_2)$ consists of six phases: *initial inputs, inputs to the trusted party, early abortion, output to the adversary, instruction of continuing or halting, outputs*. Let P_i denote the corrupted party controlled by $\mathcal{S}$, P_j be the honest party where $\{i, j\} = \{1, 2\}$, $\mathcal{T}$ be the trusted third party.

First of all, in the phase of *initial inputs*, like the ordinary setup, P_1 has the input x, P_2 has the input y and the adversary $\mathcal{S}$ has an auxiliary input z. Secondly, in the phase of *inputs to the trusted party*, honest P_j hands the initial input (x or y) to $\mathcal{T}$. What corrupted P_i sends is controlled by $\mathcal{S}$. The decision made by $\mathcal{S}$ including the early abortion option $\mathbf{abort}_i$ is based on the initial input of P_i and the auxiliary input z. Let (x', y') be the inputs to $\mathcal{F}$. Thirdly, *early abortion* is an intermediate phase, if $\mathbf{abort}_i$ is sent within the second phase by $\mathcal{S}$. Then the trusted party returns $\mathbf{abort}_i$ to both parties, and the execution terminates; otherwise, the execution continues. Fourthly, in the phase of *output to the adversary*, $\mathcal{T}$ computes $f_1(x', y')$ and $f_2(x', y')$ and returns $f_i(x', y')$ to the corrupted party P_i first. Next, in the fifth phase, the adversary replies **continue** or $\mathbf{abort}_i$ to $\mathcal{T}$. This instructs $\mathcal{T}$ to continue or terminate by returning $f_j(x', y')$ or $\mathbf{abort}_i$ to P_j, respectively. Last but not least, in the final phase *outputs*, the honest party outputs $f_j(x', y')$. The adversary $\mathcal{S}$ in place of P_i outputs something based on the knowledge of the initial input (x or y), auxiliary input z, and $f_i(x', y')$.

The output pair of the honest party and the adversary from the ideal execution of the functionality $\mathcal{F}$ described above is denoted by $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}(z),i}(x, y)$. Note that even though in oblivious transfer the sender receives no outputs from the trusted party, the adversary can still output something in place of the sender if the sender is the corrupted party. For readability, we will write in the protocol description $\mathbf{abort}_{\mathsf{S}}, \mathbf{abort}_{\mathsf{R}}$ representing aborts made by the receiver and the sender to replace $\mathbf{abort}_1, \mathbf{abort}_2$, respectively.

Real Execution. The real execution is the execution of a real protocol. Let the protocol π compute the functionality $\mathcal{F}$ where P_i is the corrupted party controlled by the adversary $\mathcal{A}$. The initial inputs are x for P_1, y for P_2 and the auxiliary input z for $\mathcal{A}$. During the execution of π, $\mathcal{A}$ will usurp P_i, interact with P_j, and finally output something. The messages and output provided by the adversary may deviate from the specification of π by a polynomial-time strategy. In contrast, the honest party P_j interacts with P_i and returns outputs as specified by the protocol. Let $\text{REAL}_{\pi,\mathcal{A}(z),i}(x,y)$ denote the output pair by P_j and $\mathcal{A}$.

The aim of the standard real/ideal paradigm is to show that the ensemble produced by the simulator through the ideal execution is indistinguishable from the ensemble produced by the adversary via the real execution. This provides strong assurance of the security irrespective of the strategies the adversary adopts since any real adversary can be simulated in the ideal world. This also permits modular constructions for larger protocols by the composition theorems [Can00, Can01]. As a corollary, a relaxed but equivalent version of the security model is the simulation in the *hybrid model*.

Hybrid Model. The hybrid model contains real messages communicated between participants and oracle access to functionality $\mathcal{G}$ (ideal messages). The two-party protocol π with input (x,y) in a hybrid model with the functionality $\mathcal{G}$ is called the $\mathcal{G}$-hybrid model. In the presence of adversary $\mathcal{A}$ who controls the i^{th} party with the auxiliary input z, we denote the output of all parties by $HYBRID^{\mathcal{G}}_{\pi,\mathcal{A}(z),i}(x,y)$.

We remark that this model is a prerequisite for constructing UC-secure oblivious transfer due to the impossibility results given in [CKL03]. In the $\mathcal{G}$-hybrid model, the simulator in the simulation process is able to exert control over the functionality $\mathcal{G}$. For example, in the common reference string (CRS) hybrid model, two parties are given a shared string in the protocol execution, while in the simulation process, the simulator can invoke the adversary with a trapdoor string to cheat [PVW08].

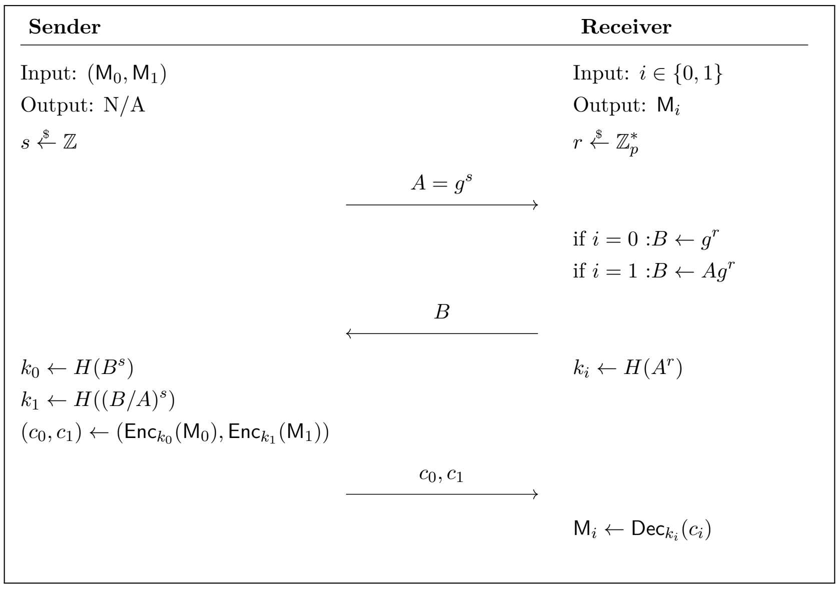
To match the security definition presented in [Can01], assume there exists an environment machine $\mathcal{Z}$ serving as an interactive distinguisher between the real execution and the ideal execution. When interacting with the machine, the environment $\mathcal{Z}$ can decide all inputs of the parties and the auxiliary input for the adversary/simulator. After the execution, $\mathcal{Z}$ outputs a single bit to judge whether it interacts with a real machine or an ideal machine. Also, the environment $\mathcal{Z}$ can interact with the adversary/simulator with any queries at any time throughout the execution in order to distinguish. Here, we denote the ensemble consisting of the output of the ideal execution of the functionality $\mathcal{F}$ involving the adversary $\mathcal{S}$, the environment $\mathcal{Z}$ by $IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ and the ensemble consisting of the outputs in the hybrid model involving the adversary $\mathcal{A}$ and the environment $\mathcal{Z}$ by $HYBRID^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}$.

Definition 4.2.5. *(UC-realize) A protocol π is said to UC-realize an ideal functionality $\mathcal{F}$ in the presence of malicious adversaries and static corruption in the hybrid model with functionality $\mathcal{G}$ if for any adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that for every interactive distinguisher environment $\mathcal{Z}$ we have*

$$IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}} =_c HYBRID^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}.$$

The advantage of an environment machine $\mathcal{Z}$ is defined as

$$\mathsf{Adv}(\mathcal{Z}) = |\Pr[\mathcal{Z}\ wins] - 1/2|\,.$$



Sender
Receiver
Input: $(\mathsf{M}_0, \mathsf{M}_1)$
Output: N/A
$s \xleftarrow{\$} \mathbb{Z}$
Input: $i \in \{0, 1\}$
Output: M_i
$r \xleftarrow{\$} \mathbb{Z}_p^*$
$\xrightarrow{A = g^s}$
if $i = 0 : B \leftarrow g^r$
if $i = 1 : B \leftarrow Ag^r$
$\xleftarrow{B}$
$k_0 \leftarrow H(B^s)$
$k_1 \leftarrow H((B/A)^s)$
$(c_0, c_1) \leftarrow (\mathsf{Enc}_{k_0}(\mathsf{M}_0), \mathsf{Enc}_{k_1}(\mathsf{M}_1))$
$k_i \leftarrow H(A^r)$
$\xrightarrow{c_0, c_1}$
$\mathsf{M}_i \leftarrow \mathsf{Dec}_{k_i}(c_i)$

Figure 4.1: Chou and Orlandi's OT scheme in a nutshell [CO15]

4.3 Construction

This section first presents the idea behind our tweaked key exchange by introducing the core of Chou and Orlandi's OT scheme [CO15]; we then derive a novel compact protocol as a prototype. Following this, we compress the three-round scheme to an optimal two rounds by using the quadratic twist technique. Finally, building on the round-optimal structure, we add a "proof of decryption" mechanism, based on the idea of the Fujisaki-Okamoto transform [FO99], which requires two extra rounds, in order to achieve security against malicious adversaries.

4.3.1 Passively Secure Schemes

Tweaked Key Exchange

Figure 4.1 presents the Chou–Orlandi OT scheme [CO15] which is based on Diffie–Hellman key exchange. In Diffie–Hellman, the sender and the receiver first share their public "keys", g^s and g^r, with each other, after which both of them can secretly obtain a shared secret g^{rs}. To adapt this for the purpose of OT, the receiver can use the second round to hide his secret bit i. In the third round, the sender can communicate an encryption of the two OT messages by deriving two keys, one which cancels out the obfuscation, and one which does not. Because of this key derivation, the receiver can then only decrypt the message corresponding to his input bit.

Proposals by de Saint Guilhem et al. and Vitse rely on a similar idea to use a fixed key from the key exchange to decrypt the chosen ciphertext [DOPS20, Vit19]. In the first OT construction of [DOPS20], two public curves are required as a trusted setup, which serve the same role as two fixed keys from the perspective of key exchange. In [Vit19], one more $p_2^{e_2}$-torsion subgroup

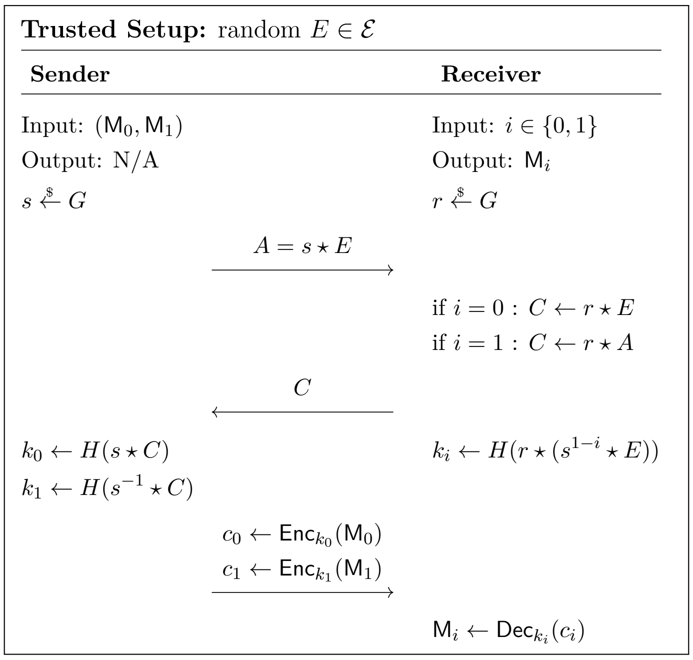

Trusted Setup: random $E \in \mathcal{E}$
Sender
Receiver
Input: $(\mathsf{M}_0, \mathsf{M}_1)$
Output: N/A
$s \xleftarrow{\$} G$
Input: $i \in \{0,1\}$
Output: M_i
$r \xleftarrow{\$} G$
$\xrightarrow{A = s \star E}$
if $i = 0 : C \leftarrow r \star E$
if $i = 1 : C \leftarrow r \star A$
$\xleftarrow{C}$
$k_0 \leftarrow H(s \star C)$
$k_1 \leftarrow H(s^{-1} \star C)$
$k_i \leftarrow H(r \star (s^{1-i} \star E))$
$c_0 \leftarrow \mathsf{Enc}_{k_0}(\mathsf{M}_0)$
$c_1 \leftarrow \mathsf{Enc}_{k_1}(\mathsf{M}_1)$
$\mathsf{M}_i \leftarrow \mathsf{Dec}_{k_i}(c_i)$

Figure 4.2: Our base 3-round OT protocol.

generated by the sender is required to obtain two fixed keys (which is based on SIDH and therefore insecure now).

Our three-round Protocol

We present our three-round protocol in Figure 4.2 using the notation of an EGA $(G, \mathcal{E}, E_0, \star)$ In this work we approach the change from key exchange to OT with a different strategy. The essence is that the sender and the receiver can exponentiate by both s and by s^{-1}, and by both r and r^{-1} respectively.

Upon receiving g^s from the sender, the receiver computes both g^r and g^{sr}, and sends one of them to the sender depending on its choice bit. The sender then exponentiates it by both s and by s^{-1} as the encryption keys, which is like doing key exchange twice. One can verify that the *shared secret* in each case is g^{rs} and g^r, respectively.

The other encryption keys are $g^{rs^{-1}}$ and g^{rs^2}, respectively. They are intractable to the honest-but-curious receiver due to the hardness of the inverse and square CSIDH problems, respectively. Furthermore, the receiver's input bit remains unknown since the sender only knows either g^r or g^{sr}.

Note that in this isogeny-based setting, it is necessary that the relation between the shared public curve $E \in \mathcal{E}$ and a fixed base curve E_0 remains unknown. Should the receiver know that $E = t \star E_0$, then he can always input $i = 0$ and compute the other key as $t^2r^2 \star (rs \star E)^t = t^2r^2 \star (trs \star E_0)^t = trs^{-1} \star E_0 = rs^{-1} \star E$.

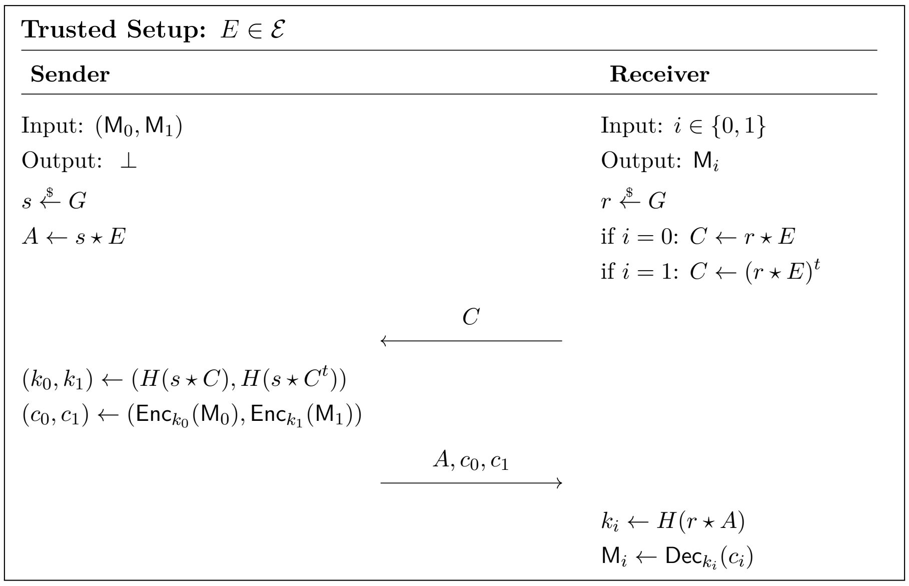

Trusted Setup: $E \in \mathcal{E}$
Sender
Receiver
Input: $(\mathsf{M}_0, \mathsf{M}_1)$
Output: $\perp$
$s \xleftarrow{\$} G$
$A \leftarrow s \star E$
Input: $i \in \{0, 1\}$
Output: M_i
$r \xleftarrow{\$} G$
if $i = 0$: $C \leftarrow r \star E$
if $i = 1$: $C \leftarrow (r \star E)^t$
C
$(k_0, k_1) \leftarrow (H(s \star C), H(s \star C^t))$
$(c_0, c_1) \leftarrow (\mathsf{Enc}_{k_0}(\mathsf{M}_0), \mathsf{Enc}_{k_1}(\mathsf{M}_1))$
A, c_0, c_1
$k_i \leftarrow H(r \star A)$
$\mathsf{M}_i \leftarrow \mathsf{Dec}_{k_i}(c_i)$

Figure 4.3: The core of our two-round OT scheme. No analogue exists in the Diffie–Hellman setting due to the use of the quadratic twist.

Our two-Round Protocol

To address the drawbacks of our three-round protocol, we observe that the quadratic twist provides additional flexibility for the curve computations.

To first break the dependency of C on A, we let the receiver compute $C = (r \star E)^t$ in the case $i = 1$, instead of $r \star A$. Now that C is independent of A, the receiver can send his message first, reducing the protocol to only two rounds. Furthermore, this removes the hypothetical attack of a malicious receiver choosing C in response to A and enables a direct reduction to the computational CSIDH problem.

We then note that the sender's second encryption curve can be computed as $(s \star C^t)^t$, instead of $s^{-1} \star C$, in the three-round version. Here again we can simplify by letting the sender compute the second curve as $s \star C^t$, without the additional twisting operation. This then results in a simplification for key computation too: for $i = 0$, the encryption curve is $s \star (r \star E) = r \star A$, and for $i = 1$ it is $s \star ((r \star E)^t)^t = r \star A$; thus we return to the idea of using a single Diffie–Hellman key by way of using the twist operation. The modified two-round protocol is described in Figure 4.3. We give a formal security proof in Section 4.4.1.

In this simplified variant the number of isogeny computations remains the same as in the three-round variant. We note that taking quadratic twists is an efficient operation via field negation.

4.3.2 The Full Construction Against Malicious Adversaries

In the previous construction, when the receiver follows the specification, the simulator is able to extract the input by observing the random oracle queries. However, this scenario changes in the malicious model, where a corrupted receiver can intentionally delay the query process. This subtle difference complicates the analysis because the encrypted messages have already been sent to the receiver before the extraction. Therefore, we deploy a mechanism based on an idea similar to the Fujisaki-Okamoto transform, where the receiver has to decrypt a random message independent of the sender's input. By deploying the mechanism, we can extract the receiver's input prior to the transmission of encrypted messages. For the case of the corrupted sender, we use a new trick of the quadratic twists in the trusted setup curve to extract the input of the corrupted sender. The full protocol shown in Figure 4.4 below is based on the following building blocks:

- CSIDH EGA $(G, \mathcal{E}, E_0, \star)$ where G acts freely and transitively on the set of supersingular elliptic curves $\mathcal{E}$ defined over $\mathbb{F}_p$ where $p = 3 \mod 4$. There are an efficient algorithm to verify whether a given curve is in $\mathcal{E}$ and a sampling method over G that is indistinguishable to the uniform sampling. The group element in G can be encoded into $\{0,1\}^{\tau\lambda}$ as a $\tau\lambda$-bit string for some $\tau \in \mathbb{R}$ [1].

- Symmetric encryptions $(\mathsf{Enc}, \mathsf{Dec})$ and $(\mathsf{Enc}', \mathsf{Dec}')$ where $(\mathsf{Enc}, \mathsf{Dec})$ is IND-CPA and with key space $\mathcal{K}$ and $(\mathsf{Enc}', \mathsf{Dec}')$ is with message space and key space $\{0,1\}^{(\tau+1)\lambda}$ defined by $\mathsf{Enc}'_k(m) := m \oplus k$ and $\mathsf{Dec}'_k(c) := c \oplus k$.

- Hash functions $H_{\mathsf{Key}} : \mathcal{E} \to \mathcal{K}$, $H_{\mathsf{Enc}} : \mathcal{E} \to \{0,1\}^{(\tau+1)\lambda}$, which are modeled by a random oracle $\mathcal{F}_{RO}$ as $\mathcal{F}_{\mathsf{RO}}(\mathsf{Key} \parallel \cdot), \mathcal{F}_{\mathsf{RO}}(\mathsf{Enc} \parallel \cdot)$, respectively. The former serves as the key derivation function from the domain $\mathcal{E}'$ to the key space $\mathcal{K}$ for $(\mathsf{Enc}, \mathsf{Dec})$. The latter serves as a PRNG to generate a masking string to encrypt for the encryption scheme $(\mathsf{Enc}', \mathsf{Dec}')$.

Protocol. ***(CSIDH-based OT)***

- ***Trusted Setup:*** *Let $E = t \star E_0$ where $t \xleftarrow{\$} G$ is not given to anyone.*

- ***Input:*** *As input, the sender $\mathcal{S}$ takes two messages M_0, M_1 of the same length; the receiver $\mathcal{R}$ takes a bit $\mathsf{i} \in \{0,1\}$.*

- ***Procedure:***

  1. *$\mathcal{S}$ samples independent ideals $s_0, s_1 \xleftarrow{\$} G$, a random string $\mathsf{str} \xleftarrow{\$} \{0,1\}^{\lambda}$ and computes $A_0 = s_0 \star E$, $A_1 = s_1 \star E$.*

  2. *$\mathcal{R}$ generates $r \xleftarrow{\$} G$ and computes $C = r \star E$; if $\mathsf{i} = 1$, overwrites $C = C^t$; and sends C to $\mathcal{S}$.*

  3. *$\mathcal{S}$ checks whether $C \in \mathcal{E}$. If not, $\mathcal{S}$ aborts and outputs $\mathbf{abort}_{\mathsf{S}}$. Otherwise, $\mathcal{S}$ computes masking keys $k_{1,0} = H_{\mathsf{Enc}}(s_1 \star C)$ and $k_{1,1} = H_{\mathsf{Enc}}(s_1 \star C^t)$. Then, $\mathcal{S}$ computes two ciphertexts $c_{1,j} \leftarrow \mathsf{Enc}'_{k_{1,j}}(s_1 \parallel \mathsf{str})$ for $j \in \{0,1\}$. $\mathcal{S}$ sends $(A_1, c_{1,0}, c_{1,1})$ to $\mathcal{R}$.*

[1]The size of G is approximately $\sqrt{p}$ [CLM+18]. For instance, one can take $\tau\lambda = 256$ for CSIDH-512.

4. *$\mathcal{R}$ runs the proof of ability to decrypt mechanism. Firstly, $\mathcal{R}$ checks whether $A_1 \in \mathcal{E}$. If not, $\mathcal{R}$ aborts and outputs $\mathbf{abort}_{\mathsf{R}}$. Otherwise, $\mathcal{R}$ computes $k'_{1,\mathsf{i}} = H_{\mathsf{Enc}}(r \star A_1)$ and $(s'_1 \parallel \mathsf{str}') \leftarrow \mathsf{Dec}'_{k'_{1,\mathsf{i}}}(c_{1,\mathsf{i}})$. Verify whether $s'_1 \star (r \star E) = r \star A_1$. If not, output $\mathbf{abort}_{\mathsf{R}}$. Otherwise, continue.*
5. *$\mathcal{R}$ computes $k'_{1,1-\mathsf{i}} = H_{\mathsf{Enc}}(s'_1 \star (r \star E)^t)$. Verify whether $\mathsf{Dec}'_{k'_{1,1-\mathsf{i}}}(c_{1,1-\mathsf{i}}) = (s'_1 \parallel \mathsf{str}')$. If not, return $\mathbf{abort}_{\mathsf{R}}$. Otherwise, send str' to $\mathcal{S}$.*
6. *$\mathcal{S}$ checks whether $\mathsf{str} = \mathsf{str}'$. If not, $\mathcal{S}$ aborts and outputs $\mathbf{abort}_{\mathsf{S}}$. Otherwise, $\mathcal{S}$ computes keys $k_{0,0} = H_{\mathsf{Key}}(s_0 \star C)$ and $k_{0,1} = H_{\mathsf{Key}}(s_0 \star C^t)$. Then, $\mathcal{S}$ computes ciphertexts $c_{0,j} \leftarrow \mathsf{Enc}_{k_{0,j}}(\mathsf{M}_j)$ for $j \in \{0,1\}$. $\mathcal{S}$ sends $(A_0, c_{0,0}, c_{0,1})$ to $\mathcal{R}$ and outputs $\perp$.*
7. *$\mathcal{R}$ verifies $A_0 \in \mathcal{E}$. If not, $\mathcal{R}$ aborts and outputs $\mathbf{abort}_{\mathsf{R}}$. Otherwise, $\mathcal{R}$ computes the decryption key $k'_{0,\mathsf{i}} = H_{\mathsf{Key}}(r \star A_0)$ and outputs $\mathsf{M}'_{\mathsf{i}} \leftarrow \mathsf{Dec}_{k'_{0,i}}(c_{0,i})$.*

Intuitively, to simulate a sender controlled by an adversary, we have to show that the receiver's message's distribution with input $i = 0$ and that with input $i = 1$ are indistinguishable. Asides from that, the simulator needs to extract the real input of the message pair since the adversary can replace the original input. The uniform sampling over G assures the first requirement. We meet the second condition by setting up a trapdoor of the functionality $\mathcal{F}_{\mathsf{TSC}}$, which allows the simulator to decrypt two ciphertexts by using the trapdoor and extract the real input of the sender.

To simulate a receiver corrupted by an adversary, the simulator should extract the adversary's input by observing the hash queries. In order to extract the input, the receiver should demonstrate the ability to decrypt. The reason to do this is that the corrupted receiver who skips all hash queries makes the input inaccessible to the simulator and gives all information to the environment machine. The additional "proof of ability to decrypt" requires the corrupted receiver to show that he can decrypt at least one message to get a random string. The mechanism also ensure privacy for an honest receiver so that a corrupted sender cannot manipulate the mechanism to extract any useful information from an honest receiver.

Here the sender will send another curve $s_1 \star E$ distinct from $s_0 \star E$ for transferring messages. The sender encrypts the group element s_1 and a concatenated random string str by using a key pair derived from $s' \star E$. The receiver decrypts one ciphertext with r, and the other ciphertext serves as a verification of the equality of encrypted messages to ensure his privacy. By assuming Problem 9, the mechanism enables the simulator to extract the correct input by observing the random oracle queries. The difference between the unchosen ciphertexts is not noticeable unless the environment machine knows the corresponding decryption key. In this case, the environment machine contains a pair of curves which is exactly the solution for the reciprocal CSIDH problem. See Section 4.4 for more details.

Trusted Setup: $E \in \mathcal{E}$

Sender		**Receiver**
Input: $(\mathsf{M}_0, \mathsf{M}_1)$, **Output:** $\perp$		**Input:** $\mathsf{i} \in \{0,1\}$, **Output:** M_i
$s_0, s_1 \xleftarrow{\$} G$		$r \xleftarrow{\$} G$
$(A_0, A_1) \leftarrow (s_0 \star E, s_1 \star E)$		If $\mathsf{i} = 0$: $C \leftarrow r \star E$
$\mathsf{str} \xleftarrow{\$} \{0,1\}^\lambda$		If $\mathsf{i} = 1$: $C \leftarrow (r \star E)^t$
	$\xleftarrow{\quad C \quad}$	
If $C \notin \mathcal{E}$: **abort**$_\mathsf{S}$.		
$(k_{1,0}, k_{1,1}) \leftarrow (H_{\mathsf{Enc}}(s_1 \star C), H_{\mathsf{Enc}}(s_1 \star C^t))$		
$(c_{1,0}, c_{1,1}) \leftarrow (\mathsf{Enc}'_{k_{1,0}}(s_1 \parallel \mathsf{str}), \mathsf{Enc}'_{k_{1,1}}(s_1 \parallel \mathsf{str}))$		
	$\xrightarrow{A_1, c_{1,0}, c_{1,1}}$	
		If $A_1 \notin \mathcal{E}$: **abort**$_\mathsf{R}$.
		$k'_{1,\mathsf{i}} \leftarrow H_{\mathsf{Enc}}(r \star A_1)$
		$(s'_1 \parallel \mathsf{str}') \leftarrow \mathsf{Dec}'_{k'_{1,\mathsf{i}}}(c_{1,\mathsf{i}})$
		If $s'_1 \star (r \star E) \neq r \star A_1$: **abort**$_\mathsf{R}$.
		$k'_{1,1-\mathsf{i}} \leftarrow H_{\mathsf{Enc}}(s'_1 \star (r \star E)^t)$
		If $\mathsf{Dec}'_{k'_{1,1-\mathsf{i}}}(c_{1,1-\mathsf{i}}) \neq (s'_1 \parallel \mathsf{str}')$: **abort**$_\mathsf{R}$.
	$\xleftarrow{\quad \mathsf{str}' \quad}$	
If $\mathsf{str} \neq \mathsf{str}'$: **abort**$_\mathsf{S}$.		
$(k_{0,0}, k_{0,1}) \leftarrow (H_{\mathsf{Key}}(s_0 \star C), H_{\mathsf{Key}}(s_0 \star C^t))$		
$(c_{0,0}, c_{0,1}) \leftarrow (\mathsf{Enc}_{k_{0,0}}(\mathsf{M}_0), \mathsf{Enc}_{k_{0,1}}(\mathsf{M}_1))$		
	$\xrightarrow{A_0, c_{0,0}, c_{0,1}}$	
Return: $\perp$		If $A_0 \notin \mathcal{E}$: **abort**$_\mathsf{R}$.
		$k'_{0,\mathsf{i}} \leftarrow H_{\mathsf{Key}}(r \star A_0)$
		Return: $\mathsf{M}'_\mathsf{i} \leftarrow \mathsf{Dec}_{k'_{0,\mathsf{i}}}(c_{0,\mathsf{i}})$

Figure 4.4: Our CSIDH-based oblivious transfer protocol secure against malicious adversaries.

4.4 Security Analysis

In this section, we prove the security of our two schemes from Sections 4.3.1 and 4.3.2 against semi-honest and malicious adversaries respectively.

4.4.1 Semi-honest security

Eavesdropper. An eavesdropper receives all the communications of parties and does not intervene in the execution. We assume that such an adversary knows the parties' inputs while the simulator tasked with simulating an indistinguishable transcript is given nothing. The reason for this assumption is to match the definition of UC-security [Can01] where the environment machine decides the inputs. In fact, security against such eavesdroppers corresponds exactly to the honest-honest case discussed in the proof below.

Semi-Honest Adversary. A static semi-honest adversary can choose to corrupt either, both or neither of the parties and will follow the protocol specification. We will prove that such an adversary cannot obtain any information from the transcript of our two-round protocol (Figure 4.3) assuming that the computational inverse CSIDH problem is hard. The property remains the same for the four-round protocol (Figure 4.4) since the second and the third messages are simulatable and independent of the input.

Theorem 4.4.1. *The protocol π of Figure 4.3 securely computes $\mathcal{F}_{\mathsf{OT}}$ in the presence of static semi-honest adversaries if the computational inverse CSIDH problem (Problem 7) is infeasible, assuming that $H(\cdot)$ is a random oracle and the encryption scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is IND-CPA.*

Proof. (**Correctness**) Let $i \in \{0, 1\}$ be the input of the receiver $\mathcal{R}$. Say the sender $\mathcal{S}$ generates ideal $s \in G$ and $\mathcal{R}$ generates $r \in G$. If $i = 0$, then $C = r \star E$. $\mathcal{S}$ computes the encryption key k_0 as $H(s \star C)$, and sends $A = s \star E$. $\mathcal{R}$ computes $k'_0 = H(r \star A)$ as the decryption key; as we have $r \star A = r \star (s \star E) = s \star (r \star E) = s \star C$, we indeed have $k'_0 = k_0$. On the other hand, if $i = 1$, then $C = (r \star E)^t$. $\mathcal{S}$ computes $k_1 = H(s \star C^t)$ while $\mathcal{R}$ computes $k'_1 = H(r \star A)$. We have $s \star C^t = s \star ((r \star E)^t)^t = s \cdot r \star E = r \star A$ which implies $k'_1 = k_1$ and shows the correctness of the protocol.

(**Corrupt sender $\mathcal{S}^*$**) The simulator S_1 takes as input $(M_0, M_1, \perp)$ and is required to simulate the view $view_1^\pi(M_0, M_1, i) = (M_0, M_1, rp, C)$ where rp is a random tape. To generate this, S_1 performs these steps:

1. Uniformly generate a random tape rp for $\mathcal{S}^*$.
2. Generate $r' \xleftarrow{\$} G$ acting as an honest $\mathcal{R}$ and using a private random tape.
3. Output $(M_0, M_1, rp, C' = r' \star E)$.

In a real execution, the curve C sent by the honest receiver is either $r \star E$ if $i = 0$, or $(r \star E)^t$ if $i = 1$. In the first case, the transcript output by S_1 is identically distributed to that produced by a real execution. The second case, due to the uniform sampling, the distribution is also identical to the real one regardless of i. Thus, any distinguisher (possibly unbounded) that is given a tuple (M_0, M_1, i) is not able to distinguish $\{S_1((M_0, M_1), \perp)\}_{(M_0,M_1),i}$ from $\{view_1^\pi(M_0, M_1, i)\}_{M_0,M_1,i}$.

(**Corrupt receiver $\mathcal{R}^*$**) The simulator S_2 takes as input (i, M_i) and is required to simulate the view $view_2^\pi(M_0, M_1, i) = (i, rp, A, c_0, c_1)$ where rp is a random tape. To generate this, S_2 performs these steps:

1. Choose a uniform generated random tape rp for $\mathcal{R}^*$.
2. Generate $s' \xleftarrow{\$} G$ acting as an honest $\mathcal{S}$ and using a private random tape, and generate $r' \xleftarrow{\$} G$ using rp. Compute the curve C as $r' \star E$ or $(r' \star E)^t$ depending on i.

3. Compute the decryption keys k'_i, k'_{1-i} honestly using s' and C. Replace k'_{1-i} with $\widetilde{k'} \xleftarrow{\$} \mathcal{K}$

4. Compute ciphertexts $c_i = \mathsf{Enc}_{k'_i}(M_i)$ and $c_{1-i} = \mathsf{Enc}_{\widetilde{k'}}(\widetilde{M})$ where $\widetilde{M}$ is a string of the same length as M_i sampled at random from the message space $\mathcal{M}$.

5. Output $(i, rp, s' \star E, c_0, c_1)$.

We claim that if there exists a successful PPT distinguisher between the simulated view and the real view, then reductions can be made to solve the computational problems (Problem 6 or the equivalent Problem 7) or to break the IND-CPA security of the encryption scheme.

To show this, we build a series of hybrid views. Let $\mathcal{H}_0$ be the view of the real adversary, and $\mathcal{H}_2$ be the view generated by S_2 (i.e., $\{view_2^{\pi}(M_0, M_1, i)\}_{(M_0,M_1),i}$ and $\{S_2((M_0, M_1), \perp)\}_{(M_0,M_1),i}$, resp). Let the intermediate $\mathcal{H}_1$ be the view produced by running a real execution and replacing the encryption key k_{1-i} with a random $\widetilde{k} \xleftarrow{\$} \mathcal{K}$. The difference between $\mathcal{H}_1$ and $\mathcal{H}_2$ is then that the real message M_{1-i} is replaced with a random one $\widetilde{M} \xleftarrow{\$} \mathcal{M}$.

Hybrid 1 ($\mathcal{H}_1$). We first claim $\mathcal{H}_0 \approx_c \mathcal{H}_1$ if the computational inverse CSIDH problem (Problem 7) is hard. To offer an intuition: let E_{1-i} denote the curve from which the replaced key k_{1-i} is derived. When $i = 0$, we have $E_{1-i} = s \star C^t = s \star (r \star E)^t = r^{-1} \star (s^{-1} \star E)^t$; and when $i = 1$, we have $E_{1-i} = s \star C = s \star (r \star E)^t = r^{-1} \star (s^{-1} \star E)^t$ as well. In both cases we see that the hard-to-compute curve contains $s^{-1} \star E$ which we use to reduce a successful distinguisher to the computational inverse CSIDH problem (Problem 7).

Let $\mathcal{Z}$ be an environment that can successfully distinguish between $\mathcal{H}_0$ and $\mathcal{H}_1$, then a solver $\mathcal{B}$ for Problem 7 with the assistance of $\mathcal{Z}$ runs as follows:

1. Receive challenge $(E', s' \star E')$ from Problem 7, where $s' \in G$ is unknown.

2. Set E' to be the public curve used by the protocol π and set $s' \star E'$ as the curve A sent to the receiver.

3. Randomly generate random tape rp for the receiver, use it to sample r, and compute C according to i.

4. While running, simulate the random oracle by assigning a random value from $\mathcal{K}$ whenever a new query is made and recording a list of past queries during the execution.

5. When deriving the real encryption key k_i, compute it as $r \star (s' \star E')$ (since s' from the challenge is unknown).

6. Replace the other encryption key k_{1-i} with $\widetilde{k} \xleftarrow{\$} \mathcal{K}$ to simulate the output of $\mathcal{H}_1$; abort if $\widetilde{k}$ already appears on the list of answers to random oracle queries.

7. Invoke the distinguisher $\mathcal{Z}$ with the produced output of $\mathcal{H}_1$.

8. When $\mathcal{Z}$ terminates, randomly select a curve $\widetilde{E}$ in the list of past queries of the simulated random oracle and return $(r \star \widetilde{E})^t$ as the computational inverse CSIDH solution.

Note that, if $\mathcal{B}$ does not abort, the only difference between $\mathcal{H}_0$ and $\mathcal{H}_1$ is the key for M_{i-1}, thus a distinguisher $\mathcal{Z}$ which does not query this key must have a zero advantage.

Let $\mathbf{A}$ denote the event that $\mathcal{B}$ aborts when sampling the replacement key. Denoting by q_H the maximum number of queries made to H during the reduction, we have that $\Pr[\mathbf{A}] \leq \frac{q_H}{|\mathcal{K}|}$. Also let $\mathbf{E}$ denote the event that the targeted curve $E'_{1-i} = r^{-1} \star (s^{-1} \star E')^t$ is present on the query list. We see that the reduction $\mathcal{B}$ wins with probability $1/q_H$ when $\mathbf{E}$ happens, and we can then write:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{iCDH}}(\mathcal{B}) = \Pr[\mathcal{B} \text{ wins}] &= \Pr[\mathcal{B} \text{ wins} \mid \neg\mathbf{A}] \cdot \Pr[\neg\mathbf{A}] + \Pr[\mathcal{B} \text{ wins} \mid \mathbf{A}] \cdot \Pr[\mathbf{A}] \\
&\geq \Pr[\mathcal{B} \text{ wins} \mid \neg\mathbf{A}] \cdot (1 - \Pr[\mathbf{A}]) \\
&\geq \Pr[\mathcal{B} \text{ wins} \mid \neg\mathbf{A}] \cdot \left(1 - \frac{q_H}{|\mathcal{K}|}\right) \\
\Leftrightarrow \frac{1}{1 - \frac{q_H}{|K|}} \cdot \Pr[\mathcal{B} \text{ wins}] &\geq \Pr[\mathcal{B} \text{ wins} \mid \neg\mathbf{A}] = \frac{1}{q_H} \cdot \Pr[\mathbf{E}].
\end{aligned} \tag{4.1}$$

Looking an arbitrary distinguisher $\mathcal{Z}$, we then have

$$\begin{aligned}
|\Pr[\mathcal{Z}(\mathcal{H}_0) = 1] - \Pr[\mathcal{Z}(\mathcal{H}_1) = 1]| = &\ |\Pr[\mathcal{Z}(\mathcal{H}_0) = 1|\mathbf{E}] \cdot \Pr[\mathbf{E}] \\
&- \Pr[\mathcal{Z}(\mathcal{H}_1) = 1|\mathbf{E}] \cdot \Pr[\mathbf{E}] \\
&+ \Pr[\mathcal{Z}(\mathcal{H}_0) = 1|\neg\mathbf{E}] \cdot \Pr[\neg\mathbf{E}] \\
&- \Pr[\mathcal{Z}(\mathcal{H}_1) = 1|\neg\mathbf{E}] \cdot \Pr[\neg\mathbf{E}]| \\
\leq &\ \Pr[\mathbf{E}]
\end{aligned} \tag{4.2}$$

since $|\Pr[\mathcal{Z}(\mathcal{H}_0) = 1|\neg\mathbf{E}] - \Pr[\mathcal{Z}(\mathcal{H}_1) = 1|\neg\mathbf{E}]| = 0$ and $|\Pr[\mathcal{Z}(\mathcal{H}_0) = 1|\mathbf{E}] - \Pr[\mathcal{Z}(\mathcal{H}_1) = 1|\mathbf{E}]| \leq 1$ by definition. By combining (4.1) and (4.2) we see that if $\mathcal{Z}$ distinguishes the two views with non-negligible advantage ϵ, then $\mathcal{B}$ successfully solves Problem 7 with probability at least $\epsilon \cdot (1 - \frac{q_H}{|\mathcal{K}|})/q_H$ which is non-negligible if $q_H = \mathsf{poly}$ and $1/|\mathcal{K}| = \mathsf{negl}$. This contradicts the assumption that Problem 7 is intractable and therefore implies that $\mathcal{H}_0$ and $\mathcal{H}_1$ are computationally indistinguishable to any PPT environment $\mathcal{Z}$.

Hybrid 2. We now claim $\mathcal{H}_1 \approx_c \mathcal{H}_2$ for any PPT distinguisher if the encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ is IND-CPA secure. The only difference is the encryption $\mathsf{Enc}_{\widetilde{k}}(M_{1-i})$ in $\mathcal{H}_1$ and the encryption $\mathsf{Enc}_{\widetilde{k}}(\widetilde{M})$ in $\mathcal{H}_2$, where $\widetilde{k}$ is uniformly sampled from $\mathcal{K}$. A successful distinguisher $\mathcal{Z}$ between the two distributions can be reduced to an adversary against the IND-CPA security of $(\mathsf{Enc}, \mathsf{Dec})$ in a straightforward manner. As this reduction is common in the literature, we only include a sketch here.

The IND-CPA adversary $\mathcal{B}$ has access to a left-right encryption oracle which uses a secret key randomly sampled from $\mathcal{K}$ to encrypt either the left or the right input; this hidden key plays the role of $\widetilde{k}$ in the generation of the view given to $\mathcal{Z}$. After setting up and executing the protocol honestly, $\mathcal{B}$ uses the left-right oracle to encrypt either M_{1-i} or a random $\widetilde{M}$ as the ciphertext c_{1-i}; depending on the hidden bit (left or right) of the oracle, the view $view_{\mathcal{B}}$ generated by $\mathcal{B}$ for $\mathcal{Z}$ is distributed identically to either $\mathcal{H}_1$ or $\mathcal{H}_2$. After the distinguisher terminates, the reduction returns its output as the guess of the oracle's hidden bit. Labelling the oracle's hidden bit as b, we then have

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{(\mathsf{Enc},\mathsf{Dec})}(\mathcal{B}) &= |\Pr[\mathcal{B} = 1 \mid \mathsf{b} = 0] - \Pr[\mathcal{B} = 1 \mid \mathsf{b} = 1]| \\
&= |\Pr[\mathcal{Z}(view_{\mathcal{B}}) = 1 \mid \mathsf{b} = 0] - \Pr[\mathcal{Z}(view_{\mathcal{B}}) = 1 \mid \mathsf{b} = 1]| \\
&= |\Pr[\mathcal{Z}(\mathcal{H}_1) = 1] - \Pr[\mathcal{Z}(\mathcal{H}_2) = 1]|
\end{aligned}$$

which immediately shows that if $\mathcal{Z}$ is successful with non-negligible advantage, then so is $\mathcal{B}$ which contradicts the assumption that $(\mathsf{Enc}, \mathsf{Dec})$ is IND-CPA secure.

(**Honest sender and honest receiver**) We now claim that there exists a PPT simulator that can generate a transcript tuple, without knowledge of the parties' inputs, which is indistinguishable from the view of an eavesdropper $\mathcal{Z}$ that knows the parties' inputs (but not their random tapes). This simulator is constructed from the following sequence:

1. $\mathcal{S}_0$ knows the real inputs (M_0, M_1) and i of the parties; by sampling random tapes and acting honestly, it produces a perfect simulation.

2. $\mathcal{S}_1$ always uses $i = 0$; due to the uniform sampling over G, the simulator $\mathcal{S}_1$ is identically distributed to the output of $\mathcal{S}_0$.

3. $\mathcal{S}_2$ replaces k_1 with a randomly sampled key; as above, this is computationally indistinguishable from the output of $\mathcal{S}_1$ assuming that Problem 7 is intractable.

4. $\mathcal{S}_3$ replaces M_1 with a randomly sampled message; as above, this is computationally indistinguishable from the output of $\mathcal{S}_2$ assuming that the encryption scheme is IND-CPA secure.

5. $\mathcal{S}_4$ always uses $i = 1$; as above, the output of $\mathcal{S}_4$ is statistically indistinguishable from the output of $\mathcal{S}_3$.

6. $\mathcal{S}_5$ and $\mathcal{S}_6$ respectively first replace k_0 and then M_0 with random values; as above, these changes are computationally indistinguishable assuming the hardness of Problem 7 and the IND-CPA security of the encryption scheme.

Finally, we observe that the last simulator $\mathcal{S}_6$ does not use any of the real inputs to produce a random transcript. By the sequence above, this simulation is indistinguishable from the transcript of a real execution.

(**Corrupt sender and corrupt receiver**) In this case, the simulator knows the inputs of both corrupt parties; as for $\mathcal{S}_0$ in the previous case, it can generate a perfect simulation of the views of the parties.

The four cases considered above cover all possible corruption strategies; this thus completes the proof that the protocol π securely computes $\mathcal{F}_{\mathsf{OT}}$. □

4.4.2 Malicious Adversary

Malicious Adversary. A malicious adversary with static corruptions can corrupt either, both or neither of the parties prior to the execution. The environment machine decides the initial inputs of all parties. The adversary will be in charge of the corrupted party or parties, and decide all messages to be sent. In particular, the adversary can replace the inputs of the participants from the environment machine and deviate from the protocol specification. We will prove that the construction in Figure 4.4 UC-realizes the functionality $\mathcal{F}_{\mathsf{OT}}$ in the presence of malicious adversaries with static corruptions.

Theorem 4.4.2. *The protocol π of Figure 4.4, where the encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ is $\mathsf{IND\text{-}CPA}$, securely UC-realizes the functionality $\mathcal{F}_{\mathsf{OT}}$ in the hybrid model with the functionality $\mathcal{F}_{\mathsf{RO}}$ and a trusted setup $\mathcal{F}_{\mathsf{TSC}}$ in the presence of malicious adversaries and static corruption if the computational reciprocal CSIDH problem is infeasible.*

Concretely, for any environment machine $\mathcal{Z}$ with the adversary $\mathcal{A}$ making at most Q queries of $\mathcal{F}_{\mathsf{RO}}$, there exist an IND-CPA adversary $\mathcal{B}_1$, a tweaked reciprocal CSIDH problem adversary $\mathcal{B}_2$, and a reciprocal CSIDH problem adversary $\mathcal{B}_3$ such that

$$\mathsf{Adv}(\mathcal{Z}) \leq \frac{1}{2^\lambda} + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{(\mathsf{Enc},\mathsf{Dec})}(\mathcal{B}_1) + (Q^2 - Q) \times \mathsf{Adv}^{\mathsf{tReGA}}(\mathcal{B}_2) + \frac{Q^2 - Q}{2} \times \mathsf{Adv}^{\mathsf{rCDH}}(\mathcal{B}_3).$$

Proof. **(Honest Sender and Honest Receiver)** We start with the honest sender and the honest receiver. The goal is to show that the execution of π is indistinguishable from the ideal functionality when the parties follow the specification.

By following the same process as the honest-sender-and-honest-receiver case in Theorem 4.4.1, we can construct the simulator that simulates the first and the fourth (final) messages. By continuing the process of $\mathcal{S}_{\mathsf{S}}$ or $\mathcal{S}_4$ in Theorem 4.4.1, the simulator can simulate the second-half messages $A_1, c_{1,0}$ and $c_{1,1}$ by generating s_1 and str. Since the second-half part requires no inputs from either the sender or the receiver, it produces a perfect simulation with respect to the second and the third messages. Therefore, the simulator outputs a transcript indistinguishable from the one of a real execution.

(Corrupted Sender and Corrupted Receiver) When two parties are corrupted, the simulator can invoke the adversary on auxiliary input z and the input $(x = (\mathsf{M}_0, \mathsf{M}_1), y = \mathsf{i})$ given by the environment $\mathcal{Z}$ to run the whole execution. The simulator outputs whatever the adversary outputs for both parties to produce a perfect simulation.

(Honest Sender and Corrupted Receiver) Let $\mathcal{A}$ be the malicious adversary controlling the receiver. In order to emulate the adversary, the simulator needs to extract the input of the adversary, and sends it to the trusted party in the ideal execution. Say the environment $\mathcal{Z}$ generates input $(x = (\mathsf{M}_0, \mathsf{M}_1), y = \mathsf{i}, z)$ and gives y to the simulator/adversary on auxiliary input z. The simulator $\mathcal{S}_{\mathsf{R}}$ passes any query from $\mathcal{Z}$ to $\mathcal{A}$ and returns the output of $\mathcal{A}$. The simulator $\mathcal{S}_{\mathsf{R}}$ on input (y, z) simulates the protocol execution of π with the adversary as follows:

1. Firstly, the simulator $\mathcal{S}_{\mathsf{R}}$ simulates a random oracle $\mathcal{F}_{\mathsf{RO}}$ in an on-the-fly manner. Recall that we have two prefixes $\mathcal{F}_{\mathsf{RO}}(\mathsf{Key} \parallel \cdot)$ and $\mathcal{F}_{\mathsf{RO}}(\mathsf{Enc} \parallel \cdot)$. For $\mathcal{F}_{\mathsf{RO}}(\mathsf{Key} \parallel \cdot)$, it keeps a list L, which is empty initially, over $\{\mathsf{Key}\} \times \mathcal{E} \times \mathcal{K}$ that records each past query. Upon receiving a random oracle query of $(\mathsf{Key}, E') \in \{\mathsf{Key}\} \times \mathcal{E}$, the simulator checks whether $(\mathsf{Key}, E', k') \in L$ for some $k' \in \mathcal{K}$. If not, generate $k' \xleftarrow{\$} \mathcal{K}$ and add (Key, E', k') to L. Finally, $\mathcal{S}_{\mathsf{R}}$ returns k' to the random oracle query of (Key, E'). $\mathcal{S}_{\mathsf{R}}$ does the same process for $\mathcal{F}_{\mathsf{RO}}(\mathsf{Enc} \parallel \cdot)$ where the list is over $\{\mathsf{Enc}\} \times \mathcal{E} \times \{0,1\}^{\tau\lambda}$.

2. Generate the public curve $E = t \star E_0$ by sampling $t \xleftarrow{\$} G$ to setup $\mathcal{F}_{\mathsf{TSC}}$. Invoke the adversary $\mathcal{A}$ on auxiliary input z with the input y and E for the protocol π. $\mathcal{S}_{\mathsf{R}}$ simulates the communication with $\mathcal{Z}$ by forwarding the received message from $\mathcal{Z}$ to $\mathcal{A}$ and forwarding the response of $\mathcal{A}$ back to $\mathcal{Z}$. If $\mathcal{A}$ aborts or halts, then $\mathcal{S}_{\mathsf{R}}$ does the same.

3. $\mathcal{S}_{\mathsf{R}}$ receives a curve C (probably malformed) from the adversary (receiver) in the execution of π. Check whether $C \in \mathcal{E}$, if not, abort the session and output **abort** to the trusted party in the ideal execution and halt. Otherwise, continue.

4. $\mathcal{S}_{\mathsf{R}}$, as an honest receiver in this step, generates $s_1 \xleftarrow{\$} G$, computes $A_1, k_{1,0}, k_{1,1}, c_{1,0}, c_{1,1}$ and sends $A_1, c_{1,0}, c_{1,1}$ to the receiver.

5. $\mathcal{S}_{\mathsf{R}}$ starts to monitor the random oracle queries to see which queries of $s_1 \star C, s_1 \star C^t$ is made by the adversary first. Obtain M_{i} by sending $\widetilde{\mathsf{i}} = 0$ to the trusted party in the ideal execution for the former case and $\widetilde{\mathsf{i}} = 1$ for the latter case. If no such queries are made and the adversary sends the correct string str, then $\mathcal{S}_{\mathsf{R}}$ aborts and halt.

6. $\mathcal{S}_{\mathsf{R}}$ follows the protocol specification π except that it replaces the other message by a random $\widetilde{\mathsf{M}}_{1-\mathsf{i}}$ of the same length and outputs whatever $\mathcal{A}$ outputs to complete the rest of the simulation.

We claim that $\{HYBRID^{\mathcal{F}_{\mathsf{RO}},\mathcal{F}_{\mathsf{TSC}}}_{\pi,\mathcal{A}(z),2}(x,y)\}_{x,y,z} \approx_c \{IDEAL_{\mathcal{F}_{\mathsf{OT}},\mathcal{S}_{\mathsf{R}}(z),2}(x,y)\}_{x,y,z}$.

To see this, we apply a hybrid argument to the UC-security experiment by introducing a series of $\mathsf{Game}_0, \mathsf{Game}_1, \cdots, \mathsf{Game}_4$. Let $\mathsf{Adv}_i(\mathcal{Z})$ denote $\Pr[\mathcal{Z}(\mathsf{Game}_i)$ wins$]$. The two differences from the real execution of π are in the additional abort made in Step 5 and the replacement of one message made in Step 6. We will show in Game_1 the former difference is information-theoretically hard to distinguish. Also, $\mathsf{Game}_2, \mathsf{Game}_3$ the latter difference implies the environment machine can solve either the IND-CPA problem for $(\mathsf{Enc}, \mathsf{Dec})$ or the tweaked reciprocal CSIDH problem.

- Game_0 is the original experiment where the environment machine interacts with the protocol π and the adversary $\mathcal{A}$ or the simulator $\mathcal{S}_{\mathsf{R}}$ and $\mathcal{F}_{\mathsf{OT}}$. By definition, $\mathsf{Adv}(\mathcal{Z}) = \mathsf{Adv}_0(\mathcal{Z})$.

- Game_1 is the same as Game_0 except that the winning condition is changed. Under the condition that the adversary sends correct string $\mathsf{str}' = \mathsf{str}$, we additionally require the adversary should make a random oracle query of one of $s_1 \star C$ or $s_1 \star C^t$. The game results in a loss if the adversary is able to decrypt without knowing the decryption key, otherwise, the winning condition is not changed. Recall that ciphertexts $c_{1,0} = (s_1 \parallel \mathsf{str}) \oplus k_{1,0}$,

$c_{1,1} = (s_1 \parallel \mathsf{str}) \oplus k_{1,1}$ and keys $k_{1,0}, k_{1,1}$ are drawn uniformly at random from $\{0,1\}^{(\tau+1)\lambda}$. The string str is information-theoretically hidden from $\mathcal{Z}$ without knowing $k_{1,0}, k_{1,1}$. The string $\mathsf{str} \in \{0,1\}^\lambda$ has λ-bit min-entropy. Therefore, we have $\mathsf{Adv}_0(\mathcal{Z}) \leq \mathsf{Adv}_1(\mathcal{Z}) + 1/2^\lambda$.

- Game_2 is the same as Game_1 except that the winning condition is changed. We additionally require the adversary should make a random oracle query of $s_0 \star C$ or $s_0 \star C^t$ to obtain $k_{0,1-\mathsf{i}}$ (for the case extracted $\mathsf{i} = 1$ or 0, resp). The game results in a loss if the environment machine can notice whether the encrypted message of $\mathsf{M}_{1-\mathsf{i}}$ is replaced without knowing the corresponding decryption key $k_{0,1-\mathsf{i}}$. We can transform $\mathcal{Z}$ to be an $\mathsf{IND\text{-}CPA}$ adversary. Concretely, we construct an $\mathsf{IND\text{-}CPA}$ adversary $\mathcal{B}_1$ against $(\mathsf{Enc}, \mathsf{Dec})$ that simulates $\mathcal{S}_\mathsf{R}$ with access to $\mathcal{A}$ for the environment machine. We assume $\mathcal{B}_1$ can extract the input $x = (\mathsf{M}_0, \mathsf{M}_1)$ of the sender which is decided by $\mathcal{Z}$ [2]. The adversary $\mathcal{B}_1$ is identical to the one we defined in 4.4.1. Concretely, $\mathcal{B}_1$ runs as the same as $\mathcal{S}_\mathsf{R}$ except that $\mathcal{B}_1$ generates a random message $\widetilde{\mathsf{M}}$ and sends $(\mathsf{M}_{1-i}, \widetilde{\mathsf{M}})$ to the $\mathsf{IND\text{-}CPA}$ challenger and assigns the ciphertext to $\widetilde{c_{1,1-\mathsf{i}}}$. Then, $\mathcal{B}_1$ sends $\widetilde{c_{1,1-\mathsf{i}}}, c_{1,\mathsf{i}}$ to the receiver (replacing $c_{1,1-\mathsf{i}}$). If $\mathcal{Z}$ returns a bit b, then $\mathcal{B}_1$ returns b in the $\mathsf{IND\text{-}CPA}$ experiment.
  When the secret bit of the $\mathsf{IND\text{-}CPA}$ challenger is b, the simulation of $\mathcal{B}_1$ is identical to the real protocol π with $\mathcal{A}$. When the secret bit of the $\mathsf{IND\text{-}CPA}$ challenger is $1-b$, the simulation of $\mathcal{B}_1$ is identical to $\mathcal{S}_\mathsf{R}$ in the ideal process. Therefore, we have $\mathsf{Adv}_1(\mathcal{Z}) \leq \mathsf{Adv}_2(\mathcal{Z}) + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{(\mathsf{Enc},\mathsf{Dec})}(\mathcal{B}_1)$.

- Game_3 is the same as Game_2 except that the winning condition is changed. We additionally require the adversary should make a random oracle query of $s_0 \star C, s_0 \star C^t$ (up to i) to obtain $k_{0,\mathsf{i}}$. The game results in a loss if the environment machine can notice the encrypted message of $\mathsf{M}_{1-\mathsf{i}}$ is replaced without the decryption key $k_{0,\mathsf{i}}$. We may assume further $\mathcal{Z}$ does not know the corresponding curve. In other words, the extraction of the simulator $\mathcal{S}_\mathsf{R}$ is false (Step 5) since the environment machine can decrypt $c_{1,\widetilde{\mathsf{i}}}$ first but only able to decrypt $c_{0,1-\widetilde{\mathsf{i}}}$. We construct a tweaked reciprocal CSIDH problem adversary $\mathcal{B}_2$ that simulates for $\mathcal{Z}$.

  Without loss of generality, we say $\mathsf{i} = 0$ and the tweaked reciprocal CSIDH problem CSIDH challenge is $(E, s_1 \star E, s_0 \star E)$. $\mathcal{B}_2$ will assign E as the trusted setup curve, $s_1 \star E$ to A_1, and $s_0 \star E$ to A_0 and $\mathcal{B}_2$ commits to C in the experiment which is given from the (corrupted) receiver. Note that $\mathcal{B}_2$ does not have s_1, so it firstly produces dummy ciphertexts $\widetilde{c_{0,0}}, \widetilde{c_{0,1}} \xleftarrow{\$} \{0,1\}^{(\tau+1)\lambda}$. Next, $\mathcal{B}_2$ guesses and extracts $s_1 \star C$ from the queries of $\mathcal{F}_{\mathsf{RO}}(\mathsf{Enc} \parallel \cdot)$ and gives $s_1 \star C$ to the tweaked reciprocal CSIDH problem challenger to obtain s_1. By sampling $\widetilde{\mathsf{str}} \xleftarrow{\$} \{0,1\}^\lambda$, $\mathcal{B}_2$ assigns $\widetilde{c_{0,0}} \oplus (s_1 \parallel \widetilde{\mathsf{str}}), \widetilde{c_{0,1}} \oplus (s_1 \parallel \widetilde{\mathsf{str}})$ to $\mathcal{F}_{\mathsf{RO}}(\mathsf{Enc} \parallel s_1 \star C), \mathcal{F}_{\mathsf{RO}}(\mathsf{Enc} \parallel s_1 \star C^t)$, respectively.
  Again, $\mathcal{B}_2$ guesses and extracts $s_0^{-1} \star C^t$ $(= (s_0 \star C^t)^t)$ from the queries for $\mathcal{F}_{\mathsf{RO}}(\mathsf{Key} \parallel \cdot)$

[2]Note that the simulator $\mathcal{S}_\mathsf{R}$ does not know $x = (\mathsf{M}_0, \mathsf{M}_1)$ but the environment machine does since it decides the input for the sender. Therefore, we can argue that we can extract $x = (\mathsf{M}_0, \mathsf{M}_1)$ from the machine.

corresponding to $k_{0,1}$. Finally, two answers given by $\mathcal{B}_2$ are $s_1 \star C, s_0^{-1} \star C^t$, respectively. A loss of a factor $2 \times \frac{Q^2-Q}{2}$ follows due to guessing. Therefore, we have $\mathsf{Adv}_2(\mathcal{Z}) \leq \mathsf{Adv}_3(\mathcal{Z}) + (Q^2 - Q) \times \mathsf{Adv}^{\mathsf{tReGA}}(\mathcal{B}_2)$.

- Game_4 is the same as Game_3 except that the winning condition is changed. We randomly pick two distinct numbers I_1, I_2 from $[Q]$ at the outset of the game. We additionally require that the game result in a win if the I_1, I_2-th random oracle queries corresponding to $s_0 \star C, s_0 \star C^t$ obtain $k_{0,0}, k_{0,1}$, respectively. Since I_1, I_2 are information theoretically hidden during the execution of the game. The guesses are correct with a chance $2/(Q^2-Q)$. Therefore, we have $\mathsf{Adv}_3(\mathcal{Z}) = \frac{Q^2-Q}{2} \times \mathsf{Adv}_4(\mathcal{Z})$.

We construct an adversary $\mathcal{B}_3$ against the reciprocal CSIDH problem which simulates Game_4 for $\mathcal{Z}$. Say the reciprocal CSIDH public curve is E, $\mathcal{B}_3$ runs as the same as $\mathcal{S}_\mathsf{R}$ except in Step 1, Step 2 and Step 6. In Step 1, $\mathcal{B}_3$ runs as the same as $\mathcal{S}_\mathsf{R}$ except for two special cases with respect to the changes made in Step 6. In Step 2, $\mathcal{B}_3$ uses E as the trusted setup curve. (For simplicity, we skip the randomization process here.) Upon receiving C from the corrupted receiver, $\mathcal{B}_3$ commits to C in the reciprocal CSIDH experiment. Say the challenger returns $s_0 \star E$ as the challenge. In Step 6, $\mathcal{B}_3$ assigns $s_0 \star E$ to A_0. Since $\mathcal{B}_3$ does not know $s_0 \star C$ and $s_0 \star C^t$, it produces two hash values $k_{0,0}, k_{0,1}$ uniformly at random from $\mathcal{K}$. From the assumptions made in $\mathsf{Game}_2, \mathsf{Game}_3$, the queries of $s_0 \star C, s_0 \star C^t$ for $\mathcal{F}_\mathsf{RO}(\mathsf{Key} \parallel \cdot)$ will be made. From the assumption made in Game_4, $\mathcal{B}_3$ can correctly assign $k_{0,0}, k_{0,1}$ to $\mathcal{F}_\mathsf{RO}(\mathsf{Key} \parallel s_0 \star C), \mathcal{F}_\mathsf{RO}(\mathsf{Key} \parallel s_0 \star C^t)$ respectively. The simulation is therefore indistinguishable. Finally, $\mathcal{B}_3$ returns $s_0 \star C$ and $s_0^{-1} \star C = (s_0 \star C^t)^t$ in the reciprocal CSIDH experiment. Therefore, we have $\mathsf{Adv}_4(\mathcal{Z}) \leq \mathsf{Adv}^{\mathsf{rCDH}}(\mathcal{B}_3)$.

Hence, in this case we have

$$\mathsf{Adv}(\mathcal{Z}) \leq \frac{1}{2^\lambda} + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}(\mathcal{B}_1) + (Q^2 - Q) \times \mathsf{Adv}^{\mathsf{tReGA}}(\mathcal{B}_2) + \frac{Q^2 - Q}{2} \times \mathsf{Adv}^{\mathsf{rCDH}}(\mathcal{B}_3).$$

(Corrupted Sender and Honest Receiver) Let $\mathcal{A}$ be a malicious adversary controlling the sender. In order to emulate the adversary, the simulator needs to extract the input of the adversary, and send it to the trusted party in the ideal execution. The input here is the message pair which the honest receiver will read. Say the environment machine $\mathcal{Z}$ generates input $(x = (\mathsf{M}_0, \mathsf{M}_1), y = i, z)$ and gives x to the simulator/adversary on auxiliary input z. The simulator $\mathcal{S}_\mathsf{S}$ with input (x, z) proceeds as follows:

1. The simulator $\mathcal{S}_\mathsf{S}$ simulates a random oracle $\mathcal{F}_\mathsf{RO}$ in an on-the-fly manner. Recall that we have two prefixes $\mathcal{F}_\mathsf{RO}(\mathsf{Key} \parallel \cdot)$ and $\mathcal{F}_\mathsf{RO}(\mathsf{Enc} \parallel \cdot)$. For $\mathcal{F}_\mathsf{RO}(\mathsf{Key} \parallel \cdot)$, it keeps a list L, which is empty initially, over $\{\mathsf{Key}\} \times \mathcal{E} \times \mathcal{K}$ that records each past query. Upon receiving a random oracle query of $(\mathsf{Key}, E') \in \{\mathsf{Key}\} \times \mathcal{E}$, the simulator checks whether $(\mathsf{Key}, E', k') \in L$ for some $k' \in \mathcal{K}$. If not, generate $k' \xleftarrow{\$} \mathcal{K}$ and add (Key, E', k') to L. Finally, $\mathcal{S}_\mathsf{S}$ returns k' to the random oracle query of (Key, E'). $\mathcal{S}_\mathsf{S}$ does the same process for $\mathcal{F}_\mathsf{RO}(\mathsf{Enc} \parallel \cdot)$ where the list is over $\{\mathsf{Enc}\} \times \mathcal{E} \times \{0,1\}^{(\tau+1)\lambda}$.

2. Generate the public curve $E = t \star E_0$ by sampling $t \xleftarrow{\$} G$ to simulate $\mathcal{F}_{TSC}$. Invoke the adversary $\mathcal{A}$ on auxiliary input z with the input x and E. Keep t as the trapdoor secret. $\mathcal{S}_{\mathsf{S}}$ simulates the communication with $\mathcal{Z}$ by forwarding the received message from $\mathcal{Z}$ to $\mathcal{A}$ and forwarding the response of $\mathcal{A}$ back to $\mathcal{Z}$. If $\mathcal{A}$ aborts or halts, then $\mathcal{S}_{\mathsf{S}}$ does the same.
3. Generate $r \xleftarrow{\$} G$, and compute $C = r \star E$. Send C to the adversary, and act as an honest receiver with the input $i = 0$ throughout the remaining execution. (Note that the simulator does not know the input of the receiver here.)
4. If the adversary aborts, then send **abort** to $\mathcal{F}_{\mathsf{OT}}$ and finish the session. Otherwise, assume the execution is not aborted. Say it receives $(A_0, c_{0,0}, c_{0,1})$ from the adversary. Compute $k_0 = H(r \star A_0)$, $k_1 = H(r^{-1}t^{-2} \star A_0)$, and $m_j = \mathsf{Dec}_{k_j}(c_{0,j})$ for $j \in \{0, 1\}$.
5. Send (m_0, m_1) to the trusted third party in the ideal execution, output whatever the adversary outputs to complete the simulation. (Note that $(\mathsf{M}_0, \mathsf{M}_1)$ and (m_0, m_1) are not necessarily the same since the adversary can change the original input.)

Claim $\{HYBRID^{\mathcal{F}_{RO},\mathcal{F}_{TSC}}_{\pi,\mathcal{A}(z),1}(x, y)\}_{x,y,z} = \{IDEAL_{\mathcal{F}_{\mathsf{OT}},\mathcal{S}(z),1}(x, y)\}_{x,y,z}$. In contrast to the real execution of π, there are two differences here. Firstly, the simulator possesses the trapdoor t of the public curve. The process is identical to $\mathcal{F}_{TSC}$, and the simulator acts as an honest receiver throughout the process. Hence, this difference is unnoticeable to the adversary.

The other difference is the receiver the simulator plays always uses input $\mathsf{i} = 0$. Concretely, we have to show that the distributions of outputs given by honest receivers are indistinguishable with respect to the secret input $\mathsf{i} \in \{0, 1\}$.

The first message. Firstly, the distribution of the first message (C) in the protocol as $\mathsf{i} = 0$ is the same as that generated as $\mathsf{i} = 1$ due to the uniformity. By assuming the existence of the uniformly sampling over G, the distributions are identical.

The second message. Claim that the second message being str' or $\mathbf{abort}_{\mathsf{R}}$ is independent of i. Concretely, if an honest receiver with secret input i sends C to the receiver, a corrupted sender cannot produce $(A_1, c_{1,0}, c_{1,1})$ such that the receiver's output differ with respect to i.

Firstly, we show that if an honest receiver does not output $\mathbf{abort}_{\mathsf{R}}$, then the honest receiver with $\mathsf{i} = 1$ will not output $\mathbf{abort}_{\mathsf{R}}$ here as well. Assume the receiver with $\mathsf{i} = 0$ does not abort, then we have the following equations:

$$\begin{aligned} c_{1,0} =& (x_0 \parallel \mathsf{str}_0) \oplus H_{\mathsf{Enc}}(x_0 \star C) \\ c_{1,1} =& (x_0 \parallel \mathsf{str}_0) \oplus H_{\mathsf{Enc}}(x_0 \star C^t) \\ A_1 =& x_0 \star E, \end{aligned}$$

where $x_0 \in G$ and distinct $\mathsf{str}_0 \in \{0, 1\}^\lambda$.

The receiver with $\mathsf{i} = 1$ sends the same C to the sender but using $r_1 \in G$ such that $C = (r_1 \star E)^t$. He computes $H_{\mathsf{Enc}}(r_1 \star A_1)$ to decrypt $c_{1,1}$. Since $r_1 \star A_1 = r_1 x_0 \star E = x_0 \star (r_1 \star E) = x_0 \star C^t$, the receiver obtains the same message $(x_0 \parallel \mathsf{str}_0)$ after decryption.

Secondly, we show that if the receiver with $\mathsf{i} = 1$ does not output $\mathbf{abort}_{\mathsf{R}}$, then the receiver with $\mathsf{i} = 0$ will not output $\mathbf{abort}_{\mathsf{R}}$ as well. Assume the receiver with $\mathsf{i} = 1$ does not abort, then we have the following equations:

$$\begin{aligned} c_{1,0} =& (x_1 \parallel \mathsf{str}_1) \oplus H_{\mathsf{Enc}}(x_1 \star C) \\ c_{1,1} =& (x_1 \parallel \mathsf{str}_1) \oplus H_{\mathsf{Enc}}(x_1 \star C_1^t) \\ A_1 =& x_1 \star E, \end{aligned}$$

where $x_1 \in G$ and distinct $\mathsf{str}_1 \in \{0,1\}^\lambda$.

The receiver with $\mathsf{i} = 0$ sends the same C to the sender but using $r_0 \in G$ such that $C = r_0 \star E$. He computes $H_{\mathsf{Enc}}(r_0 \star A_1)$ to decrypt $c_{1,0}$. Since $r_0 \star A_1 = r_0 x_1 \star E = x_1 \star (r_0 \star E) = x_0 \star C$, the receiver obtains the same message $(x_1 \parallel \mathsf{str}_1)$ after decryption.

Thirdly, We show that if both receivers do not output $\mathbf{abort}_{\mathsf{R}}$, the string given to the sender will not vary with i. This immediately follows from the proof right above. If the receiver with secret input i obtains the message $(x_{\mathsf{i}} \parallel \mathsf{str}_{\mathsf{i}})$, then the receiver with secret input $1 - \mathsf{i}$ will obtains the same message $(x_{\mathsf{i}} \parallel \mathsf{str}_{\mathsf{i}})$.

The output of the receiver. It suffices to show the correctness of the extraction in Step 4. If an honest receiver sends C to the sender with the input $\mathsf{i} = 0$, then the decryption key is $k_0 = H_{\mathsf{Key}}(r \star A_0)$. The message the receiver will obtain is $\mathsf{Dec}_{k_0}(c_{0,0}) = m_0$. Besides, if an honest receiver sends C to the sender with the input $\mathsf{i} = 1$, then the private ideal is equivalent to $r^{-1}t^{-2}$ since $(r^{-1}t^{-2} \star E)^t = (r^{-1}t^{-1} \star E_0)^t = (r^{-1} \star E^t)^t = (r \star E) = C$. Hence, the receiver will decrypt $c_{0,1}$ with $H_{\mathsf{Key}}(r^{-1}t^{-2} \star A_0)$. Due to $k_1 = H_{\mathsf{Key}}((tr \star (t^{-1} \star A_0)^t)^t) = H_{\mathsf{Key}}((tr)^{-1}t^{-1} \star A_0) = H_{\mathsf{Key}}(r^{-1}t^{-2} \star A_0)$, the receiver will therefore get the message $m_1 = \mathsf{Dec}_{k_1}(c_{0,1})$. That is, the simulator correctly extracts the input of the adversary. Hence, the real execution is perfectly indistinguishable from the ideal execution. □

4.5 Comparison

4.5.1 Efficiency

Table 4.1 illustrates a comparison between our oblivious transfer protocols with [DOPS20, Vit19, AFMP20] in terms of efficiency, including the number of curves in the domain parameters or generated by a trusted party, the number of curves in the public keys for the sender and the receiver, the total number of isogeny computations for the sender and the receiver, and the number of rounds, respectively. Among the isogeny-based OTs, our 2-round OT proposal is the most efficient with respect to every criteria against semi-honest adversaries. It takes two additional rounds and a constant number of isogeny computations for each participant to achieve UC-secure against static malicious adversaries.

Proposal	DP	PK_S	PK_R	$\# \ Iso_S$	$\# \ Iso_R$	$\# \ rounds$	Others
[DOPS20] I	2	1	1	3	2	2	
[DOPS20] II	1	3	1	5	2	3	
[Vit19]	1	2	1	4	2	3	Insecure in CSIDH
[AFMP20] I	4	2λ	2	4λ	$\lambda + 2$	2	Group-action-based
[AFMP20] II	1	2λ	5	4λ	$\lambda + 5$	2	Single Bit Transfer
This work (Fig. 4.3)	1	1	1	3	2	2	CSIDH-based
This work (Fig. 4.4)	1	1	1	6	5	4	CSIDH-based

Table 4.1: Comparison between isogeny-based OTs on efficiency where λ is the security parameter. We give the costs for both our 2-round protocol from Figure 4.3 and the full construction from Figure 4.4. DP, PK_S, PK_R counts the number of curves needed to be stored in the domain parameter and the communications by the sender and receiver respectively.

The two frameworks of [DOPS20] includes DH, SIDH, and CSIDH settings. The first construction is a two-message oblivious transfer and requires one more curve in the trusted setup phase.

The paper [Vit19] showed a construction based on exponentiation-only Diffie-Hellman. The construction can fit in the DH, SIDH, and CISDH settings. The SIDH instance is not secure now. Also, as stated in their work (in the inproceeding version), it will be totally insecure in the CSIDH setting against a malicious receiver. Specifically, their two-inverse problem is given curves $(E, a \star E, b \star E)$ to find some curve tuple $(X, a^{-1} \star X, b^{-1} \star X)$ where X is isogenous to E. This can be done in the CSIDH setting by taking quadratic twists of $(E, a \star E, b \star E)$.

In [AFMP20], the UC-secure construction is based on the decisional group action problem (the decisional CSIDH problem for instance). It requires the number of isogeny computation linear in λ and it only transfers a bit each time.

4.5.2 Security

	Adversary Model	Security Definition	Model
[DOPS20] I	Semi-honest	UC-realize	ROM+TSC
[DOPS20] II	Semi-honest	UC-realize	ROM+TSC
[Vit19]	Semi-honest	UC-realize	ROM
[AFMP20]	Malicious	UC-realize	TSC
This work (Fig. 4.3)	Semi-honest	UC-realize	ROM+TSC
This work (Fig. 4.4)	Malicious	UC-realize	ROM+TSC

Table 4.2: Comparison between previous isogeny OTs and our constructions. The models include the random oracle model (ROM) and trusted setup curves (TSC).

In [DOPS20, Vit19], the schemes, instantiated using CSIDH, are both universally composable secure in the semi-honest model.

4.6 Discussion

4.6.1 Results of Using Generic Transforms

In Chapter 7 of [DOPS20], the authors mention that their construction can be compiled into a UC-secure scheme in the malicious model using the transformation from [DGH+20]. However, this requires invoking the underlying OT λ times. One can also apply the transformation from [DGH+20] to our construction presented in Fig. 4.3, which results in a similar outcome of requiring $O(\lambda)$ isogeny computations. As the implications of the transformation in [DGH+20] are easy to overlook within the isogeny community, we provide a brief summary below and the results of the application in Tab. 4.3.

[DGH+20] comprises a series of transformations (6 in total) that convert a weak OT instance (referred to as an elementary OT) into a UC-secure OT. The second to last transformation turns a 2-round sender UC-secure OT (with receiver's indistinguishability security) into a binary-challenge zero-knowledge proof (ZKP) in a common reference string (CRS) model, necessitating λ repetitions for sufficient soundness. Subsequently, the ZKP is utilized in the last transformation to turn the sender UC-secure OT into a fully UC-secure OT. Importantly, the resulting scheme remains two-round (i.e., round-optimal). Our two-round construction is sender UC-secure in the TCS model using the proof of the corrupted-sender-and-honest-receiver case in Thm. 4.4.2, which does not require the additional rounds as in Fig. 4.4 to achieve. The receiver's indistinguishability security holds due to the uniform sampling over the group G. Thus, we can apply the transformations from Sections 8, and 9 of [DGH+20] to convert both our 2-round construction and also the first construction of [DOPS20] into UC-secure schemes. The summary of the performance is given in Tab. 4.3.

	$\# \; Iso_S$	$\# \; Iso_R$	$\# \; rounds$	Assumption	Model
[DOPS20] I + [DGH+20]	$O(3\lambda)$	$O(2\lambda)$	2	GAIP	ROM+TCS
Fig. 4.3 + [DGH+20]	$O(3\lambda)$	$O(3\lambda)$	2	GAIP	ROM+TCS

Table 4.3: Immediate results by applying the transform [DGH+20] to the two-round sender UC-secure OTs.

4.6.2 Further Directions

Several directions and open problems in isogeny-based OT research are worth investigating.

First, some relaxed notions of OT have not been considered in this thesis. For example, semantic security in [Vit19] and statistical sender privacy in [AFMP20]. These notions might be sufficient for certain applications. Additionally, we believe there is potential for efficiency improvements in the latter construction, which currently requires $O(\lambda)$ isogeny computations and transfers one bit at a time.

Another interesting direction is the study of a primitive called "OT extension." Combining public key and private key encryptions can be more cost-effective than using public key encryption alone. Similarly, an OT extension construction that uses fewer OT executions along with other inexpensive primitives can be more efficient than using OTs alone. [KOS15] presents an efficient UC-secure OT extension construction, and [CSW20] demonstrates that the underlying OT component of [KOS15] can be relaxed. Specifically, the base-OT in their extension construction only needs to be UC-secure for the sender and indistinguishable against a malicious receiver. This relaxed notion provides more flexibility when designing an OT scheme, and the OT-extension scheme achieves the same effect as using UC-secure OT as a component in turn. Conceptually, this enables a simpler construction, such as a Chou-Orlandi type construction (as seen in Fig. 4.2), which by itself may not be a UC-secure OT, but serves as a valid component for the UC-secure OT extension using [KOS15, CSW20]. Recent work [BMM$^+$22] improves efficiency in this direction, resulting in a scheme that requires only four isogeny computations.

Lastly, achieving an efficient (with a constant number of isogeny computations) and UC-secure round-optimal or adaptively UC-secure OT remains an open problem. In fact, this thesis only considers static adversaries, where the party to be corrupted is chosen before the execution begins. In the adaptive model, the adversary can corrupt parties after the execution has started. We have not identified any efficient adaptively secure or round-optimal isogeny-based constructions in the literature, making this an area worth exploring.

While the author is writing this thesis, a work by Orsini and Zanotto [OZ23] was submitted to eprint. The work claims to provide two UC-secure isogeny-based OTs in an abstract model called the explicit isogeny model. One is a 3-round construction in the Random Oracle Model (ROM) with six isogeny computations. The other is a round-optimal isogeny-based OT that improves upon our construction with only seven isogeny computations in total. The method appears to be plausible, as in an abstract model, group actions and taking twists are the only valid operations. Consequently, this constrains the behavior of a malicious adversary and reduces the overhead needed to defend against such adversaries. We look forward to further investigation to verify these claims and advancements.

Chapter 5

Ring and Group Signatures

This chapter presents the work carried out in [BDK$^+$22], which the author of the thesis co-authored. The author's contributions include proposing the project and the idea of constructing the tightly secure variant. To make the content more accessible, this chapter has been adapted by removing the lattice instances and the use of rejection sampling without compromising the essence of the original work.

Abstract. This chapter introduces a novel approach to construct a group signature and an accountable ring signature from isogenies, which provides strong security guarantees and a competitive signature size. Our construction achieves the first logarithmic group signature from isogenies and the first post-quantum accountable ring signature. Additionally, we provide a tightly secure variant for our construction, which is a less explored feature in the post-quantum group/ring signature literature.

5.1 Introduction

Group signature schemes, first proposed by Chaum and van Heyst [Cv91], offer a way for authorized members of a group to sign documents on behalf of the group while keeping their individual identities anonymous. This property makes group signatures attractive for various applications where the anonymity of the signer is important. The full anonymity even ensures the anonymity remains even the signing key is accidentally exposed. In case the situation demands, a designated entity known as the group manager (or sometimes the tracing authority) can track the signature to its originator, thereby ensuring that the group members are held accountable for their signatures. Over the last three decades, group signatures have been an active area of academic research and have also gained practical attention due to the real-world deployment of variants such as directed anonymous attestation (DAA) [BCC04] and enhanced privacy ID (EPID) [BL07].

A variety of versatile constructions of *efficient* group signatures exist for *classical* assumptions, e.g., [BBS04, DP06, Gro07, FI06, BCN$^+$10, LPY15, LMPY16, DS18, BHSB19, CS20]. When evaluating the efficiency of a group signature, one of the quintessential metrics is the signature size. In this chapter, we require the signature size to be smaller than $c \cdot \log N$ bits, where N is the group size (the number of members) and c is an explicit small polynomial in the security parameter. Bellare, Micciancio, and Warinschi's seminal work [BMW03] provided a generic construction of a group signature with a signature size of $O(1)$ from any signature scheme, IND-CCA public-key encryption scheme, and general non-interactive zero-knowledge (NIZK) proof system. However, this only yields an asymptotic feasibility result. Therefore, one of the main focuses of subsequent works, including ours, has been to construct a group signature that is practically efficient.

In contrast to the classical setting, creating efficient group signatures from any *post-quantum* assumptions has proven challenging. Since the introduction of the first lattice-based construction by Gordon, Katz, and Vaikuntanathan [GKV10], there has been an extensive line of subsequent works on lattice-based (and one code-based) group signatures, including but not limited to [LLLS13, ELL$^+$15, LLNW16, LNWX18, KY19]. However, these results remained purely asymptotic. Recently, efficient lattice-based group signatures have emerged [BCN18, dLS18, EZS$^+$19, ESZ22]. In [ESZ22], Esgin et al. report a signature size of 12KB and 19KB for a group size (the number of the members) of $N = 2^6$ and 2^{10}, respectively—a considerable improvement over prior constructions by several orders of magnitude[1]. The recent advancements in lattice-based NIZK proof systems for useful languages, as evidenced by the numerous works [YAZ$^+$19, BLS19, ESLL19, ALS20, ENS20, LNS20, LNS21], have led to significant improvements in the efficiency of lattice-based group signatures. However, these improvements cannot transfer to the isogeny setting due to the particularly structured lattices.

On top of that, constructing efficient group signatures from isogenies remains essentially challenging using current techniques. In general, most group signature constructions follow the *sign-encrypt-proof* paradigm [BMW03]. The *high-level* idea is fairly simple:

1. The signer produces a signature of a message by signing via his signing key.
2. The signer produces a ciphertext by encrypting his verification key via the manager's PKE public key.
3. The signer provides a proof for the "signature relation" R_{sig} relations:
   - The signature is signed on the message using the signing key corresponding to a verification key from the ring (a set containing several verification keys of the group).
   - The ciphertext encrypts a verification key from the ring via the manager's PKE public key.
   - The two verification keys described above are the same one.

Typically, this approach requires $\mathsf{IND\text{-}CCA}$ secure PKE [DP06, Gro07, FI06, LPY15, LMPY16, dLS18] as a building block and an efficient NIZK that proves validity of the ciphertext to have

[1]The signature size of [ESZ22] is in $\log^t N$ for a small constant $t > 1$ rather than simply by $\log N$.

full-anonymity for technical reasons[2]. In the isogeny setting, although we know how to construct an IND-CCA secure PKE based on the Fujisaki-Okamoto transform [FO99], it seems quite difficult to offer the encryption verifiability due to the use of a hash function in the construction.

The difficulties raise the central question of our work:

Can we construct an efficient secure group signature from isogenies?

In addition, as we discuss in more detail later, we notice that all works regarding efficient post-quantum group signatures [BCN18, KKW18, dLS18, EZS+19, ESZ22] do not satisfy the ideal security properties (which are by now considered standard) formalized by Bootle et al. [BCC+16a]. Thus, we are also interested in the following question:

Can we construct an efficient group signatures from isogenies satisfying the ideal security properties formalized by Bootle et al. [BCC+16a]?

To address these questions, in this work we focus on *accountable ring signatures* [XY04][3]. An accountable ring signature offers the flexibility of choosing the group of users when creating a signature (like a ring signature [RST01]). At the same time, it enforces accountability by mandating the inclusion of one of the openers in the group, much like a group signature. Despite limited research in this area [XY04, BCC+15, LZCS16, KP17, EZS+19], we believe that accountable ring signatures are as relevant and interesting as group and ring signatures, and can bridge the gap between the two primitives. As shown by Bootle et al. [BCC+15], accountable ring signatures imply group and ring signatures by naturally limiting or downgrading their functionality. Thus, an efficient post-quantum solution to an accountable ring signature also implies solutions for *both* secure (dynamic) group signatures [BSZ05] and ring signatures, making it an attractive target to focus on.

Finally, we are also interested in *tightly*-secure constructions, which have received less attention in prior efficient post-quantum group and ring signature schemes. Tightly-secure constructions have a small reduction loss in the security proof, which means that the adversary's advantage in breaking the security property of the signature scheme is almost the same as its advantage in solving the underlying hard problem. In contrast, most prior schemes are in the random oracle model and have a very loose reduction loss to the best of our knowledge. In typical security proofs, given an adversary with advantage ϵ that breaks some security property of the group signature, we can only construct an adversary with advantage at most $(N^2Q)^{-1} \cdot \epsilon^2$ against the underlying hard problem, where Q is the number of random oracle queries and N is the number of users in the system. If we aim for 128-bit security (i.e., $\epsilon = 2^{-128}$), and set for example $(N, Q) = (2^{10}, 2^{50})$, then we need at least 326-bits of security for the hard problem. When aiming for a provably-secure construction, the parameters must be set much larger to compensate for this significant reduction loss, which then leads to a less efficient scheme. As a summary, we aim to address the following research questions:

[2]To use the decryption oracle in IND-CCA experiment to answer the opening queries in the anonymity experiment.

[3]Note that the original security guarantees in [XY04] is much weaker than those formalized in [BCC+16a].

Can we construct an efficient and tightly-secure accountable ring signature from isogenies realizing ideal security properties formalized in [BCC+16a]?

5.2 Preliminaries

5.2.1 Accountable Ring Signatures

We provide the definition of accountable ring signatures (ARSs), following the formalization introduced by Bootle et al. [BCC+15].

Definition 5.2.1 (Accountable Ring Signature)**.** *An accountable ring signature* Π_{ARS} *consists of PPT algorithms* $(\mathsf{Setup}, \mathsf{MKGen}, \mathsf{UKGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Open}, \mathsf{Judge})$ *defined as follows:*

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: *On input a security parameter* 1^λ*, it returns a public parameter* pp *(sometimes implicitly) used by the scheme. We assume* pp *defines the openers' public-key space* $\mathcal{K}_{\mathsf{mpk}}$ *and users' verification-key space* $\mathcal{K}_{\mathsf{vk}}$*, with efficient algorithms to decide membership.*

$\mathsf{MKGen}(\mathsf{pp}) \to (\mathsf{mpk}, \mathsf{msk})$: *On input a public parameter* pp*, it outputs a public and a secret key* $(\mathsf{mpk}, \mathsf{msk})$ *for an* opener*.*

$\mathsf{UKGen}(\mathsf{pp}) \to (\mathsf{vk}, \mathsf{sk})$: *On input a public parameter* pp*, it outputs a pair of verification and signing keys* $(\mathsf{vk}, \mathsf{sk})$ *for a* user*.*

$\mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, \mathsf{R}, \mathsf{M}) \to \sigma$: *On input an opener's public key* mpk*, a signing key* sk*, a list of verification keys, i.e., a* ring*,* $\mathsf{R} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_N)$*, and a message* M*, it outputs a signature* σ*.*

$\mathsf{Verify}(\mathsf{mpk}, \mathsf{R}, \mathsf{M}, \sigma) \to \top/\bot$: *On input an opener's public key* mpk*, a ring* $\mathsf{R} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_N)$*, a message* M*, and a signature* σ*, it (deterministically) outputs either* $\top$ *(accept) or* $\bot$ *(reject).*

$\mathsf{Open}(\mathsf{msk}, \mathsf{R}, \mathsf{M}, \sigma) \to (\mathsf{vk}, \pi)/\bot$: *On input an opener's secret key* msk*, a ring* $\mathsf{R} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_N)$*, a message* M*, a signature* σ*, it (deterministically) outputs either a pair of verification key* vk *and a proof* π *that the owner of* vk *produced the signature, or* $\bot$*.*

$\mathsf{Judge}(\mathsf{mpk}, \mathsf{R}, \mathsf{vk}, \mathsf{M}, \sigma, \pi) \to \top/\bot$: *On input an opener's public key* mpk*, a ring* $\mathsf{R} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_N)$*, a verification key* vk*, a message* M*, a signature* σ*, and a proof* π*, it (deterministically) outputs either* $\top$ *(accept) or* $\bot$ *(reject). We assume without loss of generality that* $\mathsf{Judge}(\mathsf{mpk}, \mathsf{R}, \mathsf{vk}, \mathsf{M}, \sigma, \pi)$ *outputs* $\bot$ *if* $\mathsf{Verify}(\mathsf{mpk}, \mathsf{R}, \mathsf{M}, \sigma)$ *outputs* $\bot$*.*

An accountable ring signature is required to satisfy the following properties: correctness, anonymity, traceability, unforgeability, and tracing soundness.

First, we require correctness to hold even if the ring contains maliciously-generated user keys or the signature has been produced for a maliciously-generated opener key. Note that the correctness guarantee for the open and judge algorithms are defined implicitly in the subsequent security definitions.

Definition 5.2.2 (Correctness)**.** *An accountable ring signature* Π_{ARS} *is* correct *if, for all* $\lambda \in \mathbb{N}$*, any PPT adversary* $\mathcal{A}$ *has at most a negligible advantage in* λ *in the following game played against a challenger.*

(i) The challenger runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *and generates a user key* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{UKGen}(\mathsf{pp})$*. It then provides* $(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$ *to* $\mathcal{A}$*.*

(ii) $\mathcal{A}$ *outputs an opener's public key, a ring, and a message tuple* $(\mathsf{mpk}, \mathsf{R}, \mathsf{M})$ *to the challenger.*

(iii) The challenger runs $\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, \mathsf{R}, \mathsf{M})$*. We say* $\mathcal{A}$ *wins if*

- $\mathsf{mpk} \in \mathcal{K}_{\mathsf{mpk}}$, $\mathsf{R} \subseteq \mathcal{K}_{\mathsf{vk}}$, *and* $\mathsf{vk} \in \mathsf{R}$,
- $\mathsf{Verify}(\mathsf{mpk}, \mathsf{R}, \mathsf{M}, \sigma) = \bot$.

The advantage of $\mathcal{A}$ *is defined as* $\mathsf{Adv}^{\mathsf{Correct}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$.

Anonymity requires that a signature does not leak any information on who signed it. We consider the standard type of anonymity notion where the adversary gets to choose the signing key used to generate the signature. Moreover, we allow the adversary to make (non-trivial) opening queries that reveal who signed the messages. This notion is often called *full* (*CCA*) anonymity [BMW03, BCC+16a] to differentiate between weaker notions of anonymity such as *selfless* anonymity that restricts the adversary from exposing the signing key used to sign the signature or *CPA* anonymity where the adversary is restricted from querying the open oracle.

Definition 5.2.3 (Anonymity)**.** *An accountable ring signature* Π_{ARS} *is (CCA)* anonymous (against full key exposure) *if, for all* $\lambda \in \mathbb{N}$*, any PPT adversary* $\mathcal{A}$ *has at most a negligible advantage in the following game played against a challenger.*

(i) The challenger runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *and generates an opener key* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{MKGen}(\mathsf{pp})$*. It also prepares an empty list* $\mathsf{Q}_{\mathtt{sign}}$ *and samples a random bit* $b \leftarrow \{0, 1\}$*.*

(ii) The challenger provides $(\mathsf{pp}, \mathsf{mpk})$ *to* $\mathcal{A}$*.*

(iii) $\mathcal{A}$ *can make signing and opening queries an arbitrary polynomial number of times:*

- $(\mathtt{sign}, \mathsf{R}, \mathsf{M}, \mathsf{sk}_0, \mathsf{sk}_1)$*: The challenger runs* $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_i, \mathsf{R}, \mathsf{M})$ *for* $i \in \{0, 1\}$ *and returns* $\bot$ *if* $\mathsf{Verify}(\mathsf{mpk}, \mathsf{R}, \mathsf{M}, \sigma_i) = \bot$ *for either of* $i \in \{0, 1\}$*. Otherwise, it updates* $\mathsf{Q}_{\mathtt{sign}} \leftarrow \mathsf{Q}_{\mathtt{sign}} \cup \{(\mathsf{R}, \mathsf{M}, \sigma_b)\}$ *and returns* σ_b*.*
- $(\mathsf{open}, \mathsf{R}, \mathsf{M}, \sigma)$*: The challenger returns* $\bot$ *if* $(\mathsf{R}, \mathsf{M}, \sigma) \in \mathsf{Q}_{\mathtt{sign}}$*. Otherwise, it returns* $\mathsf{Open}(\mathsf{msk}, \mathsf{R}, \mathsf{M}, \sigma)$*.*

(iv) $\mathcal{A}$ *outputs a guess* b^**. We say* $\mathcal{A}$ *wins if* $b^* = b$*.*

The advantage of $\mathcal{A}$ *is defined as* $\mathsf{Adv}^{\mathsf{Anon}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) = |\Pr[\mathcal{A} \text{ wins}] - 1/2|$.

Unforgeability considers two types of forgeries. The first captures the natural notion of unforgeability where an adversary cannot forge a signature for a ring of honest users, i.e., a ring of users for which it does not know any of the corresponding secret keys. The second captures the fact that an adversary cannot accuse an honest user of producing a signature even if the ring contains malicious users and the opener is malicious.

Definition 5.2.4 (Unforgeability)**.** *An accountable ring signature scheme* Π_{ARS} *is* unforgeable (with respect to insider corruption) *if, for all* $\lambda \in \mathbb{N}$*, any PPT adversary* $\mathcal{A}$ *has at most negligible advantage in the following game played against a challenger.*

(i) The challenger runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *and initializes an empty keyed dictionary* $\mathsf{D}_{\mathtt{UKey}}[\cdot]$ *and three empty sets* $\mathsf{Q}_{\mathtt{UKey}}$*,* $\mathsf{Q}_{\mathtt{sign}}$ *and* $\mathsf{Q}_{\mathtt{cor}}$*. It provides* pp *to* $\mathcal{A}$*.*

(ii) $\mathcal{A}$ *can make user key generation, signing, and corruption queries an arbitrary polynomial number of times:*

- $(\mathtt{ukeygen})$*: The challenger runs* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{UKGen}(\mathsf{pp})$*. If* $\mathsf{D}_{\mathtt{UKey}}[\mathsf{vk}] \neq \bot$*, then it returns* $\bot$*. Otherwise, it updates* $\mathsf{D}_{\mathtt{UKey}}[\mathsf{vk}] = \mathsf{sk}$ *and* $\mathsf{Q}_{\mathtt{UKey}} \leftarrow \mathsf{Q}_{\mathtt{UKey}} \cup \{\mathsf{vk}\}$*, and returns* vk*.*
- $(\mathtt{sign}, \mathsf{mpk}, \mathsf{vk}, \mathsf{R}, \mathsf{M})$*: The challenger returns* $\bot$ *if* $\mathsf{vk} \notin \mathsf{Q}_{\mathtt{UKey}} \cap \mathsf{R}$*. Otherwise, it runs* $\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{D}_{\mathtt{UKey}}[\mathsf{vk}], \mathsf{R}, \mathsf{M})$*. The challenger updates* $\mathsf{Q}_{\mathtt{sign}} \leftarrow \mathsf{Q}_{\mathtt{sign}} \cup \{(\mathsf{mpk}, \mathsf{vk}, \mathsf{R}, \mathsf{M}, \sigma)\}$ *and returns* σ*.*
- $(\mathtt{corrupt}, \mathsf{vk})$*: The challenger returns* $\bot$ *if* $\mathsf{vk} \notin \mathsf{Q}_{\mathtt{UKey}}$*. Otherwise, it updates* $\mathsf{Q}_{\mathtt{cor}} \leftarrow \mathsf{Q}_{\mathtt{cor}} \cup \{\mathsf{vk}\}$ *and returns* $\mathsf{D}_{\mathtt{UKey}}[\mathsf{vk}]$*.*

(iv) $\mathcal{A}$ *outputs* $(\mathsf{mpk}, \mathsf{vk}, \mathsf{R}, \mathsf{M}, \sigma, \pi)$*. We say* $\mathcal{A}$ *wins if*

- $(\mathsf{mpk}, \star, \mathsf{R}, \mathsf{M}, \sigma) \notin \mathsf{Q}_{\mathtt{sign}}$*,* $\mathsf{R} \subseteq \mathsf{Q}_{\mathtt{UKey}} \backslash \mathsf{Q}_{\mathtt{cor}}$*,*
- $\mathsf{Verify}(\mathsf{mpk}, \mathsf{R}, \mathsf{M}, \sigma) = \top$*,*

or

- $(\mathsf{mpk}, \mathsf{vk}, \mathsf{R}, \mathsf{M}, \sigma) \notin \mathsf{Q}_{\mathtt{sign}}$*,* $\mathsf{vk} \in \mathsf{Q}_{\mathtt{UKey}} \backslash \mathsf{Q}_{\mathtt{cor}}$*,*
- $\mathsf{Judge}(\mathsf{mpk}, \mathsf{R}, \mathsf{vk}, \mathsf{M}, \sigma, \pi) = \top$*.*

The advantage of $\mathcal{A}$ *is defined as* $\mathsf{Adv}^{\mathsf{Unf}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$*.*

Traceability requires that any opener key pair $(\mathsf{mpk}, \mathsf{msk})$ in the range of the opener key-generation algorithm can open a valid signature σ to some user vk along with a proof valid π. This ensures that any opener can trace the user and produce a proof for its decision. Below, rather than assuming an efficient algorithm that checks set membership $(\mathsf{mpk}, \mathsf{msk}) \in \mathsf{MKGen}(\mathsf{pp})$, we simply ask the adversary to output the randomness used to generate $(\mathsf{mpk}, \mathsf{msk})$. Note that this definition contains the prior definitions where mpk was assumed to be uniquely defined and efficiently computable from msk [BCC+15].

Definition 5.2.5 (Traceability). *An accountable ring signature scheme* Π_{ARS} *is* traceable *if, for all* $\lambda \in \mathbb{N}$, *any PPT adversary* $\mathcal{A}$ *has at most negligible advantage in the following game played against a challenger.*

(i) The challenger runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *and provides* pp *to* $\mathcal{A}$.

(ii) $\mathcal{A}$ *returns a randomness, a ring, a message, and a signature tuple* $(\mathsf{rr}, \mathsf{R}, \mathsf{M}, \sigma)$. *We say* $\mathcal{A}$ *wins if*

- $\mathsf{Verify}(\mathsf{mpk}, \mathsf{R}, \mathsf{M}, \sigma) = \top$, *where* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{MKGen}(\mathsf{pp}; \mathsf{rr})$, *and*
- $\mathsf{Judge}(\mathsf{mpk}, \mathsf{R}, \mathsf{vk}, \mathsf{M}, \sigma, \pi) = \perp$, *where* $(\mathsf{vk}, \pi) \leftarrow \mathsf{Open}(\mathsf{msk}, \mathsf{R}, \mathsf{M}, \sigma)$.

The advantage of $\mathcal{A}$ *is defined as* $\mathsf{Adv}^{\mathsf{Tra}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$.

Finally, tracing soundness requires that a signature cannot trace to two different users in the ring. This must hold even if all the users in the ring and the opener are corrupt.

Definition 5.2.6 (Tracing Soundness). *An accountable ring signature scheme* Π_{ARS} *is* traceable sound *if, for all* $\lambda \in \mathbb{N}$, *any PPT adversary* $\mathcal{A}$ *has at most negligible advantage in the following game played against a challenger.*

(i) The challenger runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *and provides* pp *to* $\mathcal{A}$.

(ii) $\mathcal{A}$ *returns an opener's public key, a ring, a message, a signature, and two verification keys and proofs* $(\mathsf{mpk}, \mathsf{R}, \mathsf{M}, \sigma, \{(\mathsf{vk}_b, \pi_b)\}_{b \in \{0,1\}})$. *We say* $\mathcal{A}$ *wins if*

- $\mathsf{vk}_0 \neq \mathsf{vk}_1$,
- $\mathsf{Judge}(\mathsf{mpk}, \mathsf{R}, \mathsf{vk}_0, \mathsf{M}, \sigma, \pi_0) = \top$,
- $\mathsf{Judge}(\mathsf{mpk}, \mathsf{R}, \mathsf{vk}_1, \mathsf{M}, \sigma, \pi_1) = \top$.

The advantage of $\mathcal{A}$ *is defined as* $\mathsf{Adv}^{\mathsf{TraS}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$.

5.3 Components

5.3.1 An Elgamal-type PKE

We start with an Elgamal-type PKE from group actions. One feature is the encryption does not require a hash function, which makes it potentially friendly to construct a proof system for a relation involving the encryption relation.

An Elgmal-type PKE without Hashes. $\Pi_{\mathsf{PKE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$,

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: On input the security parameter 1^λ, it outputs a public parameter $\mathsf{pp} = (G, \mathcal{E}, E_0, \star, \mathcal{M})$ where $(G, \mathcal{E}, E_0, \star)$ is the public parameter specified in Def. 2.3.3, $\mathcal{M}$ is the message space which is a subset of G and $|\mathcal{M}| \leq poly(\lambda)$.

- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$: On input a public parameter pp, it outputs a public key and a secret key $(\mathsf{pk} = (E_0, g \star E_0), \mathsf{sk} = g)$ where $g \leftarrow G$.

- $\mathsf{Enc}(\mathsf{pk}, \mathsf{M}) \to \mathsf{ct}$: On input a public key $\mathsf{pk} = (E_0, E)$ and a message $\mathsf{M} \in \mathcal{M}$, it outputs a ciphertext $\mathsf{ct} = (r \star E_0, \mathsf{M} \star (r \star E))$ where $r \leftarrow G$. We let $\mathsf{Enc}(\mathsf{pk}, \mathsf{M}; r)$ the denote the encryption using the randomness r.

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{M}$ or $\perp$: On input a secret key sk and a ciphertext $\mathsf{ct} = (E_0', E')$, it enumerates $\mathsf{M} \in \mathcal{M}$, tests whether $\mathsf{M} \star (\mathsf{sk} \star E_0') = E'$, and outputs M; otherwise, it outputs $\perp$.

The correctness of Π_{PKE} holds naturally. Due to the commutativity, we have $\mathsf{M} \star (r \star E) = \mathsf{M} \star (\mathsf{sk} \star (r \star E_0))$. Moreover, the following theorem shows that Π_{PKE} is IND-CPA.

Theorem 5.3.1. *The public key encryption scheme Π_{PKE} described is IND-CPA if* DDH *over $(G, \mathcal{E}, E_0, \star)$ is hard. Concretely, given an IND-CPA adversary $\mathcal{A}$, there exists a polynomial-time* DDH *adversary $\mathcal{B}$ over $(G, \mathcal{E}, E_0, \star)$ with $O(1)$ queries to $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{DDH}}(\mathcal{B}) = \frac{1}{2}\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\Pi_{\mathsf{PKE}}}(\mathcal{A})$.*

Proof. Say $\mathsf{pp} = (G, \mathcal{E}, E_0, \star, \mathcal{M}) \leftarrow \mathsf{Setup}$. Given the challenge (E_0, E_1, E_2, E_3) and access to the IND-CPA adversary $\mathcal{A}$, a reduction $\mathcal{B}$ proceeds as follows.

1. Set $\mathsf{pk} = (E_0, E_1)$ (without knowing sk).

2. Invoke the adversary $\mathcal{A}$ with the public key pk.

3. Upon receiving $m_0, m_1 \in \mathcal{M}$ from $\mathcal{A}$, $\mathcal{B}$ replies with $(E_2, m_b \star E_3)$ where $b \leftarrow \{0, 1\}$.

4. Output the truth value $(b == b')$ where $\mathcal{A}$ returns b'.

Write $E_1 = g_1 \star E_0$ and $E_2 = g_2 \star E_0$ where $g_1, g_2 \leftarrow G$. In the case that $E_3 = (g_1 + g_2) \star E_0$, it is clear that $\mathcal{B}$ returns the encryption of m_b up to $b \in \{0, 1\}$ following the same encryption distribution. In the case that $E_3 \leftarrow \mathcal{E}$, the instance might be malformed (i.e. (E_2, E_3) is not a valid ciphertext using pk). The input $(E_2, m_b \star E_3)$ follows the uniform distribution over $\mathcal{E}^2$. The reduction $\mathcal{B}$ can only win with negligible advantage in this case. Therefore,

$$\begin{aligned}\Pr[\mathcal{B} \text{ wins}] &= 1/2 \Pr[\mathcal{B} \text{ wins} \mid E_3 = (g_1 + g_2) \star E_0] + 1/2 \Pr[\mathcal{B} \text{ wins} \mid E_3 \neq (g_1 + g_2) \star E_0] \\ &= 1/2(1/2 + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\Pi_{\mathsf{PKE}}}(\mathcal{A})) + 1/2 \times 1/2.\end{aligned}$$

We have

$$\mathsf{Adv}^{\mathsf{DDH}}(\mathcal{B}) = \frac{1}{2}\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\Pi_{\mathsf{PKE}}}(\mathcal{A}).$$

□

Remark 5.3.2. *Due to the free action, for a public key $\mathsf{pk} = (E_0, E)$ of Π_{PKE}, there exists a unique $(\mathsf{M}, r) \in G^2$ such that $\mathsf{Enc}(\mathsf{pk}, \mathsf{M}; r) = \mathsf{ct}$ for a valid ciphertext $\mathsf{ct} = (Y_0, Y)$ by solving the equations $r \star E_0 = Y_0$, $(\mathsf{M} + r) \star E = Y$.*

A Proof System of Decryption.

In Fig. 5.1, we construct a base proof system for the encryption scheme, where the owner of the decryption key sk corresponding to pk is able to prove correctness of the decryption. Formally, we consider the relation

$$R_{\mathsf{open}} = \left\{ ((\mathsf{pk} = (E_0, E), \mathsf{ct}, I), \mathsf{sk}) \;\middle|\; \begin{array}{c} E = \mathsf{sk} \star E_0 \wedge \\ I = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \end{array} \right\}.$$

The idea of the proof system is simple. By writing $\mathsf{ct} = (Y_0, Y)$, the secret key sk is the witness for both (E_0, E) and $(Y_0, -I \star Y)$ with respect to the group action inverse relation R_{GAIP}. That is, $((E_0, E), \mathsf{sk}), ((Y_0, -I \star Y), \mathsf{sk}) \in R_{\mathsf{GAIP}}$. Fig. 5.1 repeats the graph-isomorphism type proof of knowledge in a parallel manner.

Theorem 5.3.3. *The sigma protocol $\Pi_{\Sigma}^{\mathsf{OPbase}}$ described in Fig. 5.1 has correctness and relaxed special soundness for the relation $\tilde{R}_{\mathsf{open}}$, where*

$$\tilde{R}_{\mathsf{open}} = \left\{ \Big(((\mathsf{pk} = (E_0, E), \mathsf{ct}, I), \mathsf{W}) \Big) \;\middle|\; \begin{array}{c} \mathsf{W} = \mathsf{sk} \in G \wedge \\ E = \mathsf{sk} \star E_0 \;\; \wedge I = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \;\; \textit{or} \\ \mathsf{W} = (x_1, x_2) \in \{0,1\}^* \wedge x_1 \neq x_2 \wedge \mathcal{O}(x_1) = \mathcal{O}(x_2) \end{array} \right\}.$$

Here, $\tilde{R}_{\mathsf{open}}$ relaxes R_{open} by taking the hash collision $\mathsf{W} = (x_1, x_2)$ into account.

Theorem 5.3.4. *The sigma protocol $\Pi_{\Sigma}^{\mathsf{OPbase}}$ described in Fig. 5.1 has honest-verifier zero-knowledge. Precisely, there exists a PPT simulator $\mathsf{Sim}^{\mathcal{O}}$ with access to a random oracle $\mathcal{O}$ such that, for any statement-witness pair $(\mathsf{X}, \mathsf{W}) \in R_{\mathsf{open}}$, $\mathsf{ch} \in \{0, 1\}$, and any computationally-unbounded adversary $\mathcal{A}$ that makes at most Q queries to the random oracle $\mathcal{O}$, we have*

$$\left| \Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, P'^{\mathcal{O}}(\mathsf{X}, \mathsf{W}, \mathsf{ch})) = 1] - \Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, \mathsf{Sim}^{\mathcal{O}}(\mathsf{X}, \mathsf{ch})) = 1] \right| \leq \frac{Q}{2^\lambda},$$

where $P' = (P'_1, P'_2)$ is a prover run on (X, W) with a challenge fixed to ch.

The proofs are relatively simple and can be easily deduced. As a beneficial learning exercise, we encourage readers to work through them independently.

5.3.2 The "traceable" OR-Proof

This section presents a traceable OR proof for the OR relation of R_{GAIP} in Fig. 5.2. Concretely, the prover is able to generate a proof of knowledge of a witness for one of several R_{GAIP} statements. Moreover, the proof endows a candidate the ability to trace the proof back to the prover. Our proof system (Σ-protocol-based) is based on an effective group action and the derived PKE described above.[4]

[4]In fact, it is not necessary for these two components be instantiated from the same group action. We refer [BDK+22] for more details.

```
round 1: $P_1'^{\mathcal{O}}((\mathsf{pk} = (E_0, E), \mathsf{ct} = (Y_0, Y), I), \mathsf{sk})$
  1: $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$
  2: $s' \leftarrow \mathcal{O}(\mathsf{Expand} \,\|\, \mathsf{seed})$                    ▷ Sample $r \in G$
  3: $\mathsf{C}_0 \leftarrow \mathcal{O}(\mathsf{Com} \,\|\, s' \star E)$
  4: $\mathsf{C}_1 \leftarrow \mathcal{O}(\mathsf{Com} \,\|\, (s' - I) \star Y)$
  5: $(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MerkleTree}(\mathsf{C}_1, \mathsf{C}_2)$
  6: Prover sends $\mathsf{com} \leftarrow \mathsf{root}$ to Verifier.

round 2: $V_1'(\mathsf{com})$
  1: $c \xleftarrow{\$} \{0,1\}$
  2: Verifier sends $\mathsf{ch} \leftarrow c$ to Prover.

round 3: $P_2'(\mathsf{sk}, \mathsf{ch})$
  1: $c \leftarrow \mathsf{ch}$
  2: if $c = 1$ then
  3:     $s'' \leftarrow s' + \mathsf{sk}$
  4:     $\mathsf{resp} \leftarrow s''$
  5: else
  6:     $\mathsf{resp} \leftarrow \mathsf{seed}$
  7: Prover sends $\mathsf{resp}$ to Verifier

Verification: $V_2'^{\mathcal{O}}(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$
  1: $(\mathsf{root}, c) \leftarrow (\mathsf{com}, \mathsf{ch})$
  2: if $c = 1$ then
  3:     $s'' \leftarrow \mathsf{resp}$
  4:     $\widetilde{\mathsf{C}}_1 \leftarrow \mathcal{O}(\mathsf{Com} \,\|\, s'' \star E_0)$
  5:     $\widetilde{\mathsf{C}}_2 \leftarrow \mathcal{O}(\mathsf{Com} \,\|\, s'' \star Y_0)$
  6:     $\widetilde{\mathsf{root}} \leftarrow \mathsf{MerkleTree}(\widetilde{\mathsf{C}}_1, \widetilde{\mathsf{C}}_2)$
  7:     Verifier accepts only if $\widetilde{\mathsf{root}} = \mathsf{root}$.
  8: else
  9:     Repeat round 1 with $\mathsf{seed} \leftarrow \mathsf{resp}$.
 10:     Output accept if the computation results in root, and reject otherwise.
```

Figure 5.1: Construction of the base decryption sigma protocol $\Pi_\Sigma^{\mathsf{OPbase}} = (P' = (P_1', P_2'), V' = (V_1', V_2'))$ for the relation R_{open}. Informally, $O(\mathsf{Expand}\|\cdot)$ and $O(\mathsf{Com}\|\cdot)$ are a PRG and a commitment scheme instantiated by the random oracle, respectively.

Let $(G, \mathcal{E}, E_0, \star)$ forms an effective group action and $\mathsf{pp}' = (G, \mathcal{E}, E_0, \star, \mathcal{M})$ be the PKE public parameter sharing the same group. Moreover, let $(\mathsf{pk} = (E_0, E), \mathsf{sk}) \in \mathsf{KeyGen}(\mathsf{pp}')$. The relation R_{sig}, conceptually described in Sec. 5.1, can be equivalently rewritten as follows:

$$R_{\mathsf{sig}} = \left\{ \Big(((X_i)_{i\in[N]}, \mathsf{pk}, \mathsf{ct}), (I, s, r) \Big) \;\middle|\; \begin{array}{c} (I, s, r) \in [N] \times G \times G \\ X_I = s \star E_0 \,\wedge \\ \mathsf{ct} = (Y_0, Y) = \mathsf{Enc}(\mathsf{pk}, I; r) \end{array} \right\},$$

where N represents the number of members this ring/group and X_i indicates the verification key of each member for each $i \in [N]$. To clarify, the "OR" relation here is slightly different from the one considered in [CDS94] where there are N witnesses for the same statement $(X_i)_{i\in[N]}$. Instead, the "OR" relation here means the ciphertext ct is encrypted from one of the N messages.

High-level Idea. The high-level idea of the construction is fairly simple. The relation R_{sig} can be decomposed as relations R_1 and R_2 as follows.

$$R_1 = \left\{ \Big((X_i)_{i\in[N]}, (I, s) \Big) \;\middle|\; \begin{array}{c} (I, s) \in [N] \times G \\ X_I = s \star E_0 \end{array} \right\},$$

$$R_2 = \left\{ ((\mathsf{pk}, \mathsf{ct}), (I, r)) \middle| \begin{array}{c} (I, r) \in [N] \times G \\ \mathsf{ct} = (Y_0, Y) = \mathsf{Enc}(\mathsf{pk}, I; r) \end{array} \right\}.$$

We apply the OR-proof [BKP20] and construct two proof systems for R_1 and R_2 respectively. Due to statistical zero-knowledge, even if the verifier knows the message space $\mathcal{M}$ and the size is in $poly(\lambda)$, the proof does not disclose the information of I. Different from the challenge-XOR method from [CDS94], the logarithm OR-proof of [BKP20] hides the index by shuffling the commitments of the instances. To ensure the consistency of the index I in the two proofs, the prover concatenates the commitments in two proofs with respect to the index $i \in [N]$, and shuffles the concatenated commitments. This leads to our proof system for the relation R_{sig} with the "traceable" feature. That is, the owner of the public key pk can decrypt and obtain the index. Looking ahead, this is precisely the "traceability" for the ring signature and naturally gives us a group signature. On top of the traceability, the member who has the decryption key (i.e. the manager) generates another proof for the correctness of the decryption wrt the public key using Fig. 5.1.

Sigma Protocol for R_{sig}. We now sketch the base traceable OR sigma protocol $\Pi_\Sigma^{\mathsf{ARSbase}}$. A prover with witness $(I, s, r) \in [N] \times G \times G$ first samples $(s', r') \xleftarrow{\$} G \times G$, and $(\{\mathsf{bits}_i\}_{i\in[N]}) \leftarrow \{0,1\}^{\lambda N}$. Then, it computes commitments

$$\mathsf{C}_i = \mathcal{O}(\mathsf{Com} \parallel s' \star X_i \parallel (r' \star Y_0, (-i + r') \star Y) \parallel \mathsf{bits}_i) \quad \forall i \in [N],$$

and builds a (index-hiding) Merkle tree with $(\mathsf{C}_1, \ldots, \mathsf{C}_N)$ as its leaves, obtaining root. Here, notice $r' \star Y_0, (-i + r') \star Y$ is simply $((r' + r) \star E_0, (r' + r) \star E)$ when $i = I$. Then, the prover sends $\mathsf{com} = \mathsf{root}$ to the verifier as the commitment of the sigma protocol. The verifier, in turn, responds with a uniform challenge $\mathsf{ch} \in \{0, 1\}$.

If the challenge bit ch is 0, then the prover sends (s', r') and the commitment randomness $\{\mathsf{bits}_i\}_{i\in[N]}$. That is, all the randomness it generated in the first round. The verifier then can reconstruct the Merkle tree and verify that the root of the obtained tree is equal to root.

If the challenge bit ch is equal to 1, then the prover computes $s'' = s' + s$, $r'' = r' + r$ and sends to the verifier. The verifier computes $\widetilde{\mathsf{C}}_I = \mathcal{O}(\mathsf{Com} \parallel s'' \star E_0 \parallel (r'' \star Y_0, r'' \star Y) \parallel \mathsf{bits}_I)$ and uses the received path to reconstruct $\widetilde{\mathsf{root}}$ of the Merkle tree. The verifier checks whether $\widetilde{\mathsf{root}} = \mathsf{root}$.

Optimization. To reduce the communication bandwidth, we use a pseudorandom number generator (PRNG), denoted by Expand, on input a uniform seed $\mathsf{seed} \in \{0,1\}^\lambda$ to produce all randomness, including the group elements s', r' and all commitment randomness values bits_1, $\ldots, \mathsf{bits}_N$ (part of the response for $\mathsf{ch} = 0$). As a consequence, if the challenge bit is 0, the prover responds with seed so that the verifier can generate $(s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N)$ with Expand. The response for the challenge bit $\mathsf{ch} = 1$ remains the same. We instantiate the PRNG by a random oracle $\mathcal{O}(\mathsf{Expand} \parallel \cdot)$. Looking ahead, using a PRNG not only provides efficiency, but also proves to be essential for our cryptosystem by providing the online extractability after compilation into a NIZK under the random oracle model. The insight is the seeds, binding with the response for the challenge bit $\mathsf{ch} = 0$, prevents a malicious prover from forging a response afterward. This allows an efficient and online extraction in the random oracle model. Finally, we instantiate the collision-resistant hash function $\mathcal{H}_{\mathsf{Coll}}(\cdot)$ used in our Merkle tree by a random oracle $\mathcal{O}(\mathsf{Coll} \parallel \cdot)$. The resulting scheme is given in Fig. 5.2.

```
round 1: $P_1'^{\mathcal{O}}(((X_i)_{i\in[N]}, \mathsf{pk} = (E_0, E), \mathsf{ct} = (Y_0, Y)), (I, s, r))$
 1: $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$
 2: $(s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N) \leftarrow \mathcal{O}(\mathsf{Expand} \parallel \mathsf{seed})$   ▷ Sample $(s', r') \in G \times G$ and
    $\mathsf{bits}_i \in \{0,1\}^\lambda$
 3: for $i$ from 1 to $N$ do
 4:     $(T_i, \mathsf{ct}_i) \leftarrow (s' \star X_i, (r' \star Y_0, (r' - i) \star Y))$
 5:     $\mathsf{C}_i \leftarrow \mathcal{O}(\mathsf{Com} \parallel T_i \parallel \mathsf{ct}_i \parallel \mathsf{bits}_i)$   ▷ Create commitments $\mathsf{C}_i \in \{0,1\}^{2\lambda}$
 6: $(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MerkleTree}(\mathsf{C}_1, \cdots, \mathsf{C}_N)$
 7: Prover sends $\mathsf{com} \leftarrow \mathsf{root}$ to Verifier.

round 2: $V_1'(\mathsf{com})$
 1: $c \xleftarrow{\$} \{0,1\}$
 2: Verifier sends $\mathsf{ch} \leftarrow c$ to Prover.

round 3: $P_2'((I, s, r), \mathsf{ch})$
 1: $c \leftarrow \mathsf{ch}$
 2: if $c = 1$ then
 3:     $(s'', r'') \leftarrow (s' + s, r' + r)$
 4:     $\mathsf{path} \leftarrow \mathsf{getMerklePath}(\mathsf{tree}, I)$
 5:     $\mathsf{resp} \leftarrow (s'', r'', \mathsf{path}, \mathsf{bits}_I)$
 6: else
 7:     $\mathsf{resp} \leftarrow \mathsf{seed}$
 8: Prover sends $\mathsf{resp}$ to Verifier

Verification: $V_2'^{\mathcal{O}}(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$
 1: $(\mathsf{root}, c) \leftarrow (\mathsf{com}, \mathsf{ch})$
 2: if $c = 1$ then
 3:     $(s'', r'', \mathsf{path}, \mathsf{bits}) \leftarrow \mathsf{resp}$
 4:     $(\widetilde{T}, \widetilde{\mathsf{ct}}) \leftarrow (s'' \star E_0, (r'' \star E_0, r'' \star E))$
 5:     $\widetilde{\mathsf{C}} \leftarrow \mathcal{O}(\mathsf{Com} \parallel \widetilde{T} \parallel \widetilde{\mathsf{ct}} \parallel \mathsf{bits})$
 6:     $\widetilde{\mathsf{root}} \leftarrow \mathsf{ReconstructRoot}(\widetilde{\mathsf{C}}, \mathsf{path})$
 7:     Verifier accepts only if $\widetilde{\mathsf{root}} = \mathsf{root}$.
 8: else
 9:     Repeat round 1 with $\mathsf{seed} \leftarrow \mathsf{resp}$.
10:     Output accept if the computation res-
        ults in root, and reject otherwise.
```

Figure 5.2: Construction of the base traceable OR sigma protocol $\Pi_\Sigma^{\mathsf{ARSbase}} = (P' = (P_1', P_2'), V' = (V_1', V_2'))$ for the relation R_{sig}. Informally, $O(\mathsf{Expand}\|\cdot)$ and $O(\mathsf{Com}\|\cdot)$ are a PRG and a commitment scheme instantiated by the random oracle, respectively.

The following Thms. 5.3.5 and 5.3.6 summarize the security of our sigma protocol. It should be noted that in Thm. 5.3.5, our sigma protocol satisfies special soundness for the relations R_{sig} and $\tilde{R}_{\mathsf{sig}}$ such that $R_{\mathsf{sig}} \subset \tilde{R}_{\mathsf{sig}}$. The subtle difference is that $\tilde{R}_{\mathsf{sig}}$ is a relaxation of R_{sig} capturing the scenario where the extractor may extract a witness that forms a collision in the random oracle. This distinction has no real-world implications, as we are capable of converting such a sigma protocol into a multi-proof online extractable NIZK for both relations R_{sig} and $\tilde{R}_{\mathsf{sig}}$.

Theorem 5.3.5. *The sigma protocol $\Pi_\Sigma^{\mathsf{ARSbase}}$ has correctness and relaxed special soundness for the relation $\tilde{R}_{\mathsf{sig}}$, where*

$$\tilde{R}_{\mathsf{sig}} = \left\{ \Big(((X_i)_{i\in[N]}, \mathsf{pk}, \mathsf{ct}), \mathsf{W} \Big) \;\middle|\; \begin{array}{c} \mathsf{W} = (I, s, r) \in [N] \times G \times G \\ \wedge\ X_I = s \star X_0 \wedge \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, I; r) \textit{ or} \\ \mathsf{W} = (x_1, x_2) \in \{0,1\}^* \wedge\ x_1 \neq x_2 \wedge \mathcal{O}(x_1) = \mathcal{O}(x_2) \end{array} \right\}.$$

Here, $\tilde{R}_{\mathsf{sig}}$ relaxes R_{sig} by taking the hash collision $\mathsf{W} = (x_1, x_2)$ into account.

Proof. Correctness. Say the prover honestly runs $\Pi_\Sigma^{\mathsf{ARSbase}}$ on an input (I, s, r) satisfying $X_I = s \star X_0$ and $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, I; r)$. If $\mathsf{ch} = 0$, then the verifier repeats the computation in the commitment phase (i.e. Round 1 in Fig. 5.2) and therefore obtains the same output. If $\mathsf{ch} = 1$, then the verifier computes $\widetilde{T} = s'' \star X_0$ and $\widetilde{\mathsf{ct}} = (r'' \star E_0, r'' \star E)$ where $s'' = s' + s$ and $r'' = r' + r$. Besides, since $\widetilde{T}$ is equal to $T_I = s' \star X_I$, $\widetilde{\mathsf{ct}}$ is equal to $\mathsf{ct}_I = (r' \star Y_0, (r' - I) \star Y)$ and $\widetilde{\mathsf{C}} = \mathcal{O}(\mathsf{Com} \parallel \widetilde{T} \parallel \widetilde{\mathsf{ct}} \parallel \mathsf{bits})$ is equal to the leaf $\widetilde{\mathsf{C}} = \mathsf{C}_I \in \{\mathsf{C}_1, \cdots, \mathsf{C}_N\}$, the verifier reconstructs the root $\widetilde{\mathsf{root}}$ which results in the same root. Hence, the protocol has correctness.

Relaxed Special Soundness. Given two valid transcripts for the same statement and on the same commitment, $(\mathsf{com}, 0, \mathsf{seed})$ and $(\mathsf{com}, 1, (s'', r'', \mathsf{path}, \mathsf{bits}))$ where $\mathsf{com} = \mathsf{root}$, an extraction algorithm $\mathsf{Extract}$ for a witness for the relation $\tilde{R}_{\mathsf{sig}}$ proceeds as follows.

1. Generate $(s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N) \leftarrow \mathcal{O}(\mathsf{Expand} \parallel \mathsf{seed})$.
2. Construct $\mathsf{C}_1, \cdots, \mathsf{C}_N$ such that the Merkle Tree with leaves $(\mathsf{C}_1, \cdots, \mathsf{C}_N)$ has the root equal to root. Remark that running $V_2'(\mathsf{com}, 1, (s'', r'', \mathsf{path}, \mathsf{bits}))$ will result in the same root via $\mathsf{ReconstructRoot}$.
3. In the two procedures of computing the root using $\mathsf{MerkleTree}$, if there exists $x_1 \neq x_2$ such that $\mathcal{O}(\mathsf{Coll} \parallel x_1) = \mathcal{O}(\mathsf{Coll} \parallel x_2)$, output $\mathsf{W} = (\mathsf{Coll} \parallel x_1, \mathsf{Coll} \parallel x_2)$ and stop. Otherwise, continue.
4. In the two procedures of computing the root using $\mathcal{O}(\mathsf{Com} \parallel \cdot)$, if there exists $x_1 \neq x_2$ such that $\mathcal{O}(\mathsf{Com} \parallel x_1) = \mathcal{O}(\mathsf{Com} \parallel x_2)$, output $\mathsf{W} = (\mathsf{Com} \parallel x_1, \mathsf{Com} \parallel x_2)$ and stop. Otherwise, find and set $\widetilde{I} \in [N]$ such that $\mathsf{bits} = \mathsf{bits}_{\widetilde{I}}$ and continue.
5. Output $(\widetilde{I}, -s' + s'', -r' + r'')$.

If the condition of Step 3. fails, we have $\widetilde{\mathsf{C}} = \mathcal{O}(\mathsf{Com} \parallel s'' \star X_0 \parallel r'' \star_{\mathsf{pk}} E_0, \star E \parallel \mathsf{bits})$ is equal to $\mathsf{C}_{\widetilde{I}} \in \{\mathsf{C}_1, \cdots, \mathsf{C}_N\}$ for some $\widetilde{I} \in [N]$. If the condition of Step 4. fails, due to $\widetilde{\mathsf{C}} = \mathsf{C}_{\widetilde{I}}$, we have $s' \star X_{\widetilde{I}} = s'' \star X_0$, $(r' \star Y_0, (r' - \widetilde{I}) \star Y) = (r'' \star E_0, r'' \star E)$, and $\mathsf{bits} = \mathsf{bits}_{\widetilde{I}}$.

Write $\widetilde{s} = -s' + s''$ and $\widetilde{r} = -r' + r''$ where $\mathsf{W} = (\widetilde{I}, \widetilde{s}, \widetilde{r})$ is the final output of $\mathsf{Extract}$, if no collisions occur. Here, the equalities $\widetilde{s} \star X_0 = X_{\widetilde{I}}$ and $(\widetilde{r} \star E_0, (\widetilde{r} + \widetilde{I}) \star E) = \mathsf{ct}$ follow directly from the relations $s' \star X_{\widetilde{I}} = s'' \star X_0$ and $(r' \star Y_0, (r' - \widetilde{I}) \star Y) = (r'' \star E_0, r'' \star E)$, respectively. Therefore, the extractor outputs $\mathsf{W} \in \tilde{R}_{\mathsf{sig}}$. □

Theorem 5.3.6. *The sigma protocol* $\Pi_\Sigma^{\mathsf{ARSbase}}$ *has honest-verifier zero-knowledge. Precisely, there exists a PPT simulator* $\mathsf{Sim}^{\mathcal{O}}$ *with access to a random oracle* $\mathcal{O}$ *such that, for any statement-witness pair* $(\mathsf{X}, \mathsf{W}) \in R_{\mathsf{sig}}$*,* $\mathsf{ch} \in \{0,1\}$*, and any computationally-unbounded adversary* $\mathcal{A}$ *that makes at most* Q *queries to the random oracle* $\mathcal{O}$*, we have*

$$\left|\Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, P'^{\mathcal{O}}(\mathsf{X}, \mathsf{W}, \mathsf{ch})) = 1] - \Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, \mathsf{Sim}^{\mathcal{O}}(\mathsf{X}, \mathsf{ch})) = 1]\right| \leq \frac{Q}{2^\lambda},$$

where $P' = (P'_1, P'_2)$ *is a prover run on* (X, W) *with a challenge fixed to* ch.

Proof. Assume the adversary makes Q_{Expand} and Q_{Com} queries to the random oracles of the form $\mathcal{O}(\mathsf{Expand} \parallel \cdot)$ and $\mathcal{O}(\mathsf{Com} \parallel \cdot)$, respectively. We have $Q_{\mathsf{Expand}} + Q_{\mathsf{Com}} \leq Q$. The PPT simulator $\mathsf{Sim}^{\mathcal{O}}$, on input $(\mathsf{X}, \mathsf{ch})$, proceeds as follows.

- If $\mathsf{ch} = 0$, the simulator executes as $P'^{\mathcal{O}}(\mathsf{X}, \perp, \mathsf{ch})$, where notice P' does not require the witness when $\mathsf{ch} = 0$. Concretely, the simulator outputs $(\mathsf{com} = \mathsf{root}, \mathsf{ch} = 0, \mathsf{resp} = \mathsf{seed})$ where $\mathsf{root}, \mathsf{seed}$ are honestly generated as in the execution of $P_1'^{\mathcal{O}}$.
- If $\mathsf{ch} = 1$, the simulator uniformly samples (s'', r'') from $G \times G$, and bits from $\{0,1\}^\lambda$. It computes $\mathsf{C}_1 = \mathcal{O}(\mathsf{Com} \parallel s'' \star X_0 \parallel (r'' \star E_0, r'' \star E) \parallel \mathsf{bits})$. It then uniformly samples dummy commitments C_i for $i \in \{2, \ldots, N\}$ from $\{0,1\}^{2\lambda}$, and computes the (index-hiding) Merkle tree $(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MerkleTree}(\mathsf{C}_1, \ldots, \mathsf{C}_N)$. After that, it extracts the path $\mathsf{path} \leftarrow \mathsf{getMerklePath}(\mathsf{tree}, 1)$ in the tree and sets $\mathsf{com} = \mathsf{root}$, and $\mathsf{resp} = (s'', r'', \mathsf{path}, \mathsf{bits})$. Finally, the simulator returns $(\mathsf{com}, \mathsf{ch} = 1, \mathsf{resp})$.

In the first case, the procedure of Sim is identical to a real honest prover, the transcripts are following the same distribution. Hence transcripts generated by $P'^{\mathcal{O}}$ and $\mathsf{Sim}^{\mathcal{O}}$ are indistinguishable to the adversary $\mathcal{A}$. Therefore, we have

$$\left|\Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, P'^{\mathcal{O}}(\mathsf{X}, \mathsf{W}, \mathsf{ch} = 0)) = 1]\right| = \left|\Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, \mathsf{Sim}^{\mathcal{O}}(\mathsf{X}, \mathsf{ch} = 0)) = 1]\right|.$$

To conclude the proof, it suffices to show that the statistical difference between the output produced by adversary $\mathcal{A}$ and the output of a real prover is bounded by $\frac{Q}{2^\lambda}$.

We use a hybrid argument by introducing a series of simulators $\mathsf{Sim}_0 = P', \ldots, \mathsf{Sim}_4 = \mathsf{Sim}$, gradually changing from the honest prover P' to Sim, to show that they are indistinguishable with overwhelming probability. We fix an adversary $\mathcal{A}$, $(\mathsf{X}, \mathsf{W}) \in R_{\mathsf{sig}}$, and for each $i \in \{0, 1, \ldots, 4\}$, we denote by E_i the event that $\mathcal{A}^{\mathcal{O}}(1^\lambda, \mathsf{Sim}_i^{\mathcal{O}}(\mathsf{X}, \mathsf{ch} = 1)) = 1$.

- Sim_1 is identical to Sim_0 except that instead of using Expand to generate $s', r', \{\mathsf{bits}_i\}_{i \in [N]}$, the simulator generates these by sampling uniformly at random from the corresponding domains. This does not change the view of $\mathcal{A}$, unless the adversary queries $\mathcal{O}$ on input $(\mathsf{Expand} \parallel \mathsf{seed})$. Since seed has λ bits of min-entropy and because it is information-theoretically hidden from $\mathcal{A}$, the probability that $\mathcal{A}$ queries $\mathcal{O}$ on this input is bounded by $Q_{\mathsf{Expand}}/2^\lambda$. That is, $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_0]| \leq \frac{Q_{\mathsf{Expand}}}{2^\lambda}$.

- Sim_2 is identical to Sim_2 except that all the commitments C_i for $i \in [N] \setminus \{I\}$ are generated uniformly at random. This does not change the view of $\mathcal{A}$, unless the adversary queries $\mathcal{O}$ on input $(\mathsf{Com} \parallel T_i \parallel \mathsf{ct}_i \parallel \mathsf{bits}_i)$ for any $i \in [N] \setminus \{I\}$, where $T_i = s' \star X_i$ and $\mathsf{ct}_i = (r' \star Y_0, (r' - i) \star Y)$. Since for any $i \in [N] \setminus \{I\}$ the string bits_i has λ bits of min-entropy and because it is information-theoretically hidden from $\mathcal{A}$, the probability that $\mathcal{A}$ queries $\mathcal{O}$ on input $(\mathsf{Com} \parallel T_i \parallel \mathsf{ct}_i \parallel \mathsf{bits}_i)$ is bounded by $Q_{\mathsf{Com}}/2^\lambda$. That is, $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_1]| \le \frac{Q_{\mathsf{Com}}}{2^\lambda}$.

- Sim_3 is identical to Sim_3 except that instead of computing s'', r'' as $s' + s, r' + r$, the simulator generates these two values by sampling uniformly at random from G instead. Both the distributions are uniform over G. Therefore, we have $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_2]| = 0$.

- $\mathsf{Sim}_4 = \mathsf{Sim}$ is identical to Sim_4 except that the simulator uses $I = 1$ instead of the value I in the witness W. These two simulators are indistinguishable because the Merkle tree is index-hiding by Lemma 2.10 of [BKP20]. Formally, we have $|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_3]| = 0$.

Collecting the bounds, the result follows. □

Repetitions and Adding Salts. Next, we use a standard parallel repetition approach to reduce the soundness error to be negligible. Also, we add salts to the random oracle. Concretely, we prefix a salt (a random string) and the session identifier (i.e. $(\mathsf{salt} \parallel i)$) to the random oracle when used within the i-th parallel execution of the base sigma protocol (e.g. $\Pi_\Sigma^{\mathsf{ARSbase}}, \Pi_\Sigma$). In particular, throughout each such execution, the participants use the random oracle $\mathcal{O}_i(\cdot) = \mathcal{O}(\mathsf{salt} \parallel i \parallel \cdot)$. The salt is used as a prefix also within the construction of Merkle trees. Salt benefits the protocol in having a tighter reduction and resisting multi-target attacks, such as those in [DN19]. At a glance, adding a salt makes no difference in a sigma protocol but it's quite beneficial to a group (ring) signature scheme after we apply Fiat-Shamir transform. Roughly, in the anonymity game (Def. 5.2.3) each oracle $\mathcal{O}$ query made by the adversary will only give useful information to at most one challenge signature due to the distinct prefix salt. In contrast, without salts an oracle query of $\mathcal{O}$ can give useful information to *each* challenge signature. We refer to Thm. 5.3.13 to see how salts mitigate the quadratic looseness in the reduction.

Theorem 5.3.7. *The sigma protocol $\Pi_\Sigma^{\mathsf{tOR}}$, compiled from $\Pi_\Sigma^{\mathsf{ARSbase}}$ in Fig. 5.2 via the compiler in Fig. 5.3, has correctness, high min-entropy, and relaxed special soundness for the relations $\tilde{R}_{\mathsf{sig}}$, where the relations are identical to those used in Thm. 5.3.5.*
Similarly, the result holds the same for compiled Fig. 5.1 for the relation $\tilde{R}_{\mathsf{open}}$.

Proof. *Correctness.* The correctness follows straightforwardly from Thm. 5.3.5.
High Min-Entropy. Containing a random salt of length 2λ, the commitment com has at least 2λ bits of min-entropy.

Relaxed Special Soundness. The proof is similar to the one for the relaxed special soundness of $\Pi_\Sigma^{\mathsf{ARSbase}}$. Given $(\mathsf{com}, \mathsf{ch} = \mathbf{c}, \mathsf{resp})$ $(\mathsf{com}, \mathsf{ch}' = \mathbf{c}', \mathsf{resp}')$ be two accepting transcripts for the same statement, a extractor proceeds as follows.

```
round 1: $P_1^{\mathcal{O}}(\mathsf{X}, \mathsf{W})$
1: $\mathsf{salt} \xleftarrow{\$} \{0,1\}^{2\lambda}$
2: for $j \in [\lambda]$ do
3:     $\mathcal{O}_j(\cdot) := \mathcal{O}(\mathsf{salt} \parallel j \parallel \cdot)$
4:     $\mathsf{com}_j \leftarrow P_1'^{\mathcal{O}_j}(\mathsf{X}, \mathsf{W}; \mathsf{seed}_j)$
5: Prover sends $\mathsf{com} \leftarrow (\mathsf{salt}, \mathsf{com}_1, \cdots, \mathsf{com}_\lambda)$ to Verifier.
round 2: $V_1(\mathsf{com})$
1: $\mathbf{c} \xleftarrow{\$} \{0,1\}^{\lambda}$
2: Verifier sends $\mathsf{ch} \leftarrow \mathbf{c}$ to Prover.
round 3: $P_2^{\mathcal{O}}(\mathsf{X}, \mathsf{ch})$
1: $\mathbf{c} = (c_1, \cdots, c_\lambda) \leftarrow \mathsf{ch}$
2: for $j \in [\lambda]$ do
3:     if $c_j = 1$ then
4:         $\mathsf{resp}_j \leftarrow P_2'^{\mathcal{O}_j}(\mathsf{X}, c_j; \mathsf{seed}_j)$        ▷ Execute $P_2'$ on state $\mathsf{seed}_j$
5:     else
6:         $\mathsf{resp}_j \leftarrow \mathsf{seed}_j$
7: Prover sends $\mathsf{resp} \leftarrow (\mathsf{resp}_1, \cdots, \mathsf{resp}_\lambda)$ to Verifier
Verification: $V_2^{\mathcal{O}}(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$
1: $((\mathsf{salt}, \mathsf{com}_1, \cdots, \mathsf{com}_\lambda), \mathbf{c} = (c_1, \cdots, c_\lambda)) \leftarrow (\mathsf{com}, \mathsf{ch})$
2: $\mathsf{resp} \leftarrow (\mathsf{resp}_1, \cdots, \mathsf{resp}_\lambda)$
3: for $j \in [\lambda]$ do
4:     $\mathcal{O}_j(\cdot) := \mathcal{O}(\mathsf{salt} \parallel j \parallel \cdot)$
5:     Verifier outputs reject if $V_2'^{\mathcal{O}_j}(\mathsf{com}_j, c_j, \mathsf{resp}_j)$ outputs reject.
6: Verifier outputs accept.
```

Figure 5.3: The main sigma protocol $\Pi_\Sigma = (P = (P_1, P_2), V = (V_1, V_2))$ compiler using λ-repetitions and adding salts for a base sigma protocol (e.g. $\Pi_\Sigma^{\mathsf{ARSbase}}, \Pi_\Sigma^{\mathsf{OPbase}}$ in Figs. 5.1 and 5.2). The base sigma protocol here is given access to salted random oracles derived from $\mathcal{O}$. We denote two derived protocols $\Pi_\Sigma^{\mathsf{tOR}}$ and Π_Σ^{OP} respectively.

1. Find $j \in [\lambda]$ such that $c_j = 0, c'_j = 1$.
2. Obtain $(\mathsf{com}_j, 0, \mathsf{resp}_j)$ and $(\mathsf{com}_j, 1, \mathsf{resp}'_j)$ where $\mathsf{resp}_j, \mathsf{resp}'_j$ are the j-th entries extracted from $\mathsf{resp}, \mathsf{resp}'$ respectively.
3. Invoke the extractor $(\mathsf{Extract})$ of $\Pi_\Sigma^{\mathsf{ARSbase}}$ (Fig. 5.2) in Thm. 5.3.5, we can output a witness for the relation $\tilde{R}_{\mathsf{sig}}$.

Remark that in Step 2., $(\mathsf{com}_j, 0, \mathsf{resp}_j)$ and $(\mathsf{com}_j, 1, \mathsf{resp}'_j)$ are the valid transcripts for $\Pi_\Sigma^{\mathsf{ARSbase}}$, which justify the application of the extractor in Step 3. Therefore, by Thm. 5.3.5, the result follows.

□

Theorem 5.3.8. *The sigma protocol $\Pi_\Sigma^{\mathsf{tOR}}$, compiled $\Pi_\Sigma^{\mathsf{ARSbase}}$ in Fig. 5.2 via the compiler in Fig. 5.3, has honest-verifier zero-knowledge. Precisely, there exists a PPT simulator $\mathsf{Sim}^{\mathcal{O}}$ with access to a random oracle $\mathcal{O}$ such that, for any statement-witness pair $(\mathsf{X}, \mathsf{W}) \in R_{\mathsf{sig}}$, $\mathsf{ch} \in \{0,1\}^\lambda$ and any computationally-unbounded adversary $\mathcal{A}$ that makes at most Q queries of the form $(\mathsf{salt} \parallel \cdot)$ to the random oracle $\mathcal{O}$, where salt is the salt value included in the transcript returned by $\widetilde{P}$ or Sim, we have*

$$\left|\Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, P^{\mathcal{O}}(\mathsf{X}, \mathsf{W}, \mathsf{ch})) = 1] - \Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, \mathsf{Sim}^{\mathcal{O}}(\mathsf{X}, \mathsf{ch})) = 1]\right| \leq \frac{Q}{2^\lambda},$$

where $P = (P_1, P_2)$ is an honest prover running on (X, W) with a challenge fixed to ch. Similarly, the result holds the same for the base sigma protocol in Fig. 5.1 transformed via Fig. 5.3 for the relation R_{open}.

Proof. The PPT simulator $\mathsf{Sim}^{\mathcal{O}}(\mathsf{X}, \mathsf{ch})$ for the main sigma protocol $\Pi_\Sigma^{\mathsf{tOR}}$ proceeds as in Fig. 5.4 where the simulator, used in the base sigma protocol $\Pi_\Sigma^{\mathsf{ARSbase}}$ in Thm. 5.3.6, is denoted by Sim' as a subroutine. Say the adversary makes Q_i queries to the random oracle of the form $\mathcal{O}(\mathsf{salt} \parallel i \parallel \cdot)$ for $i \in [\lambda]$. We have $\Sigma_1^\lambda Q_i \leq Q$.

```
Sim^O(X, ch)
1: $\mathbf{c} = (c_1, \cdots, c_\lambda) \leftarrow \mathsf{ch}$
2: $\mathsf{salt} \leftarrow \{0,1\}^{2\lambda}$
3: for $j \in [\lambda]$ do
4:     $\mathcal{O}_j(\cdot) := \mathcal{O}(\mathsf{salt} \parallel j \parallel \cdot)$
5:     $(\mathsf{com}_j, c_j, \mathsf{resp}_j) \leftarrow \mathsf{Sim}'^{\mathcal{O}_i}(\mathsf{X}, c_i)$
6: $\mathsf{com} \leftarrow (\mathsf{salt}, \mathsf{com}_1, \cdots, \mathsf{com}_\lambda)$
7: $\mathsf{resp} \leftarrow (\mathsf{resp}_1, \cdots, \mathsf{resp}_\lambda)$
8: return $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$
```

Figure 5.4: Zero-knowledge simulator Sim for the main sigma protocol $\Pi_\Sigma^{\mathsf{tOR}}$

The simulator Sim is identical to P except that the simulator uses the base simulator subroutine Sim' of Fig. 5.2 in Thm. 5.3.6 to generate the transcripts $(\mathsf{com}_j, c_j, \mathsf{resp}_j)$ for each $j \in [\lambda]$. By Theorem 5.3.6, the distinguishing advantage of the adversary is bounded by $\frac{Q_i}{2^\lambda}$ for each $j \in [\lambda]$ such that $c_j = 1$. Therefore,

$$\left|\Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, P^{\mathcal{O}}(\mathsf{X}, \mathsf{W}, \mathsf{ch})) = 1] - \Pr[\mathcal{A}^{\mathcal{O}}(1^\lambda, \mathsf{Sim}^{\mathcal{O}}(\mathsf{X}, \mathsf{ch})) = 1]\right| \leq \frac{\Sigma_1^\lambda Q_i}{2^\lambda} \leq \frac{Q}{2^\lambda}.$$

Collecting the bounds, we obtain the bound in the statement. □

5.3.3 Non-interactive Zero Knowledge

In this subsection, we make the protocol non-interactive using the Fiat-Shamir transform. Then we prove the multi-proof online-extractability as a NIZK of the resulting scheme.

```
$\mathsf{Prove}^{\mathcal{O}}(\mathsf{lbl}, ((X_i)_{i\in[N]}, \mathsf{pk}, \mathsf{ct}), (I, s_I, r))$
  1: $\mathsf{resp} := \perp$
  2: while $\mathsf{resp} = \perp$ do
  3:     $\mathsf{com} \leftarrow P_1^{\mathcal{O}}(((X_i)_{i\in[N]}, \mathsf{pk}, \mathsf{ct}), (I, s_I, r))$
  4:     $\mathsf{ch} \leftarrow \mathcal{O}(\mathsf{FS} \parallel \mathsf{lbl} \parallel ((X_i)_{i\in[N]}, \mathsf{pk}, \mathsf{ct}) \parallel \mathsf{com})$
  5:     $\mathsf{resp} \leftarrow P_2^{\mathcal{O}}((I, s_I, r), \mathsf{ch})$
  6: return $\pi \leftarrow (\mathsf{com}, \mathsf{ch}, \mathsf{resp})$
$\mathsf{Verify}^{\mathcal{O}}(\mathsf{lbl}, ((X_i)_{i\in[N]}, \mathsf{pk}, \mathsf{ct}), \pi)$
  1: $(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \leftarrow \pi$
  2: if $\mathsf{accept} \leftarrow V_2^{\mathcal{O}}(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \wedge \mathsf{ch} = \mathcal{O}(\mathsf{FS} \parallel \mathsf{lbl} \parallel ((X_i)_{i\in[N]}, \mathsf{pk}, \mathsf{ct}) \parallel \mathsf{com})$ then
  3:     return $\top$
  4: else
  5:     return $\perp$
```

Figure 5.5: A multi-proof online extractable NIZK with labels $\Pi_{\mathsf{NIZK,lbl}}$ for the relation R_{sig} obtained by applying the Fiat-Shamir transform to the traceable OR sigma protocol $\Pi_\Sigma^{\mathsf{tOR}} = (P = (P_1, P_2), V = (V_1, V_2))$ in Fig. 5.3 where $\mathcal{O}(\mathsf{FS} \parallel \cdot)$ outputs a binary string of length λ.

The following theorem shows that our construction in Fig. 5.5 is multi-proof online extractable. The proof is an adaptation of the proof of the original work [BDK+22] on removal of the uses of the seed trees and the unbalanced challenge space.

Theorem 5.3.9. *The* NIZK *with labels* $\Pi_{\mathsf{NIZK,lbl}}$ *in Fig. 5.5 is multi-proof online extractable for the family of relations* R_{sig} *and* $\tilde{R}_{\mathsf{sig}}$ *considered in Thm. 5.3.5,*

$$\tilde{R}_{\mathsf{sig}} = \left\{ \Big(((X_i)_{i\in[N]}, \mathsf{pk}, \mathsf{ct}), \mathsf{W}\Big) \;\middle|\; \begin{array}{c} \mathsf{W} = (I, s, r) \in [N] \times G \times G \\ \wedge\ X_I = s \star X_0 \wedge \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, I; r) \textit{ or} \\ \mathsf{W} = (x_1, x_2) \in \{0,1\}^* \wedge x_1 \neq x_2 \wedge \mathcal{O}(x_1) = \mathcal{O}(x_2) \end{array} \right\}.$$

More precisely, for any (possibly computationally-unbounded) adversary $\mathcal{A}$ making at most Q queries to the random oracle and T queries to the extract oracle, we have

$$\mathsf{Adv}^{\mathsf{mpOE}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{A}) \leq T \cdot (Q/2^{2\lambda-1} + (\lambda \cdot Q)/2^{\lambda} + 1/2^{\lambda}).$$

Proof. We begin the proof by providing the description of the online extractor $\mathsf{OnlineExtract}$. Below, it is given as input $(\mathsf{lbl}, \mathsf{X}, \pi, L_{\mathcal{O}})$, where π is guaranteed to be valid by definition and $L_{\mathcal{O}}$ is the oracle query list.

1. Parses $((X_i)_{i\in[N]}, \mathsf{pk}, \mathsf{ct}) \leftarrow \mathsf{X}$, $(\overline{\mathsf{com}}, \overline{\mathsf{chall}}, \overline{\mathsf{resp}}) \leftarrow \pi$, $((\mathsf{salt}, \mathsf{com}_1, \cdots, \mathsf{com}_\lambda), \mathbf{c} = (c_1, \cdots, c_\lambda)) \leftarrow (\overline{\mathsf{com}}, \overline{\mathsf{chall}})$, $(\mathsf{resp}_1, \cdots, \mathsf{resp}_\lambda) \leftarrow \overline{\mathsf{resp}}$, and $\mathsf{root}_j \leftarrow \mathsf{com}_j$ for $j \in [\lambda]$.[5]
2. For $j \in [\lambda]$ such that $c_j = 1$, it proceeds as follows:
   (a) Parse $(s''_j, r''_j, \mathsf{path}_j) \leftarrow \mathsf{resp}_j$.
   (b) For every $((\mathsf{salt}\|j\|\mathsf{Expand}\|\mathsf{seed}), (s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N)) \in L_{\mathcal{O}}$, where $\mathsf{salt}\|j\|\mathsf{Expand}$ is fixed, it proceeds as follows:
      i. Invoke the extractor in Thm. 5.3.5 on input $(\mathsf{com}_j, 0, \mathsf{seed}), (\mathsf{com}_j, 1, \mathsf{resp}_j)$ and obtain W.
      ii. Output W.
3. If it finds no witness W of the above form, then it returns $\mathsf{W} = \perp$.

We analyze the probability of $\mathcal{A}$ winning the multi-proof online extractability game with the above online extractor $\mathsf{OnlineExtract}$. Below, P' and V' are the prover and verifier of the base traceable OR sigma protocol $\Pi_\Sigma^{\mathsf{ARSbase}}$ in Fig. 5.2.

- We say a tuple $\mathsf{input}_{\mathsf{base}} = (\mathsf{X}, \mathsf{salt}, j, \mathsf{com}_j, c_j, \mathsf{resp}_j)$ is valid if the following three properties hold:
  - $c_j = 1$;
  - $V_2'^{\mathcal{O}(\mathsf{salt}\|j\|\cdot)}(\mathsf{com}_j, c_j, \mathsf{resp}_j)$ outputs accept (i.e. it is a valid transcript for $\Pi_\Sigma^{\mathsf{ARSbase}}$ with challenge 1);
  - there exists $(\mathsf{seed}, s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N)$ such that $((\mathsf{salt}\|j\|\mathsf{Expand}\|\mathsf{seed}), (s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N)) \in L_{\mathcal{O}}$, and if we execute $P_1'^{\mathcal{O}(\mathsf{salt}\|j\|\cdot)}$ with randomness seed, it produces com. (Here, we use the fact that $P_1'^{\mathcal{O}(\mathsf{salt}\|j\|\cdot)}$ can be executed without the witness.) By correctness of $\Pi_\Sigma^{\mathsf{ARSbase}}$, this implies that $(\mathsf{com}_j, 0, \mathsf{seed}_j)$ is a valid transcript.
- We say a tuple $\mathsf{input}_{\mathsf{base}} = (\mathsf{X}, \mathsf{salt}, j, \mathsf{com}_j, c_j, \mathsf{resp}_j)$ is $\mathsf{invalid}$ if $\mathsf{ch} = 1$, $V_2'^{\mathcal{O}(\mathsf{salt}\|j\|\cdot)}(\mathsf{com}_j, c_j, \mathsf{resp}_j)$ outputs accept, but it is not valid. Roughly speaking, the root com_j is generated without known valid response for the challenge 0 from the oracle query.

[5] Throughout the proof, we use overlines for $(\overline{\mathsf{com}}, \overline{\mathsf{chall}}, \overline{\mathsf{resp}})$ to indicate that it is a transcript of $\Pi_\Sigma^{\mathsf{tOR}}$. We use resp_i without overlines to indicate elements of $\overline{\mathsf{resp}}$.

Observe that if $\mathsf{input}_{\mathsf{base}}$ is valid, then the online extractor can recover a valid transcript $(\mathsf{com}_j, 0, \mathsf{seed}_j)$ from $\mathsf{input}_{\mathsf{base}}$. Then, it can (informally) extract a witness by combining it with $(\mathsf{com}_j, 1, \mathsf{resp}_j)$ and using the extractor from $\Pi_{\Sigma}^{\mathsf{ARSbase}}$ constructed in Thm. 5.3.5.

In contrast, if $\mathsf{input}_{\mathsf{base}}$ is $\mathsf{invalid}$, then intuitively, no adversary (even an unbounded one) would be able to prepare a valid response $\mathsf{resp}_j = \mathsf{seed}_j$ for the challenge $\mathsf{ch} = 0$ since the list random oracle list $L_{\mathcal{O}}$ does not contain a valid response. To make this claim formal, we need to also take into account the fact that the adversary may learn non-trivial information about $\mathsf{resp}_j = \mathsf{seed}_j$ from the proof via the prove query $P^{\mathcal{O}}$ instead of directly from the random oracle. In this case, even though no useful information is stored in $L_{\mathcal{O}}$, the adversary may still be able to forge a proof. We will show this is not feasible except for a negligible chance.

Lemma 5.3.10. *Assume an adversary $\mathcal{A}$ submits a total of T extract queries in the experiment of the form $\{(\mathsf{lbl}_k, \mathsf{X}_k, \pi_k)\}_{k\in[T]}$, where every π_k is a valid proof including the same* salt *and satisfies $(\mathsf{lbl}_k, \mathsf{X}_k, \pi_k) \notin L_P$. Let $\{(\mathsf{com}_{k,j}, \mathsf{ch}_{k,j}, \mathsf{resp}_{k,j})\}_{j\in[\lambda]}$ be the transcript of the base traceable OR sigma protocol $\Pi_{\Sigma}^{\mathsf{ARSbase}}$ that the verification algorithm reconstructs when verifying π_k. Then, with probability at least $1 - T \cdot (Q_{\mathsf{salt}}/2^{2\lambda-1} + (\lambda \cdot Q_{\mathsf{salt}})/2^{\lambda} + 1/2^{\lambda})$, for all $k \in T$ there exists at least one $j \in [\lambda]$ such that $\mathsf{input}_{\mathsf{base}} = (\mathsf{X}_k, \mathsf{salt}, j, \mathsf{com}_{k,j}, \mathsf{ch}_{k,j} = 1, \mathsf{resp}_{k,j})$ is* valid *where Q_{salt} represents the number of the random oracle queries of form $\mathcal{O}(\mathsf{salt} \,\|\, \cdot)$ made in the experiment.*

Proof. For any $k \in [T]$, let us redefine $\pi_k = (\overline{\mathsf{com}}, \overline{\mathsf{chall}}, \overline{\mathsf{resp}})$, and $(\overline{\mathsf{com}}, \overline{\mathsf{chall}}) = ((\mathsf{salt}, \mathsf{com}_1, \cdots, \mathsf{com}_\lambda), \mathbf{c} = (c_1, \cdots, c_\lambda))$, and $\overline{\mathsf{resp}} = (\mathsf{resp}_1, \cdots, \mathsf{resp}_\lambda)$ where $\mathbf{c} = \mathcal{O}(\mathsf{FS} \| \mathsf{lbl} \| \mathsf{X} \| \overline{\mathsf{com}})$. Namely, we omit the subscript k for better readability. We consider two cases: (1) there exists $(\mathsf{lbl}, \mathsf{X}, \pi') \in L_P$ such that $\pi' = (\overline{\mathsf{com}}, \overline{\mathsf{chall}}, \overline{\mathsf{resp}}')$ and $\overline{\mathsf{resp}}' \neq \overline{\mathsf{resp}}$ and (2) no such entry in L_P exists.

We consider the first case (1). This corresponds to the case where $\mathcal{A}$ reuses the proof π' from the prove query to generate π by simply modifying the response. We claim that this cannot happen with overwhelming probability.

Let $\overline{\mathsf{resp}}' = (\mathsf{resp}'_1, \cdots, \mathsf{resp}'_\lambda)$. Due to the regular action and Rem. 5.3.2, the event occurs only if there is a collision of $\mathcal{O}(\cdot)$. Concretely, consider $\mathsf{resp}'_j \neq \mathsf{resp}_j$ for some $j \in [\lambda]$ such that $c_j = 1$. Then, it either finds a collision for $\mathcal{O}(\mathsf{Coll} \,\|\, \cdot)$ (used by the Merkle tree) or for $\mathcal{O}(\mathsf{Com} \,\|\, \cdot)$ (to generate the leaf). Therefore, case (1) occurs with probability at most $Q_{\mathsf{salt}}/2^{2\lambda-1}$.

We next consider the second case (2): there exists no π' in L_P that contains the same $\overline{\mathsf{com}}$ as π. This, in particular, implies that the output $\overline{\mathsf{chall}} \leftarrow \mathcal{O}(\mathsf{FS} \| \mathsf{lbl} \| \mathsf{X} \,\|\, \overline{\mathsf{com}})$ is distributed uniform random from the view of $\mathcal{A}$ before it makes the hash query.

Now, we suppose, for the purpose of contradiction, that $\mathsf{input}_{\mathsf{base},j} = (\mathsf{X}, \mathsf{salt}, j, \mathsf{com}_j, c_j, \mathsf{resp}_j)$ is $\mathsf{invalid}$ for all $j \in [\lambda]$ such that $c_j = 1$. Let $L_{\mathcal{O}_P}$ be a list that contains all the inputs/outputs of the random oracle queries $\mathsf{Prove}^{\mathcal{O}}$ makes when the challenger answers the prove query made by $\mathcal{A}$. We prove the following sub-lemma.

Lemma 5.3.11. *For any $j^* \in [\lambda]$, if $\mathsf{input}_{\mathsf{base},j^*}$ is* $\mathsf{invalid}$*, then either of the following holds:*

- *there exists no tuple $(s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N, \mathsf{seed})$ and $j' \in [\lambda]$ such that $((\mathsf{salt} \,\|\, j' \,\|\, \mathsf{Expand} \,\|\, \mathsf{seed}), (s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N)) \in L_{\mathcal{O}_P}$, but if we execute $P_1'^{\mathcal{O}(\mathsf{salt} \| j' \| \cdot)}$ with randomness* seed*, it produces com_{j^*};*

- *there exists such a tuple but* seed *retains* λ*-bits of min-entropy from the view of* $\mathcal{A}$ *except with probability at most* $(MQ_{\mathsf{salt}})/2^{\lambda}$.

Proof. Assume such an entry is found in $L_{\mathcal{O}_P}$. This corresponds to the case $\mathcal{A}$ is reusing com_{j^*} that was included in a proof π obtained through the prove query. Let $\{(\mathsf{com}'_j, c'_j, \mathsf{resp}'_j)\}_{j\in[\lambda]}$ be the transcript of the base traceable OR sigma protocol $\Pi_{\Sigma}^{\mathsf{ARSbase}}$ that the verification algorithm reconstructs from such π (see Line 5 of **Verification** $V_2^{\mathcal{O}}$ in Fig. 5.2) such that $\mathsf{com}'_{j'} = \mathsf{com}_{j^*}$ for some $j' \in [\lambda]$. Claim that $c'_{j'} = 1$ (i.e. seed was not used as a response). Since $\mathsf{com}'_{j'}$ and com_{j^*} are roots of a Merkle tree and the indices j' and j^* are used as prefix to the hash when constructing the roots, respectively, the equality implies a collision of $\mathcal{O}(\mathsf{Coll} \parallel \cdot)$. Hence, the probability of $\mathcal{A}$ outputting com_{j^*} such that $j' \neq j^*$ is upper bounded by $((\lambda - 1)Q_{\mathsf{salt}})/2^{\lambda}$.

We may thereby assume $j' = j^*$. Recall by definition of the online extractability game (see Def. 3.4.5), $\mathcal{A}$ runs the verification algorithm to check if π is valid. Therefore, if $\mathsf{input}_{\mathsf{base},j^*}$ is $\mathsf{invalid}$, then we have $c'_{j'} = 1$. Otherwise, there must already exist an entry $((\mathsf{salt} \parallel j^* \parallel \mathsf{Expand} \parallel \mathsf{seed}), (s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N)) \in L_{\mathcal{O}}$ (to run the verification algorithm), which contradicts that $\mathsf{input}_{\mathsf{base},j^*}$ is $\mathsf{invalid}$. This further implies that $\mathsf{resp}'_{j'}$ does not include seed. Therefore, seed that was used to construct $\mathsf{com}_{j'} = \mathsf{com}_{j^*}$ is statistically hidden to the adversary with all but probability $Q_{\mathsf{salt}}/2^{\lambda}$. By collecting all the bounds, the result of Lem. 5.3.11 follows. □

By Lem. 5.3.11, if $\mathsf{input}_{\mathsf{base},j}$ is $\mathsf{invalid}$, then $\mathcal{A}$ cannot prepare a valid response for the challenge $c_j = 0$ with all but probability at most $(\lambda Q_{\mathsf{salt}})/2^{\lambda}$. This is because such response is either not recorded in both $L_{\mathcal{O}}$ and $L_{\mathcal{O}_P}$, or it is recorded in $L_{\mathcal{O}_P}$ but the seed retains λ-bits of min-entropy from the view of $\mathcal{A}$ except with probability $(\lambda Q_{\mathsf{salt}})/2^{\lambda}$. Moreover, since $\overline{\mathsf{chall}}$ is statistically hidden to $\mathcal{A}$ before it queries the random oracle, the probability that $\overline{\mathsf{chall}}$ coincides with challenges for which $\mathcal{A}$ can open to is at most $1 - 1/2^{\lambda}$.

Taking the union bound and collecting all the bounds together, at least one of the $\mathsf{input}_{\mathsf{base}}$ must be valid with the probability stated in the statement. This completes the proof of Lem. 5.3.10. □

We are now prepared to analyze the probability that $\mathcal{A}$ wins the multi-proof online extractability game with the aforementioned online extractor $\mathsf{OnlineExtract}$. By Lem. 5.3.10, if $\mathcal{A}$ makes at most T extract queries, then by a simple union bound and using the inequality $\sum_i Q_{\mathsf{salt}_i} \leq Q$ where Q_{salt_i} represents the number of random oracle queries made by $\mathcal{A}$ of form $\mathcal{O}(\mathsf{salt}_i \parallel \cdot)$. With probability at least $1 - T \cdot ((2Q)/2^{2\lambda} + (\lambda \cdot Q)/2^{\lambda} + 1/2^{\lambda})$, all the $\mathsf{input}_{\mathsf{base}}$ included in the queried proof are valid. By collecting the bounds, we complete the proof of Thm. 5.3.9. □

Remark 5.3.12 (mpOE for non-relaxed R_{sig})**.** *In fact, one can also modify the proof above to extract the witness for* R_{sig} *instead of the relaxed one* $\tilde{R}_{\mathsf{sig}}$ *(i.e. a collision in* $\mathcal{O}(\mathsf{salt} \parallel j \parallel \mathsf{Coll} \parallel \cdot)$ *or* $\mathcal{O}(\mathsf{salt} \parallel j \parallel \mathsf{Com} \parallel \cdot)$ *for some* $j \in [\lambda]$*). Concretely, we can modify Item 2b in the proof above to extract for a witness for* R_{sig} *only. The only change in the analysis is to exclude the probability of having a collision among the random oracle queries. That is, with all but probability* $Q^2/2^{2\lambda}$,

$\mathsf{OnlineExtract}$ *succeeds in extracting a witness* $\mathsf{W} = (I, s, r)$ *as desired, conditioned on all the* $\mathsf{input_{base}}$ *included in the queried proof are* valid*. The bound will be*

$$\mathsf{Adv}^{\mathsf{mpOE}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{A}) \leq T \cdot (Q^2/2^{2\lambda-2} + (\lambda \cdot Q)/2^{\lambda} + 1/2^{\lambda}).$$

Theorem 5.3.13. *The* NIZK *with labels* $\Pi_{\mathsf{NIZK,lbl}}$ *in Fig. 5.5 is zero-knowledge. Precisely, there exists a PPT simulator* $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ *such that, for any statement-witness pair* $(\mathsf{X}, \mathsf{W}) \in R_{\mathsf{sig}}$ *and any computationally-unbounded adversary* $\mathcal{A}$ *that makes at most* Q_1 *queries to* $\mathcal{O}$ *or* Sim_0*, and* Q_2 *queries to* Prove *or* $\mathcal{S}$*, we have*

$$\mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{A}) = \left|\Pr\left[\mathcal{A}^{\mathcal{O},\mathsf{Prove}}(1^{\lambda}) = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{Sim}_0,\mathcal{S}}(1^{\lambda}) = 1\right]\right| \leq \frac{Q_2 \cdot (Q_1 + Q_2)}{2^{2\lambda}} + \frac{Q_1}{2^{\lambda}}.$$

Proof. A zero-knowledge simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$[6] proceeds as in Fig. 5.6, where Sim_0 and Sim_1 share states, including a list L which is initially empty. At a high level, the first subroutine Sim_0 simulates the random oracle $\mathcal{O}$ in an on-the-fly manner except for specific queries for the purpose of consistency with Sim_1. On the other hand, Sim_1 simulates the prover oracle using the simulator from abovementioned sigma protocol (Fig. 5.3) in which the simlator is denoted here by Sim_{Σ} (see Thm. 5.3.8) as a subroutine.

Specifically, Sim_1 is given a valid statement $\mathsf{X} = ((X_i)_{i \in [N]}, \mathsf{pk}, \mathsf{ct})$, and samples a random challenge ch from the challenge space $\{0,1\}^{\lambda}$, which is also the domain of the Fiat-Shamir transform $\mathcal{O}(\mathsf{FS} \parallel \cdot)$. Then, Sim_1 executes Sim_{Σ} on challenge ch using simulated oracle access to Sim_0, and updates the list L accordingly. In Fig. 5.6, we denote by D_x the distribution of $\mathcal{O}(x)$, where the probability is taken over the random choice of the random oracle $\mathcal{O}$. We may therefore assume D_x to be efficiently sampleable.

```
$\mathsf{Sim}_0(x)$
1: if $x \in L$ then
2:     return $L[x]$
3: $y \leftarrow D_x$
4: $L[x] := y$
5: return $y$
```

```
$\mathsf{Sim}_1(\mathsf{lbl}, \mathsf{X})$
1: $\mathsf{ch} \leftarrow \{0,1\}^{\lambda}$
2: $(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \leftarrow \mathsf{Sim}_{\Sigma}^{\mathsf{Sim}_0}(\mathsf{X}, \mathsf{ch})$
3: if $(\mathsf{FS} \parallel \mathsf{lbl} \parallel \mathsf{X} \parallel \mathsf{com}) \in L$ then
4:     return $\perp$
5: $L[(\mathsf{FS} \parallel \mathsf{lbl} \parallel \mathsf{X} \parallel \mathsf{com})] := \mathsf{ch}$
6: return $\pi \leftarrow (\mathsf{com}, \mathsf{ch}, \mathsf{resp})$
```

Figure 5.6: Zero-knowledge simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ for $\Pi_{\mathsf{NIZK,lbl}}$

To show the indistinguishability of $(\mathcal{O}, \mathsf{Prove})$ and $(\mathsf{Sim}_0, \mathcal{S})$, we use a hybrid argument by introducing an intermediate pair of simulators $(\mathsf{Sim}_0, \mathsf{Sim}_{\mathsf{int}})$, where $\mathsf{Sim}_{\mathsf{int}}$ is defined in Fig. 5.7. Intuitively, $\mathcal{S}_{\mathsf{int}}$ is a real prover with access to the simulated oracle Sim_0.

Suppose $\mathcal{A}$ makes Q_1 queries to the oracles $\mathcal{O}$ or Sim_0, and Q_2 queries to the oracles $\mathsf{Prove}, \mathcal{S}_{\mathsf{int}}$, or $\mathcal{S}$. For each $i \in \{1,2,3\}$, we denote by E_i the event that $\mathcal{A}$ returns 1 respectively. We analyze the differences by defining three games as follows:

[6]Remark that the notations $(\mathsf{Sim}_0, \mathsf{Sim}_1)$ and $(\mathsf{Sim}_0, \mathcal{S})$ are not typos. The former is the simulator we need to construct. Meanwhile, the latter is the interface corresponding to $(\mathcal{O}, \mathsf{Prove})$. See Def. 3.4.3.

```
$\mathsf{Sim_{int}}(\mathsf{lbl}, \mathsf{X}, \mathsf{W})$
1: $\mathsf{com} \leftarrow P_1^{\mathsf{Sim_0}}(\mathsf{X}, \mathsf{W})$
2: $\mathsf{ch} \leftarrow \{0,1\}^\lambda$
3: if $(\mathsf{FS} \| \mathsf{lbl} \| \mathsf{X} \| \mathsf{com}) \in L$ then
4:     return $\perp$
5: $L[(\mathsf{FS} \| \mathsf{lbl} \| \mathsf{X} \| \mathsf{com})] := \mathsf{ch}$
6: $\mathsf{resp} \leftarrow P_2^{\mathsf{Sim_0}}(\mathsf{X}, \mathsf{ch})$
7: return $\pi \leftarrow (\mathsf{com}, \mathsf{ch}, \mathsf{resp})$
```

Figure 5.7: Intermediate simulator $\mathsf{Sim_{int}}$, where $P = (P_1, P_2)$ is the prover of the traceable OR sigma protocol $\Pi_\Sigma^{\mathsf{tOR}}$ in Fig. 5.3.

Game_1 : This is the real zero-knowledge game where $\mathcal{A}$ is given access to $\mathcal{O}$ and Prove.

Game_2 : The game is modified to provide $\mathcal{A}$ access to Sim_0 and $\mathcal{S}_{\mathsf{int}}$ instead. The view of $\mathcal{A}$ is identical to the previous game unless $\mathsf{Sim_{int}}$ outputs $\perp$ in Line 4. Roughly, this occurs when the reprogramming of the random oracle fails due to the input being already defined. By Thm. 5.3.7, com has 2λ bits of min-entropy. Since at most $Q_1 + Q_2$ queries of the form $(\mathsf{FS}\|\mathsf{lbl}\|\mathsf{X}\|\mathsf{com})$ are made in this game, we have $|\Pr[E_1] - \Pr[E_2]| \leq \frac{Q_2 \cdot (Q_1+Q_2)}{2^{2\lambda}}$.

Game_3 : The game is modified to provide $\mathcal{A}$ access to Sim_0 and $\mathcal{S}$ instead. The only difference is that rather than computing honestly via (P_1, P_2) from the underlying sigma protocol $\Pi_\Sigma^{\mathsf{tOR}}$ (Fig. 5.3), the simulator Sim_1 simulates these using the simulator Sim_Σ provided by $\Pi_\Sigma^{\mathsf{tOR}}$.

Let salt_i represent the salt that $\mathsf{Sim_{int}}$ or Sim_1 samples on its i-th invocation. For $i \in [Q_2]$, let Q_i' be the number of queries the adversary makes to oracle Sim_0 of the form $(\mathsf{salt}_i\|\cdot)$. By Thm. 5.3.8, the advantage of the adversary in distinguishing $\mathsf{Sim_{int}}$ or Sim_1 is bounded by $\frac{Q_i'}{2^\lambda}$ for each $i \in [Q_2]$. Therefore, $|\Pr[E_2] - \Pr[E_3]| \leq \frac{\sum_1^{Q_2} Q_i'}{2^\lambda} \leq \frac{Q_1}{2^\lambda}$

By collecting the bounds, the result follows. □

Remark 5.3.14 (NIZK for $\tilde{R}_{\mathsf{open}}$). *By applying the same transform (Fig. 5.5) to the base sigma protocol Fig. 5.1 for the opening relations R_{open} and $\tilde{R}_{\mathsf{open}}$, we can have a complete, zero-knowledge, and multi-proof online-extractable (therefore statistical sound Rems. 3.4.7 and 3.4.8). We can therefore skip the proofs here.*

5.4 Construction

5.4.1 Accountable Ring Signature

We are now ready to construct an accountable ring signature based on the abovedmentioned components:

- A public parameter $\mathsf{pp} = (G, \mathcal{E}, E_0, \star, \mathcal{M})$, as in Sec. 5.3.1, where $(G, \mathcal{E}, E_0, \star)$ is the public parameter specified in Def. 2.3.3, $\mathcal{M}$ is the message space which is a subset of G and $|\mathcal{M}| \leq poly(\lambda)$.
- A group-action-based IND-CPA public key encryption $\Pi_{\mathsf{PKE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ described in Sec. 5.3.1, which can be derived from $(G, \mathcal{E}, E_0, \star)$.
- Sound NIZK $\Pi_{\mathsf{NIZK}} = (\mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ without labels for R_{open} and $\tilde{R}_{\mathsf{open}}$ in (based on $\Pi_{\Sigma}^{\mathsf{OPbase}}$ in Fig. 5.1) where

$$R_{\mathsf{open}} = \left\{ ((\mathsf{pk} = (E_0, E), \mathsf{ct}, I), \mathsf{sk}) \;\middle|\; \begin{array}{c} E = \mathsf{sk} \star E_0 \;\wedge \\ I = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \end{array} \right\},$$

$$\tilde{R}_{\mathsf{open}} = \left\{ \Big(((\mathsf{pk} = (E_0, E), \mathsf{ct}, I), \mathsf{W}) \Big) \;\middle|\; \begin{array}{c} \mathsf{W} = \mathsf{sk} \in G \;\wedge \\ E = \mathsf{sk} \star E_0 \;\; \wedge I = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \text{ or} \\ \mathsf{W} = (x_1, x_2) \in \{0,1\}^* \wedge \\ x_1 \neq x_2 \wedge \mathcal{O}(x_1) = \mathcal{O}(x_2) \end{array} \right\}.$$

- Online-extractable NIZK $\Pi_{\mathsf{NIZK,lbl}} = (\mathsf{NIZK.Prove_{lbl}}, \mathsf{NIZK.Verify_{lbl}})$ with labels for R_{sig} and $\tilde{R}_{\mathsf{sig}}$ in Fig. 5.5 (based on $\Pi_{\Sigma}^{ARSbase}$ in Fig. 5.2).

$$R_{\mathsf{sig}} = \left\{ \Big(\big((X_i)_{i\in[N]}, \mathsf{pk}, \mathsf{ct} \big), (I, s, r) \Big) \;\middle|\; \begin{array}{c} (I, s, r) \in [N] \times G \times G \\ X_I = s \star E_0 \;\wedge \\ \mathsf{ct} = (Y_0, Y) = \mathsf{Enc}(\mathsf{pk}, I; r) \end{array} \right\}$$

$$\tilde{R}_{\mathsf{sig}} = \left\{ \Big(((X_i)_{i\in[N]}, \mathsf{pk}, \mathsf{ct}), \mathsf{W} \Big) \;\middle|\; \begin{array}{c} \mathsf{W} = (I, s, r) \in [N] \times G \times G \\ \wedge\; X_I = s \star X_0 \wedge \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, I; r) \text{ or} \\ \mathsf{W} = (x_1, x_2) \in \{0,1\}^* \wedge \\ x_1 \neq x_2 \wedge \mathcal{O}(x_1) = \mathcal{O}(x_2) \end{array} \right\}.$$

Accountable Ring Signature. The high-level idea of an accountable ring signature based on the above ingredients is as follows. First, each member generates a signing-and-verification key pair, a R_{GAIP} instance. Also, an opener generates an encryption-and-decryption key pair from Π_{PKE}, also a R_{GAIP} instance. Collecting key pairs forms a ring of members, which can be updated dynamically after adding an identifier. Next, each member of the ring can specify an opener's public key and signs a message using $\Pi_{\mathsf{NIZK,lbl}}$. Recall that the signature is derived from a "traceable" OR proof, which allows the encryption key's owner (called the opener) to decrypt and open the signature. After obtaining the result, the opener uses Π_{NIZK} to prove the correctness of the opening result.

Throughout the following theorems, we let $\mathsf{Adv}_{\mathcal{O}}^{\mathsf{Coll}}(\mathcal{B})$ denote the advantage of an adversary $\mathcal{B}$ producing a collision of the random oracle $\mathcal{O}$.

Theorem 5.4.1. *The accountable ring signature scheme Π_{ARS} in Fig. 5.8 is correct.*

Proof. Due to the completeness of $\Pi_{\mathsf{NIZK,lbl}}$, any signature output by $\mathsf{ARS.Sign}$ will be accepted by $\mathsf{ARS.Verify}$ with probability 1. □

```
ARS.UKGen(1^λ)
  1: $\mathsf{sk} \leftarrow G$
  2: $\mathsf{vk} = \mathsf{sk} \star E_0$
  3: return $(\mathsf{vk}, \mathsf{sk})$
```

```
ARS.MKGen(1^λ)
  1: $\mathsf{msk} \leftarrow G$
  2: $\mathsf{mpk} \leftarrow \mathsf{msk} \star E_0$
  3: return $(\mathsf{mpk}, \mathsf{msk})$
```

```
ARS.Sign(mpk, sk, R, M)
  1: $(\mathsf{vk}_i)_{i\in[N]} \leftarrow \mathsf{R}$
  2: if $\nexists I : (\mathsf{vk}_I, \mathsf{sk}) \in R_{\mathsf{GAIP}}$ then
  3:     return $\perp$.
  4: $r \xleftarrow{\$} G$
  5: $\mathsf{ct} = \mathsf{Enc}(\mathsf{mpk}, I; r)$
  6: $\pi_{\mathtt{sign}} \leftarrow \mathsf{NIZK.Prove}_{\mathsf{lbl}}(\mathsf{M}, (\mathsf{R}, \mathsf{mpk}, \mathsf{ct}), (I, \mathsf{sk}, r))$
  7: return $\sigma := (\mathsf{ct}, \pi_{\mathtt{sign}})$
```

```
ARS.Verify(mpk, R, M, σ)
  1: $(\mathsf{ct}, \pi_{\mathtt{sign}}) \leftarrow \sigma$
  2: return $\mathsf{NIZK.Verify}_{\mathsf{lbl}}(\mathsf{M}, (\mathsf{R}, \mathsf{mpk}, \mathsf{ct}), \pi_{\mathtt{sign}})$
```

```
ARS.Judge(mpk, R, vk, M, σ, π_open)
  1: $(\mathsf{ct}, \pi_{\mathtt{sign}}) \leftarrow \sigma$
  2: if $\nexists I : \mathsf{vk} = \mathsf{R}_I$ then
  3:     return $\perp$.
  4: $b_0 \leftarrow \mathsf{ARS.Verify}(\mathsf{mpk}, \mathsf{R}, \mathsf{M}, \sigma)$
  5: $b_1 \leftarrow \mathsf{NIZK.Verify}((\mathsf{mpk}, \mathsf{ct}, I), \pi_{\mathtt{open}})$
  6: return $b_0 \wedge b_1$
```

```
ARS.Open(msk, R, M, σ)
  1: if $\mathsf{ARS.Verify}(\mathsf{mpk}, \mathsf{R}, \mathsf{M}, \sigma) = \perp$ then
  2:     return $\perp$
  3: $(\mathsf{ct}, \pi_{\mathtt{sign}}) \leftarrow \sigma$
  4: $I \leftarrow \mathsf{Dec}(\mathsf{msk}, \mathsf{ct})$
  5: $\pi_{\mathtt{open}} \leftarrow \mathsf{NIZK.Prove}((\mathsf{mpk}, \mathsf{ct}, I), \mathsf{msk})$
  6: return $\pi := (\mathsf{R}_I, \pi_{\mathtt{open}})$
```

Figure 5.8: Our construction of an accountable ring signature Π_{ARS}. The public parameter $\mathsf{pp} = (G, \mathcal{E}, E_0, \star, \mathcal{M})$ as in Sec. 5.3.1, is used in every algorithm, where $(G, \mathcal{E}, E_0, \star)$ is the public parameter specified in Def. 2.3.3, $\mathcal{M}$ is the message space which is a subset of G and $|\mathcal{M}| \leq poly(\lambda)$. The construction consists of a group action-based public-key encryption algorithm $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ using pp, a NIZK with labels $\Pi_{\mathsf{NIZK,lbl}}$ for the relation $R_{\mathsf{sig}}, \tilde{R}_{\mathsf{sig}}$, and a NIZK without labels Π_{NIZK} for the opening relation R_{open}. The openers' public-key space and users' verification-key space $\mathcal{K}_{\mathsf{mpk}} = \mathcal{K}_{\mathsf{vk}} = \mathcal{E}$.

Theorem 5.4.2. *The accountable ring signature scheme Π_{ARS} in Fig. 5.8 is (CCA) anonymous (against full key exposure) in the random oracle model, assuming Π_{PKE} is multi-challenge* IND-CPA *secure, $\Pi_{\mathsf{NIZK,lbl}}$ is zero-knowledge, multi-challenge online-extractable, and Π_{NIZK} is zero-knowledge. Precisely, for an adversary $\mathcal{A}$, running in time T, there exist PPT adversaries $\mathcal{B}_1, \mathcal{B}_1', \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$, with running time* $\mathsf{poly}(\lambda)T$ *such that*

$$\mathsf{Adv}^{\mathsf{Anon}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{mpOE}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{Coll}}_{\mathcal{O}}(\mathcal{B}_1') + \mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{B}_2) + \mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_3) + \mathsf{Adv}^{\mathsf{Multi\text{-}CPA}}_{\Pi_{\mathsf{PKE}}}(\mathcal{B}_4)\,.$$

Proof. We prove anonymity using a hybrid argument by introducing a series of games $\mathsf{Game}_1, \cdots,$ Game_5 and turning an anonymity adversary into an IND-CPA adversary in the last game. Let the advantage of the adversary $\mathcal{A}$ in Game_i be denoted by $\mathsf{Adv}_i(\mathcal{A})$.

Game_1 : This is the original anonymity experiment defined in Def. 5.2.3. The adversary's advantage in this game is $\mathsf{Adv}_1(\mathcal{A}) = \mathsf{Adv}^{\mathsf{Anon}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A})$ by definition.

Game_2 : This is the same as Game_1, except that the way the challenger answers opening queries is modified. Rather than using the secret key msk to decrypt the ciphertext ct and identify the index I of the real signing key (i.e. following $\mathsf{ARS.Open}$), the challenger instead runs the online extractor $\mathsf{OnlineExtract}$ for $\Pi_{\mathsf{NIZK,lbl}}$ to extract the witness (I, sk, r) from $(\mathsf{ct}, \pi_{\mathtt{sign}})$, and then returns the user R_I. (The challenger still uses msk to generate the opening proof.)

The modified game is identical to Game_2 for $\mathcal{A}$ unless the extraction changes the adversary's view. Write the corresponding statement (to be signed) to be X. Remark that whenever the extractor outputs W such that $(\mathsf{X}, \mathsf{W}) \in R_{\mathsf{sig}}$, the modification does not change the view due to the uniqueness of the encryption Rem. 5.3.2.

That is, the modification changes the view of $\mathcal{A}$ only if the extractor outputs W such that either (X, W) is not in $\tilde{R}_{\mathsf{sig}}$ or W in $\tilde{R}_{\mathsf{sig}} - R_{\mathsf{sig}}$ (i.e. a collision of $\mathcal{O}$, recall Thm. 5.3.5). We can therefore turn $\mathcal{A}$ into an adversary $\mathcal{B}_1$ against the multi-challenge online-extractability game of Π_{NIZK} (Def. 3.4.5) with a subroutine $\mathcal{B}'_1$ finding a collision of the random oracle. Concretely, $\mathcal{B}_1$ simulates Game_2 for $\mathcal{A}$ on input mpk where $\mathsf{mpk} \leftarrow \mathsf{MKGen}$ such that

- random-oracle queries from $\mathcal{A}$ are answered by querying $(\mathtt{hash}, \cdot)$ (see Def. 3.4.5);
- the ring in the relation $\tilde{R}_{\mathsf{sig}}$ of the multi-proof online extractability experiment is determined by the ring R chosen by $\mathcal{A}$ on the signing queries $(\mathtt{sign}, \mathsf{R}, -, -, -)$,
- instead of computing $\pi_{\mathtt{sign}}$, $\mathcal{B}_1$ makes a query $(\mathtt{prove}, \mathsf{M}, \mathsf{pk}, \mathsf{w})$ for the signing queries $(\mathtt{sign}, \mathsf{R}, \mathsf{M}, \mathsf{sk}_0, \mathsf{sk}_1)$, where $(\mathsf{pk}, \mathsf{w}) = ((\mathsf{R}, \mathsf{mpk}, \mathsf{ct}), (I, \mathsf{sk}, r))$, sk is randomly chosen from $\mathsf{sk}_0, \mathsf{sk}_1$, and I is determined thereby, and
- instead of running $\mathsf{OnlineExtract}$, $\mathcal{B}_1$ makes a query $(\mathtt{extract}, \mathsf{M}, \mathsf{pk}, \pi_{\mathtt{sign}})$ for the open queries;
- the subroutine $\mathcal{B}'_1$ only collects the output of the extraction oracle forwarded from $\mathcal{B}_1$;
- $\mathcal{B}_1$ outputs 1 to the online-extractability challenger to represent the termination of the experiment when $\mathcal{A}$ terminates;
- If there exists a collision of the random oracle (i.e. an element in $\tilde{R}_{\mathsf{sig}} - R_{\mathsf{sig}}$) output by the extraction oracle, $\mathcal{B}'_1$ outputs the collision. Otherwise, $\mathcal{B}'_1$ outputs $\perp$.

Note that $\mathtt{extract}$ for proofs originating from $\mathtt{prove}$ queries are answered with $\perp$, which is compatible with the fact that the challenger outputs $\perp$ for opening queries that correspond to signatures originating from the signing oracle in Game_2. If $\mathcal{B}_1$ loses the game (i.e., $\mathcal{B}_1$ did not cause the extractor to fail) and $\mathcal{B}'_1$ outputs $\perp$, then the view of $\mathcal{A}$ remains the same. Therefore, we have $\mathsf{Adv}_1(\mathcal{A}) \leq \mathsf{Adv}_2(\mathcal{A}) + \mathsf{Adv}^{\mathsf{mpOE}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{Coll}}_{\mathcal{O}}(\mathcal{B}'_1)$.

Game_3 : This is the same as Game_2, except that we change the way the challenger responds to the random oracle queries and generate the proof for the opening queries from the adversary, which are replaced by the simulator $\mathsf{NIZK.Sim} = (\mathsf{NIZK.Sim}_0, \mathsf{NIZK.Sim}_1)$ for

Π_{NIZK}. Concretely, upon receiving a random oracle query from $\mathcal{A}$, the challenger forwards the query to the simulator $\mathsf{NIZK.Sim}_0$, records the query-answer pair, and forwards the answer to $\mathcal{A}$. Similarly, upon receiving a signing query from $\mathcal{A}$, the challenger does not compute π_{open} using $\mathsf{NIZK.Prove}$ and msk. Instead, the challenger executes $\mathsf{NIZK.Sim}_1$ and forwards the output to $\mathcal{A}$.

The modification results in a negligible loss for $\mathcal{A}$ unless the adversary $\mathcal{A}$ can distinguish the simulation of Π_{NIZK} with a non-negligible chance. Concretely, we can turn $\mathcal{A}$ into an adversary $\mathcal{B}_2$ against the zero-knowledge experiment of Π_{NIZK} (Def. 3.4.3) where $\mathcal{B}_2$ simulates Game_2 or Game_3 for $\mathcal{A}$ and outputs whatever $\mathcal{A}$ returns. Precisely, $\mathcal{B}_2$ forwards either the real prover response or the simulated response (from $\mathsf{NIZK.Sim} = (\mathsf{NIZK.Sim}_0, \mathsf{NIZK.Sim}_1)$) from the ZK experiment to $\mathcal{A}$ for the random-oracle or signing queries. The adversary $\mathcal{B}_2$ simulates Game_2 or Game_3 for $\mathcal{A}$ for the former or the latter case respectively. Therefore, we have $\mathsf{Adv}_2(\mathcal{A}) \leq \mathsf{Adv}_3(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{B}_2)$.

Game_4 : This is the same as Game_3, except that we change the way the challenger responds to the signing queries from the adversary. Instead of using $\mathsf{NIZK.Prove}_{\mathsf{lbl}}$ in $\mathsf{ARS.Sign}$, the challenger generates ct as in Game_3, but uses the zero-knowledge simulator Sim for $\Pi_{\mathsf{NIZK,lbl}}$ to generate the proof $\pi_{\mathtt{sign}}$. It then outputs $(\mathsf{ct}, \pi_{\mathtt{sign}})$ as the signature. Similar to the previous transition, we can define an adversary $\mathcal{B}_3$ against the zero-knowledge property of $\Pi_{\mathsf{NIZK,lbl}}$ such that $\mathsf{Adv}_3(\mathcal{A}) \leq \mathsf{Adv}_4(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_3)$.

Game_5 : This is the same as Game_4, except we change the way the challenger responds to the signing queries further: Instead of encrypting the correct index I to obtain ct, the challenger encrypts a random index I'.

Observe that the secret bit b of the anonymity experiment is not used in Game_5, which leads to $\mathsf{Adv}_5(\mathcal{A}) = 0$. The modification results in a loss for the adversary if $\mathcal{A}$ can distinguish the ciphertexts of different messages of Π_{PKE}. Concretely, we turn $\mathcal{A}$ into $\mathcal{B}_4$ against multi-challenge $\mathsf{IND\text{-}CPA}$ for Π_{PKE} where $\mathcal{B}_4$ simulates either Game_4 or Game_5 for $\mathcal{A}$. Instead of generating a key pair from KeyGen, the adversary $\mathcal{B}_4$ receives mpk from the multi-challenge $\mathsf{IND\text{-}CPA}$ challenger. Also, instead of producing the ciphertexts ct itself $\mathcal{B}_4$ makes encryption queries (I, I'), where I is the correct index, and I' is a random index. Remark that $\mathcal{B}_4$ does not use msk to simulate Game_4 or Game_5 due to the removal of the use of msk made in Game_1 to Game_3.

If the hidden bit b in the $\mathsf{IND\text{-}CPA}$ game is 0, then the experiment is identical to Game_4, and if the bit is 1, then the experiment is equal to Game_5. Therefore, we have that $\mathsf{Adv}_4(\mathcal{A}) \leq \mathsf{Adv}_5(\mathcal{A}) + \mathsf{Adv}^{\mathsf{Multi\text{-}CPA}}_{\Pi_{\mathsf{PKE}}}(\mathcal{B}_4)$.

By collecting the bounds, the result follows. □

Remark 5.4.3. *In the previous proof we really relied on the online extractability property (without rewinding). This is because, even if we allow for a non-tight reduction, we cannot resort to rewinding (i.e., the forking lemma) since there can be polynomially many open queries and* *the reduction loss will be exponential if we try to extract from all of them. Here, keep in mind that the online extractor must succeed with (roughly)* $1 - \mathsf{negl}(\lambda)$ *rather than any non-negligible function* $1/\mathsf{poly}(\lambda)$ *since there can be polynomially many open queries. Namely, even a success probability of* $1/2$ *will not be good enough. Most, if not all, prior works circumvent this issue by using an IND-CCA* PKE *as building block rather than a (possibly inefficient) online extractable* NIZK *to simulate the decryption of* ct.

Theorem 5.4.4. *The accountable ring signature scheme* Π_{ARS} *in Fig. 5.8 is unforgeable in the random oracle model. More precisely, for any adversary* $\mathcal{A}$ *that runs in time* T *and makes* $\mathcal{Q}_u$ *queries to the* `ukeygen` *oracle, there exist adversaries* $\mathcal{B}_1, \mathcal{B}_1', \mathcal{B}_2, \mathcal{B}_3$*, running in time* $\mathsf{poly}(\lambda)T$*, such that*

$$\mathsf{Adv}^{\mathsf{Unf}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{mpOE}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{Coll}}_{\mathcal{O}}(\mathcal{B}_1') + \mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_2) + \mathcal{Q}_u \mathsf{Adv}^{\mathsf{GAIP}}(\mathcal{B}_3)$$

Proof. We prove unforgeability using a hybrid argument with the following $\mathsf{Game}_1, \cdots, \mathsf{Game}_4$ and then turn the adversary into a GAIP adversary. Let the advantage of the adversary $\mathcal{A}$ in Game_i be denoted by $\mathsf{Adv}_i(\mathcal{A})$.

Game_1 : This is the original unforgeability game defined in Def. 5.2.4. The adversary's advantage in this game is $\mathsf{Adv}_1(\mathcal{A}) = \mathsf{Adv}^{\mathsf{Unf}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A})$ by definition.

Game_2 : This is the same as Game_1, but the winning condition is changed. We let the challenger maintain a list $L_{\mathcal{O}}$ of all the random oracle queries that $\mathcal{A}$ makes. After $\mathcal{A}$ finishes the game by outputting $(\mathsf{mpk}, \mathsf{vk}, \mathsf{R}, \mathsf{M}, \sigma = (\mathsf{ct}, \pi_{\mathtt{sign}}), \pi)$, we additionally require $\mathcal{A}$ wins if

1. The original condition holds.
2. The challenger runs $\mathsf{OnlineExtract}(\mathsf{M}, (\mathsf{R}, \mathsf{mpk}, \mathsf{ct}), \pi_{\mathtt{sign}}, L_{\mathcal{O}})$, and the output is not a collision of $\mathcal{O}$, which justifies denoting the output to be (I, sk, r).
3. $((\mathsf{R}, \mathsf{mpk}, \mathsf{ct}), (I, \mathsf{sk}, r)) \notin \tilde{R}_{\mathsf{sig}}$.

The advantage of $\mathcal{A}$ noticeably changes due to the modification if $\mathcal{A}$ can break online-extractability or find a collision of $\mathcal{O}$. Concretely, we construct an online-extractability adversary $\mathcal{B}_1$ with a collision-finding subroutine $\mathcal{B}_1'$ for $\Pi_{\mathsf{NIZK,lbl}}$ that simulates Game_2 for $\mathcal{A}$ such that

1. $\mathcal{B}_1$ answers random oracle queries by querying $(\mathtt{hash}, \cdot)$ (see Def. 3.4.5).
2. $\mathcal{B}_1$ answers signing queries by making an oracle call $(\mathtt{prove}, \mathsf{M}, (\mathsf{R}, \mathsf{mpk}, \mathsf{ct}), (I, \mathsf{sk}, r))$ instead of computing $\pi_{\mathtt{sign}}$ itself.
3. $\mathcal{B}_1$ answers the opening queries by making the oracle call $(\mathtt{extract}, \mathsf{M}, (\mathsf{R}, \mathsf{mpk}, \mathsf{ct}),$ $\pi_{\mathtt{sign}})$ instead of running $\mathsf{OnlineExtract}$.
4. The subroutine $\mathcal{B}_1'$ only collects the output of the extraction oracle forwarded from $\mathcal{B}_1$.

5. $\mathcal{B}_1$ outputs 1 to represent the end of the online-extractability experiment after $\mathcal{A}$ terminates and outputs a signature.
6. If there exists a collision of the random oracle output by the extraction oracle (i.e. an element in $\tilde{R}_{\mathsf{sig}} - R_{\mathsf{sig}}$), $\mathcal{B}_1'$ outputs the collision. Otherwise, $\mathcal{B}_1'$ outputs $\perp$.

Say $\mathcal{B}_1'$ outputs $\perp$, $\mathcal{A}$ wins in Game_1 but loses in Game_2. The win of Game_1 implies that $(\mathsf{ct}, \pi_{\mathtt{sign}})$ was not the output of a query $(\mathtt{sign}, \mathsf{mpk}, \mathsf{vk}', \mathsf{R}, \mathsf{M})$ for any vk' and $\mathsf{NIZK.Verify}^{\mathcal{O}}_{\mathsf{lbl}}(\mathsf{M}, (\mathsf{R}, \mathsf{mpk}, \mathsf{ct}), \pi_{\mathtt{sign}}) = \top$. Also, the loss in Game_2 implies that $((\mathsf{R}, \mathsf{mpk}, \mathsf{ct}), (I, \mathsf{sk}, r)) \notin \tilde{R}_{\mathsf{sig}}$. These meet the requirements for $\mathcal{B}_1$ to win the online extractability game. Therefore, we have $\mathsf{Adv}_1(\mathcal{A}) \leq \mathsf{Adv}_2(\mathcal{A}) + \mathsf{Adv}^{\mathsf{mpOE}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{Coll}}_{\mathcal{O}}(\mathcal{B}_1')$.

Game_3 : This is the same as Game_2 except that we change the way the challenger answers signing queries from $\mathcal{A}$. Specifically, the challenger generates ct as in Game_2 but uses the zero-knowledge simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ for $\Pi_{\mathsf{NIZK,lbl}}$ to create the proof $\pi_{\mathtt{sign}}$. That is, it forwards the random-oracle queries to Sim_0, and runs Sim_1 to get $\pi_{\mathtt{sign}}$. It then outputs $(\mathsf{ct}, \pi_{\mathtt{sign}})$ as the signature.

The modification results in a negligible loss for $\mathcal{A}$ in its advantage unless the adversary $\mathcal{A}$ can distinguish the simulation of $\Pi_{\mathsf{NIZK,lbl}}$ with a non-negligible chance. Concretely, we can turn $\mathcal{A}$ into an adversary $\mathcal{B}_2$ against the zero-knowledge experiment of $\Pi_{\mathsf{NIZK,lbl}}$ (Def. 3.4.3) where $\mathcal{B}_2$ simulates Game_2 or Game_3 for $\mathcal{A}$ and outputs whatever $\mathcal{A}$ returns. Precisely, $\mathcal{B}_2$ forwards either the real prover response or the simulated response (from $\mathsf{NIZK.Sim} = (\mathsf{NIZK.Sim}_0, \mathsf{NIZK.Sim}_1)$) from the ZK experiment to $\mathcal{A}$ for the random-oracle or signing queries. The adversary $\mathcal{B}_2$ simulates Game_2 or Game_3 for $\mathcal{A}$ for the former or the latter case respectively. Therefore, we have $\mathsf{Adv}_2(\mathcal{A}) \leq \mathsf{Adv}_3(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_2)$.

Game_4 : This is the same as Game_3 except that we change the winning condition again: the challenger guesses a random index $\widetilde{I} \in \{1, \ldots, \mathcal{Q}_u\}$ at the outset of the game. If $\mathcal{A}$ makes a corruption query to corrupt the verification key returned in the $\widetilde{I}$-th user key generation query, then Game_4 aborts. The game results in a win if the winning condition of Game_3 is met and if $\widetilde{I} = I$. Since $\widetilde{I}$ is information-theoretically hidden during the execution of the game, we have $\widetilde{I} = I$ with probability $1/\mathcal{Q}_u$. Therefore, we have $\mathsf{Adv}_3(\mathcal{A}) = \mathcal{Q}_u \mathsf{Adv}_4(\mathcal{A})$.

Let $\mathcal{B}_3$ be a GAIP adversary, which simulates Game_4 for $\mathcal{A}$. Given a GAIP challenge (E_0, E'), the adversary $\mathcal{B}_3$ simulates an execution of Game_4 such that

- instead of running $\mathsf{ARS.UKGen}$ to answer the $\widetilde{I}$-th $\mathtt{ukeygen}$ query, $\mathcal{B}_3$ assigns $\mathsf{vk}_{\widetilde{I}} = (E_0, E')$ and then sends it to the adversary;
- instead of running $\mathsf{ARS.Sign}$ to answer the signing query of the verification key $\mathsf{vk}_{\widetilde{I}}$, $\mathcal{B}_3$ uses the simulator of $\Pi_{\mathsf{NIZK,lbl}}$ to generate the signature;
- $\mathcal{B}_3$ aborts the simulation if $\mathcal{A}$ makes a corruption query of $\mathsf{vk}_{\widetilde{I}}$.

The view of $\mathcal{A}$ during $\mathcal{B}_3$'s simulation is the same as the execution of Game_4, so $\mathsf{OnlineExtract}$ outputs a valid witness $(\widetilde{I}, \mathsf{sk}, r)$ with probability at least $\mathsf{Adv}_4(\mathcal{A})$. Therefore, we have $\mathsf{Adv}_4(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{GAIP}}(\mathcal{B}_3)$. By collecting the bounds, the result follows. $\square$

Theorem 5.4.5. *The accountable ring signature scheme Π_{ARS} in Fig. 5.8 is traceable and tracing sound in the random oracle model. More precisely, for any adversary $\mathcal{A}$ that runs in time T, we have adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ that run in time $\mathsf{poly}(\lambda)T$, such that*

$$\mathsf{Adv}^{\mathsf{Tra}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{soundness}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_1)$$

and

$$\mathsf{Adv}^{\mathsf{TraS}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{soundness}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_2) + 2\mathsf{Adv}^{\mathsf{soundness}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{B}_3)$$

Proof. We prove the two properties separately as follows:

Traceability. Traceability follows from the statistical soundness of $\Pi_{\mathsf{NIZK,lbl}}$, the uniqueness encryption Π_{PKE} (Rem. 5.3.2), and the correctness of Π_{NIZK}. Suppose for the purpose of a contradiction that $\mathcal{A}$ outputs $(\mathsf{rr}, \mathsf{R}, \mathsf{M}, (\mathsf{ct}, \pi_{\mathtt{sign}}))$ such that the following two conditions are satisfied.

1. $\mathsf{Verify}(\mathsf{mpk}, \mathsf{R}, \mathsf{M}, (\mathsf{ct}, \pi_{\mathtt{sign}})) = \top$, where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{MKGen}(\mathsf{pp}; \mathsf{rr})$.

2. $\mathsf{Judge}(\mathsf{mpk}, \mathsf{R}, \mathsf{vk}, \mathsf{M}, (\mathsf{ct}, \pi_{\mathtt{sign}}), \pi_{\mathsf{open}}) = \perp$, where $(\mathsf{vk}, \pi_{\mathsf{open}}) \leftarrow \mathsf{Open}(\mathsf{msk}, \mathsf{R}, \mathsf{M}, (\mathsf{ct}, \pi_{\mathtt{sign}}))$.

It follows from the first condition that $\mathsf{NIZK.Verify}_{\mathsf{lbl}}(\mathsf{M}, \mathsf{X} = (\mathsf{R}, \mathsf{mpk}, \mathsf{ct}), \pi_{\mathtt{sign}}) = \top$. Let $\mathsf{st} = (\mathsf{R}, \mathsf{mpk}, \mathsf{ct})$. Suppose for the purpose of a contradiction that there exists a witness $\mathsf{wt} = (I, \mathsf{sk}, r)$ such that $(\mathsf{st}, \mathsf{wt}) \in \tilde{R}_{\mathsf{sig}}$. Note that the witness here will not be a hash collision; otherwise, Open will not output a verification key. It follows from the second condition that $\mathsf{Dec}(\mathsf{msk}, \mathsf{ct} = \mathsf{Enc}(\mathsf{mpk}, I; r)) = I$, which implies that $((\mathsf{mpk}, \mathsf{ct}, I), \mathsf{msk}) \in R_{\mathsf{open}}$. The correctness of Π_{NIZK} implies that $\mathsf{NIZK.Verify}((\mathsf{mpk}, \mathsf{ct}, I), \pi_{\mathsf{open}}) = \top$. This leads to a contradiction that Judge outputs $\perp$.

Therefore, $\mathcal{A}$ produces valid proofs for statements not in $\tilde{R}_{\mathsf{sig}}$ with probability at least $\mathsf{Adv}^{\mathsf{Tra}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A})$. We can therefore turn $\mathcal{A}$ into a statistical soundness adversary $\mathcal{B}_1$ against $\Pi_{\mathsf{NIZK,lbl}}$ as follows.

1. Generate $\mathsf{pp} \leftarrow \mathsf{ARS.Setup}(1^\lambda)$ for a security parameter λ.

2. Invoke $\mathcal{A}$ with pp and obtain $(\mathsf{rr}, \mathsf{R}, \mathsf{M}, \sigma)$ from $\mathcal{A}$ where $\sigma = (\mathsf{ct}, \pi_{\mathtt{sign}})$.

3. Run $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{ARS.MKGen}(\mathsf{pp}; \mathsf{rr})$.

4. Output $(\mathsf{M}, \mathsf{pk} := (R, \mathsf{mpk}, \mathsf{ct}), \pi_{\mathtt{sign}})$

If $\mathcal{A}$ wins, $\mathcal{B}_1$ succeeds. Therefore, $\mathsf{Adv}^{\mathsf{Tra}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{soundness}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_1)$.

Tracing soundness. Similarly, tracing soundness follows from the statistical soundness of Π_{NIZK} and $\Pi_{\mathsf{NIZK,lbl}}$, and the unique encryption of the Π_{PKE} (Rem. 5.3.2). Recall that if $\mathcal{A}$ wins the tracing soundness game, $\mathcal{A}$ outputs three valid proofs $\pi_{\mathtt{sign}}$, π_0, π_1 together with $\mathsf{mpk}, \mathsf{R}, \mathsf{M}, \sigma = (\mathsf{ct}, \pi_{\mathtt{sign}})$. Similarly, suppose for the purpose of a contradiction that three witnesses of the proofs for the corresponding relations exist. That is, there exist witnesses (I, sk, r), msk_0 and msk_1 such that

$$((\mathsf{R}, \mathsf{mpk}, \mathsf{ct}), (I, \mathsf{sk}, r)) \in \tilde{R}_{\mathsf{sig}}$$
$$((\mathsf{mpk}, \mathsf{ct}, I_0), \mathsf{msk}_0) \in \tilde{R}_{\mathsf{open}}$$
$$((\mathsf{mpk}, \mathsf{ct}, I_1), \mathsf{msk}_1) \in \tilde{R}_{\mathsf{open}},$$

with $I_0 \neq I_1$.

However, it follows from the unique encryption of Π_{PKE} and the first two equations that $I = I_0$. Similarly, we have $I = I_1$ from the first and the last equations. This contradicts that $I_0 \neq I_1$. Therefore, at least one of $\pi_{\mathtt{sign}}, \pi_0, \pi_1$ is a valid proof of an invalid statement, i.e. a X for which does not exist W such that $(\mathsf{X}, \mathsf{W}) \in \tilde{R}_{\mathsf{sig}}$ (or $(v) \in \tilde{R}_{\mathsf{open}}$), with probability at least $\mathsf{Adv}^{\mathsf{TraS}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A})$. We can thereby turn $\mathcal{A}$ into statistical-soundness adversaries $\mathcal{B}_2$ and $\mathcal{B}_3$ for $\Pi_{\mathsf{NIZK,lbl}}$ and Π_{NIZK}, respectively, that simulate the tracing soundness game and output $\pi_{\mathtt{sign}}$, π_0 or π_1, respectively. The procedure is the same as $\mathcal{B}_1$ described above. Then we have $\mathsf{Adv}^{\mathsf{TraS}}_{\Pi_{\mathsf{ARS}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{soundness}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_2) + 2\mathsf{Adv}^{\mathsf{soundness}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{B}_3)$. □

5.5 Tightly Secure Variant

Observe the only source of lossiness in the previous section, among Thms. 5.4.2, 5.4.4 and 5.4.5, was in the unforgeability proof where the loss depends on the number of members in the ring/group. In this section, we apply the Katz-Wang technique [KW03] to modify our construction in Fig. 5.8 to obtain a variant with a tight reduction.

We start by giving an intuition of the method. Recall that in the proof of Thm. 5.4.4, the reduction is given a challenge instance $\mathsf{st} = E_1 \in \mathcal{E}$, guesses which member's signature the adversary will forge, and assigns E_1 to be the verification key vk of the selected user. If the adversary queries the corruption oracle on the key vk, the reduction fails and aborts due to the unknown corresponding secret key for vk. If the guess is correct and the adversary successfully forges the signature, then the reduction can recover a witness wt such that $(\mathsf{st}, \mathsf{wt}) \in \tilde{R}_{\mathsf{sig}}$, which is either a collision of the hash function or containing $s \in G$ such that $(E_1, s) \in R_{\mathsf{GAIP}}$. Therefore, if the adversary with an advantage ϵ makes $\mathcal{Q}_u$ user key generation queries, then the reduction can extract a witness with probability roughly $\epsilon/\mathcal{Q}_u$.

A high-level viewpoint of the Katz-Wang method is that each user is given a pair of statements $(E^{(1)}, E^{(2)})$ as the verification key vk, with only one witness s as the secret signing key, such that either $(E^{(1)}, s)$ or $(E^{(2)}, s)$ is in the relation R_{GAIP}. Now, given a GAIP challenge E_1 and $\mathcal{Q}_u$ key generation queries, the reduction uses the self-reducibility to generate

$\mathsf{vk}_i = (r_i^{(1)} \star E_{b_i}, r_i^{(2)} \star E_{1-b_i})$ where $r_i^{(1)}, r_i^{(2)} \leftarrow G, b_i \leftarrow \{0,1\}$, and then records $(i, \mathsf{vk}_i, b_i, r_i^{(1)}, r_i^{(2)})$ for $i \in [\mathcal{Q}_u]$. Upon receiving the corruption query of the i-th verification key, it returns $r_i^{(b_i+1)}$ satisfying $(r_i^{(b_i+1)} \star E_0, r_i^{(b_i+1)}) \in R_{\mathsf{GAIP}}$. It follows that the guessing step above is not required anymore. As long as the adversary wins the unforgeability game by forging a signature, the reduction can extract a witness for one of the $\{r_i^{(2-b_i)} \star E_1\}_{i\in[\mathcal{Q}_u]}$ with probability 1/2. Roughly speaking, if the success rate of the adversary is ϵ, then by adding the witness by $-r_i^{(2-b_i)}$ for some $i \in [\mathcal{Q}_u]$ the reduction can return the answer for the challenge E_1 with probability around $\epsilon/2$.

Formally, we build a tightly secure accountable ring signature scheme $\Pi_{\mathsf{ARS}}^{\mathsf{Tight}} = (\mathsf{ARS.Setup}, \mathsf{ARS.MKGen}, \mathsf{ARS.UKGen}, \mathsf{ARS.Sign}, \mathsf{ARS.Verify}, \mathsf{ARS.Open}, \mathsf{ARS.Judge})$ based on the following tools. In the NIZK for the signing relation $R_{\mathsf{sig}}^{\mathsf{Tight}}$ lies the only difference between the tools used in Sec. 5.4.

- An effective group action $(G, E_0, \star)$.
- A group-action-based IND-CPA public key encryption $\Pi_{\mathsf{PKE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ described in Sec. 5.3.1, which can be derived from $(G, E_0, \star)$.
- Sound NIZK $\Pi_{\mathsf{NIZK}} = (\mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ without labels for R_{open} and $\tilde{R}_{\mathsf{open}}$ in (based on $\Pi_{\Sigma}^{\mathsf{OPbase}}$ in Fig. 5.1) where

$$R_{\mathsf{open}} = \left\{ ((\mathsf{pk} = (E_0, E), \mathsf{ct}, I), \mathsf{sk}) \;\middle|\; \begin{array}{c} E = \mathsf{sk} \star E_0 \;\wedge \\ \mathsf{ct} = \mathsf{Dec}(\mathsf{sk}, I) \end{array} \right\},$$

$$\tilde{R}_{\mathsf{open}} = \left\{ \Big(((\mathsf{pk} = (E_0, E), \mathsf{ct}, I), \mathsf{W}\Big) \;\middle|\; \begin{array}{c} \mathsf{W} = \mathsf{sk} \in G \;\wedge \\ E = \mathsf{sk} \star E_0 \;\wedge \mathsf{ct} = \mathsf{Dec}(\mathsf{sk}, I) \text{ or} \\ \mathsf{W} = (x_1, x_2) \in \{0,1\}^* \wedge \\ x_1 \neq x_2 \wedge \mathcal{O}(x_1) = \mathcal{O}(x_2) \end{array} \right\}.$$

- Online-extractable NIZK $\Pi_{\mathsf{NIZK,lbl}} = (\mathsf{NIZK.Prove_{lbl}}, \mathsf{NIZK.Verify_{lbl}})$ with labels for $R_{\mathsf{sig}}^{\mathsf{Tight}}$ and $\tilde{R}_{\mathsf{sig}}^{\mathsf{Tight}}$ in Fig. 5.5 (based on $\Pi_{\Sigma}^{ARSbase}$ in Fig. 5.2).

$$R_{\mathsf{sig}}^{\mathsf{Tight}} = \left\{ \Big(((X_i^{(j)})_{(i,j)\in[N]\times[2]}, \mathsf{pk}, \mathsf{ct}), (I, b, s, r) \Big) \;\middle|\; \begin{array}{c} (I, b, s, r) \in [N] \times [2] \times G \times G \\ X_I^{(b)} = s \star E_0 \;\wedge \\ \mathsf{ct} = (Y_0, Y) = \mathsf{Enc}(\mathsf{pk}, I; r) \end{array} \right\}$$

$$\tilde{R}_{\mathsf{sig}}^{\mathsf{Tight}} = \left\{ \Big(((X_i^{(j)})_{(i,j)\in[N]\times[2]}, \mathsf{pk}, \mathsf{ct}), \mathsf{W} \Big) \;\middle|\; \begin{array}{c} \mathsf{W} = (I, b, s, r) \in [N] \times G \times G \\ X_I^{(b)} = s \star X_0 \wedge \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, I; r) \text{ or} \\ \mathsf{W} = (x_1, x_2) \in \{0,1\}^* \wedge \\ x_1 \neq x_2 \wedge \mathcal{O}(x_1) = \mathcal{O}(x_2) \end{array} \right\},$$

where the differences compared to $R_{\mathsf{sig}}, \tilde{R}_{\mathsf{sig}}$ are highlighted in red.

Bootstrapping NIZK for $\tilde{R}_{\mathsf{sig}}^{\mathsf{Tight}}$. At a glance, to build an NIZK for $\tilde{R}_{\mathsf{sig}}^{\mathsf{Tight}}$ based on the NIZK for $\tilde{R}_{\mathsf{sig}}$ in the previous section using [CDS94]. It is clear that applying an OR-proof $\tilde{R}_{\mathsf{sig}}$ to the NIZK for $\tilde{R}_{\mathsf{sig}}$ immediately gives an NIZK for $\tilde{R}_{\mathsf{sig}}$ as desired. However, the method will result in a larger signature size by a factor of two.

Here, we present a novel technique by bootstrapping the NIZK for $\tilde{R}_{\mathsf{sig}}$ based on Fig. 5.2. The high-level idea is fairly simple. By encoding both $(I, 1)$ and $(I, 2)$ into the same message (e.g. I) in G to be encrypted by Enc, the decryption of the ciphertext is I while the auxiliary input $b \in \{1, 2\}$ remains hidden. Also, the special soundness of $\Pi_{\Sigma}^{\mathsf{ARSbase}}$ (Thm. 5.3.5) can also be extended to recover b by testing from the verification key $(X_I^{(1)}, X_I^{(2)})$. The zero-knowledge property of the scheme (the same as Thm. 5.3.6) and the fact that the auxiliary input $b \in \{1, 2\}$ remains hidden are essential to show the indistinguishability of the $X_i^{(1)}$ and $X_i^{(2)}$ in the reduction and result in a reduction loss of only 2. We give a concrete base construction in Fig. 5.9 for $R_{\mathsf{sig}}^{\mathsf{Tight}}$ and $\tilde{R}_{\mathsf{sig}}^{\mathsf{Tight}}$. The proofs for the essential properties as a sigma protocol are identical to Thms. 5.3.5 and 5.3.6 and the differences are highlighted in red. We hence skip the proofs.

The building blocks listed above are combined similarly to Fig. 5.8. We detail the resulting protocol in Fig. 5.10. Regarding the security notions, we only focus on unforgeability. The others are a direct consequence of the proofs given for the non-tight construction in Fig. 5.8.

Theorem 5.5.1. *The accountable ring signature scheme $\Pi_{\mathsf{ARS}}^{\mathsf{Tight}}$ in Fig. 5.10 is unforgeable in the random oracle model. More precisely, for any adversary $\mathcal{A}$ that runs in time T and makes Q_u queries to the* `ukeygen` *oracle, there exist adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$, running in time $\mathsf{poly}(\lambda)T$, such that*

$$\mathsf{Adv}_{\Pi_{\mathsf{ARS}}}^{\mathsf{Unf}}(\mathcal{A}) \leq \mathsf{Adv}_{\Pi_{\mathsf{NIZK,lbl}}}^{\mathsf{mpOE}}(\mathcal{B}_1) + \mathsf{Adv}_{\Pi_{\mathsf{NIZK,lbl}}}^{\mathsf{ZK}}(\mathcal{B}_2) + 2\mathsf{Adv}^{\mathsf{GAIP}}(\mathcal{B}_3).$$

Proof. We prove unforgeability using a hybrid argument with the following series of games. Let the advantage of an adversary $\mathcal{A}$ in Game_i be denoted by $\mathsf{Adv}_i(\mathcal{A})$.

- The first game, Game_1, is the original unforgeability game defined in Def. 5.2.4. The adversary's advantage in this game is $\mathsf{Adv}_1(\mathcal{A}) = \mathsf{Adv}_{\mathsf{ARS}}^{\mathsf{Unf}}(\mathcal{A})$ by definition.

- Game_2 is the same as Game_1, but with a modified winning condition. We let the challenger maintain a list $L_{\mathcal{O}}$ of all the random-oracle queries that $\mathcal{A}$ makes. When $\mathcal{A}$ finishes the game by outputting $(\mathsf{mpk}, \mathsf{vk}, \mathsf{R}, \mathsf{M}, \sigma = (\mathsf{ct}, \pi_{\mathtt{sign}}), \pi)$, the challenger runs $(I, b, \mathsf{sk}, r) \leftarrow \mathsf{OnlineExtract}(\mathsf{M}, (\mathsf{pp}_1, \mathsf{R}, \mathsf{mpk}, \mathsf{ct}), \pi_{\mathtt{sign}}, L_{\mathcal{O}})$. The game results in a loss if $((\mathsf{pp}_1, \mathsf{R}, \mathsf{mpk}, \mathsf{ct}), (I, , b, \mathsf{sk}, r)) \notin \tilde{R}_{\mathsf{sig}}^{\mathsf{Tight}}$, otherwise, the winning condition is not changed. As we have shown in the proof of Thm. 5.4.4, there exists an online-extractability adversary $\mathcal{B}_1$ for $\Pi_{\mathsf{NIZK,lbl}}$ running in time $\mathsf{poly}(\lambda)T$ with a collision-finding subroutine $\mathcal{B}_1'$ such that $\mathsf{Adv}_1(\mathcal{A}) \leq \mathsf{Adv}_2(\mathcal{A}) + \mathsf{Adv}_{\Pi_{\mathsf{NIZK,lbl}}}^{\mathsf{mpOE}}(\mathcal{B}_1) + \mathsf{Adv}_{\mathcal{O}}^{\mathsf{Coll}}(\mathcal{B}_1')$.

- The third game, Game_3, is the same as Game_2 except that we change the way the challenger answers signing queries from $\mathcal{A}$. Specifically, the challenger generates ct as in Game_2 but

```
round 1: $P_1'^{\mathcal{O}}(((X_i^{(j)})_{(i,j)\in[N]\times[2]}, \mathsf{pk} = (E_0, E), \mathsf{ct} = (Y_0, Y)), (I, b, s, r))$
1: $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$
2: $(s', r', \mathsf{bits}_1, \cdots, \mathsf{bits}_N, \mathsf{bits}_{N+1}, \cdots, \mathsf{bits}_{2N}) \leftarrow \mathcal{O}(\mathsf{Expand} \| \mathsf{seed})$   ▷ Sample $(s', r') \in G \times G$ and $\mathsf{bits}_i \in \{0,1\}^\lambda$
3: for $i$ from 1 to $N$ do
4:     $\mathsf{ct}_i \leftarrow (r' \star Y_0, (r' - i) \star Y)$
5:     for $j \in \{1,2\}$ do
6:         $T_{2(i-1)+j} \leftarrow s' \star X_i^{(j)}$
7:         $\mathsf{C}_{2(i-1)+j} \leftarrow \mathcal{O}(\mathsf{Com} \| T_{2(i-1)+j} \| \mathsf{ct}_i \| \mathsf{bits}_{2(i-1)+j})$   ▷ Create commitments $\mathsf{C}_i \in \{0,1\}^{2\lambda}$
8: $(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MerkleTree}(\mathsf{C}_1, \cdots, \mathsf{C}_N, \mathsf{C}_{N+1}, \cdots, \mathsf{C}_{2N})$
9: Prover sends $\mathsf{com} \leftarrow \mathsf{root}$ to Verifier.
```

```
round 2: $V_1'(\mathsf{com})$
1: $c \xleftarrow{\$} \{0,1\}$
2: Verifier sends $\mathsf{ch} \leftarrow c$ to Prover.

round 3: $P_2'((I, s, r), \mathsf{ch})$
1: $c \leftarrow \mathsf{ch}$
2: if $c = 1$ then
3:     $(s'', r'') \leftarrow (s' + s, r' + r)$
4:     $\mathsf{path} \leftarrow \mathsf{getMerklePath}(\mathsf{tree}, 2(I-1)+b)$
5:     $\mathsf{resp} \leftarrow (s'', r'', \mathsf{path}, \mathsf{bits}_I)$
6: else
7:     $\mathsf{resp} \leftarrow \mathsf{seed}$
8: Prover sends $\mathsf{resp}$ to Verifier
```

```
Verification: $V_2'^{\mathcal{O}}(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$
1: $(\mathsf{root}, c) \leftarrow (\mathsf{com}, \mathsf{ch})$
2: if $c = 1$ then
3:     $(s'', r'', \mathsf{path}, \mathsf{bits}) \leftarrow \mathsf{resp}$
4:     $(\widetilde{T}, \widetilde{\mathsf{ct}}) \leftarrow (s'' \star E_0, (r'' \star E_0, r'' \star E))$
5:     $\widetilde{\mathsf{C}} \leftarrow \mathcal{O}(\mathsf{Com} \| \widetilde{T} \| \widetilde{\mathsf{ct}} \| \mathsf{bits})$
6:     $\widetilde{\mathsf{root}} \leftarrow \mathsf{ReconstructRoot}(\widetilde{\mathsf{C}}, \mathsf{path})$
7:     Verifier accepts only if $\widetilde{\mathsf{root}} = \mathsf{root}$.
8: else
9:     Repeat round 1 with $\mathsf{seed} \leftarrow \mathsf{resp}$.
10:    Output accept if the computation results in root, and reject otherwise.
```

Figure 5.9: The base tightly-secure traceable OR sigma protocol $(P' = (P_1', P_2'), V' = (V_1', V_2'))$ for the relations R_{sig} and $\tilde{R}_{\mathsf{sig}}^{\mathsf{Tight}}$. The changes compared to Fig. 5.2 are highlighted in red. Informally, $O(\mathsf{Expand}\|\cdot)$ and $O(\mathsf{Com}\|\cdot)$ are a PRG and a commitment scheme instantiated by the random oracle, respectively.

uses the $\Pi_{\mathsf{NIZK,lbl}}$ zero-knowledge simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ to create the proof $\pi_{\mathtt{sign}}$. As we have shown in the proof of Thm. 5.4.4, there exists a zero-knowledge adversary $\mathcal{B}_2$ for $\Pi_{\mathsf{NIZK,lbl}}$ running in time $\mathsf{poly}(\lambda)T$ and such that $\mathsf{Adv}_2(\mathcal{A}) \le \mathsf{Adv}_3(\mathcal{A}) + \mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK,lbl}}}(\mathcal{B}_2)$.

- Finally, we consider an adversary $\mathcal{B}_3$ against GAIP which simulates Game_3 for $\mathcal{A}$. At the beginning of the game, the adversary $\mathcal{B}_3$ is given the challenge E_1 over $\mathsf{pp} = (G, \mathcal{E}, E_0, \star)$. The adversary $\mathcal{B}_3$ invokes $\mathcal{A}$ with pp, and simulates an execution of Game_3 with one difference. When answering the i-th $\mathtt{ukeygen}$ query, $\mathcal{B}_3$ computes $\mathsf{vk}_i = (r_i^{(1)} \star E_{b_i}, r_i^{(2)} \star E_{1-b_i})$ where $r_i^{(1)}, r_i^{(2)} \leftarrow G, b_i \leftarrow \{0,1\}$, and then records $(i, \mathsf{vk}_i, b_i, r_i^{(1)}, r_i^{(2)})$ for $i \in [\mathcal{Q}_u]$. Note that now $\mathcal{B}_3$ is able to respond to any valid corruption query $\mathtt{corrupt}$. Upon receiving the corruption query of the i-th verification key, $\mathcal{B}_3$ returns $r_i^{(b_i+1)}$ satisfying

$(r_i^{(b_i+1)} \star E_0, r_i^{(b_i+1)}) \in R_{\mathsf{GAIP}}$. The view of $\mathcal{A}$ during $\mathcal{B}_3$'s simulation is the same as its view during a real execution of Game_3, so $\mathsf{OnlineExtract}$ outputs a valid witness $(\widetilde{I}, \mathsf{sk} = (b', \mathsf{sk}'), r)$ with probability at least $\mathsf{Adv}_3(\mathcal{A})$. Since the simulated process of the key generation follows the same distribution determined by $\mathsf{ARS.UKGen}$ in the real execution, there is an $1/2$ chance that $b' = (2 - b_{\widetilde{I}})$. That is, $(E_1, \mathsf{sk}' - r_i^{(2-b_i)}) \in R_{\mathsf{GAIP}}$. Therefore, we have $\mathsf{Adv}_3(\mathcal{A})/2 \leq \mathsf{Adv}^{\mathsf{GAIP}}(\mathcal{B}_3)$.

$\square$

```
ARS.UKGen(1^λ)
1: $\mathsf{sk}^{(1)}, \mathsf{sk}^{(2)} \leftarrow G$
2: $b \xleftarrow{\$} \{1,2\}$
3: $\mathsf{vk} \leftarrow (\mathsf{sk}^{(1)} \star E_0, \mathsf{sk}^{(2)} \star E_0)$
4: return $\mathsf{vk}, \mathsf{sk} := (b, \mathsf{sk}^{(b)})$
```

```
ARS.MKGen(1^λ)
1: $\mathsf{msk} \leftarrow G$
2: $\mathsf{mpk} \leftarrow \mathsf{msk} \star E_0$
3: return $(\mathsf{mpk}, \mathsf{msk})$
```

```
ARS.Sign(mpk, sk, R, M)
1: $\{(\mathsf{vk}_i^{(1)}, \mathsf{vk}_i^{(2)})\}_{i\in[N]} \leftarrow \mathsf{R}$
2: if $\nexists (I, b) : (\mathsf{pk}_I^{(b)}, \mathsf{sk}) \in R_{\mathsf{GAIP}}$ then
3:     return $\perp$.
4: $r \xleftarrow{\$} G$
5: $\mathsf{ct} = \mathsf{Enc}(\mathsf{mpk}, I; r)$
6: $\pi_{\mathtt{sign}} \leftarrow \mathsf{NIZK.Prove}_{\mathsf{lbl}}(\mathsf{M}, (\mathsf{R}, \mathsf{mpk}, \mathsf{ct}), (I, b, \mathsf{sk}, r))$
7: return $\sigma := (\mathsf{ct}, \pi_{\mathtt{sign}})$
```

```
ARS.Verify(mpk, R, M, σ)
1: $(\mathsf{ct}, \pi_{\mathtt{sign}}) \leftarrow \sigma$
2: return $\mathsf{NIZK.Verify}_{\mathsf{lbl}}(\mathsf{M}, (\mathsf{R}, \mathsf{mpk}, \mathsf{ct}), \pi_{\mathtt{sign}})$
```

```
ARS.Judge(mpk, R, vk, M, σ, π_open)
1: $(\mathsf{ct}, \pi_{\mathtt{sign}}) \leftarrow \sigma$
2: if $\nexists I : \mathsf{vk} = \mathsf{R}_I$ then
3:     return $\perp$.
4: $b_0 \leftarrow \mathsf{ARS.Verify}(\mathsf{mpk}, \mathsf{R}, \mathsf{M}, \sigma)$
5: $b_1 \leftarrow \mathsf{NIZK.Verify}((\mathsf{mpk}, \mathsf{ct}, I), \pi_{\mathtt{open}})$
6: return $b_0 \wedge b_1$
```

```
ARS.Open(msk, R, M, σ)
1: if $\mathsf{ARS.Verify}(\mathsf{mpk}, \mathsf{R}, \mathsf{M}, \sigma) = \perp$ then
2:     return $\perp$
3: $(\mathsf{ct}, \pi_{\mathtt{sign}}) \leftarrow \sigma$
4: $I \leftarrow \mathsf{Dec}(\mathsf{msk}, \mathsf{ct})$
5: $\pi_{\mathtt{open}} \leftarrow \mathsf{NIZK.Prove}((\mathsf{mpk}, \mathsf{ct}, I), \mathsf{msk})$
6: return $\pi := (\mathsf{R}_I, \pi_{\mathtt{open}})$
```

Figure 5.10: Modified tightly-secure construction of an accountable ring signature $\Pi_{\mathsf{ARS}}^{\mathsf{Tight}}$ using Katz-Wang method and self-reducibility of the group action inverse problem, a public-key encryption algorithm $\Pi_{\mathsf{PKE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ described in Sec. 5.3.1 where Setup is assume to generate $(G, E_0, \star)$ and KeyGen is implicitly used in $\mathsf{ARS.MKGen}$. $\Pi_{\mathsf{NIZK,lbl}}$ is an NIZK proof system with labels for $R_{\mathsf{sig}}^{\mathsf{Tight}}$, and Π_{NIZK} is a NIZK proof system without labels for R_{open}.

5.6 Further Optimization and Performance

We can ameliorate the signature size by utilizing the two techniques presented in [BKP20]. We briefly summarise the techniques.

	N					Hardness	Security	Anonymity	Manager
	2	2^5	2^6	2^{10}	2^{21}	Assumption	Level		Account.
Ours	3.6	6.0	6.6	9.0	15.5	GAIP	$\star$	CCA	Yes
[ESZ22]	/	12	/	19	/	MSIS/MLWE	NIST 2	CPA	No
[KKW18]	/	/	280	418	/	LowMC	NIST 5	selfless-CCA	No

Table 5.1: Comparison of the signature size (KB) of some concretely efficient post-quantum group signature schemes. The number N in the table represent the number of users (i.e. the group size). The first row is our scheme. Manager accountability states whether the (possibly malicious) group manager is accountable when opening a signature to some user. Namely, it is "Yes" when even a malicious group manager cannot falsely accuse an honest user of signing a signature that it hasn't signed.

* 128 bits of classical security and 60 bits of quantum security [Pei20].

Unbalanced Challenge Space. One can observe the response of a prover in the proof system Fig. 7.2 for the challenge 0 is much shorter than the one for challenge one. The former is a single seed, while the latter is a bunch of group elements. To do this we introduce the unbalanced challenge space $C_{M,K} = \{\mathsf{ch} \in \{0,1\}^M \mid |\mathsf{ch}| = K\}$, where $|\cdot|$ is the ℓ_1-norm and $2^\lambda \leq \frac{M!}{K!(M-K)!}$. Concretely, we chose the challenge space as string of length $M = 855$ with Hamming weight $K = 19$ and we thereby obtain a much smaller proof size while the online-extractability and zero-knowledge remain the same.

Seed Trees. The seed tree technique allows the prover to produce a large amount of the seeds using PRNG and iteratively generating binary subtrees (see Sec 2.7 of [BKP20]). The leaves of the tree are the seeds to be used. The prover can later reveal the generating nodes while not disclosing the information of those unrevealed leaves. The method reduces the size of responses for the challenge 0 in our case.

Our performance is given in Tab. 5.1 using CSIDH-512 [BKV19] together with a comparison with the state-of-art from different branches [KKW18, ESZ22]. Our accountable ring signature (also the group signature) gives the strongest security guarantees with a comparable signature size. Regarding the anonymity, the CPA anonymity given in [ESZ22] does not give an opening oracle to the adversary. While the selfless-CCA of [KKW18] gives an opening oracle to the adversary, the anonymity is not guaranteed if a member's signing key is exposed.

5.7 Discussion

We would like to suggest two potential directions for future work in this area. Firstly, it remains an open problem to construct a QROM secure variant of our construction or a different construction from isogenies. To our knowledge, there are no efficient (logarithmic) group signatures proven secure in the QROM. One possible approach to achieve this would be to utilize the online-extractability property established in [DFMS22] through the commit-and-open technique, also known as the Pass transform [Pas03], though this will come with additional overhead. Further investigation is required to determine the feasibility of this approach.

Secondly, we note that there is room for improvement in terms of the signature size for our ring/group signature scheme, as well as the ring signature scheme proposed in [BKP20]. One possible approach would be to develop a base sigma protocol for the R_{sig} relation that uses a ternary or larger challenge space. However, this approach may not be straightforward due to several issues. Firstly, it is unclear how to expand the challenge space for a simpler ring signature relation such as the one considered in [BKP20]. Secondly, to increase the challenge space, we typically require a structural challenge space [7]. However, the current construction relies on a shorter string (seed) to sample a group element using a PRNG, where the response in each iteration is either a seed or group elements along with a Merkle path. The string can be shorter than a group element representation due to the subexponential cryptoanalysis [Kup05]. A larger challenge space may require an additional group element as a response because a random string might not support a structural challenge space, which could lead to an overhead. Therefore, a structural challenge space may benefit the signature size by reducing the number of repetitions, but could result in an overhead from the response. We leave these two questions as open problems for future research.

[7]For example, we introduce a *cyclic structure* of the signing key in the generalization technique in Sec. 6.4.1 for our blind signature to extend the challenge space in each repetition.

Chapter 6

Blind Signatures

This chapter presents the work carried out in [KLLQ23], which the author of the thesis co-authored. The author contributed to the ideas of the generalization of the base scheme of the project (see **Generalization** of Sec. 6.4.1), proposes the *ring group action inverse problem* along with its structural analysis, and has a partial contribution of compiling the interface for the OMUF proof (Sec. 6.3). This chapter has been adapted by removing the partial blind signature construction without compromising the essence of the original work.

Abstract. In this chapter, we present a new technique for constructing blind signatures from isogenies that cleverly utilizes the quadratic twist base on the group action inverse problem. With our approach, we have accomplished the first provably secure blind signature – CSI-Otters[1] – from isogenies based on the group action inverse problem. Then, we generalize the result into a framework – Otters – for the known-order abelian group actions without twists and develop a potential tradeoff between the efficiency and the signature size.

6.1 Introduction

Blind signatures, which were first proposed by Chaum [Cha82], enable a user to receive a signature on a message from a signer, without revealing the content of the message to the signer. The signer remains unaware of the message they have signed, hence the term "blind". Meanwhile, the signature can also be publicly verifiable by using the signer's public key. In practice, it is sometimes necessary to consider the extension of *partially* blind signatures, introduced by Abe and Fujisaki [AF96], that further allow embedding a message agreed by both the signer and the user into the signature. Unlike a blind signature, where the entire message is blinded, in a partial blind signature scheme, only a part of the message is blinded, while the other parts are left unblinded, where the public part can include, for instance, the expiration date of the signature. Blind signatures[2] were initially utilized to construct various cryptographic

[1] CSI-fish with Or-proof Twisted ThreE-Round Signature.

[2] For readability, we focus on blind signatures below when the distinction between the partial and non-partial difference is insignificant.

protocols such as e-cash [Cha82, CFN90, OO92], anonymous credentials [Bra94, CL01], and e-voting [Cha88, FOO93]. However, these concepts have recently experienced a surge in interest, primarily due to their potential and applicability in blockchain technology [YL19, BDE$^+$23], and privacy-preserving authentication tokens [VPN22, HIP$^+$22].

Blind signature schemes based on lattices are currently considered the most promising and efficient class of schemes that can resist attacks by quantum computers. Recent research has made significant progress in lattice-based blind signatures, as demonstrated in [HKLN20, LNP22, AKSY22, dK22], where the signature size ranges from 50 KB to 10 MB. However, these signature sizes are still an order of magnitude larger than their classical counterparts, which typically range from a few hundred bytes to 1 KB. As the demand for post-quantum security and user privacy grows, our objective is to investigate the possibility of developing a post-quantum blind signature scheme that offers a more compact signature size.

Isogeny-based constructions offer a potential avenue for developing a post-quantum blind signature scheme with a compact signature size. Although their signing and verification times are generally less efficient, standard isogeny-based signature schemes [DG19, BKV19, DKL$^+$20] are known to produce comparable or smaller signatures than those produced by lattice-based schemes. Furthermore, for more advanced forms of signature schemes, such as ring signatures and group signatures, isogenies can produce much shorter signatures than their lattice-based counterparts [BKP20, BDK$^+$22] and provide stronger cryptographic guarantees (e.g. full anonymity and tight reductions) in contrast to others in the literature.

However, this approach seems challenging to pursue. Generally, there are two methods for constructing a blind signature. The first one is based on the Schnorr blind signature [CP93]. This approach builds on a sigma (or an identification) protocol with a "nice" algebraic property and boosts it into a blind signature by appropriately randomizing the interaction. This nice algebraic property has recently been stated informally as *modules* [HKL19, HKLN20]. However, this property is not known to be available when using group actions such as CSIDH. The second approach is based on the generic round-optimal construction (possibly based in a common reference string model) proposed by Fischlin [Fis06]. It requires proving, at the minimum, possession of a valid signature of a standard signature scheme using a non-interactive zero-knowledge proof (NIZK) for a sophisticated relation involving the commitment and encryption schemes. While del Pino and Katsumata [dK22] and Agrawal et al. [AKSY22] (and a very recent work [BLNS23]) recently used this approach to construct more efficient lattice-based blind signatures than previously known, this seems impractical to translate to the isogeny setting due to the lack of efficient NIZKs for such complex relations.

In summary, while isogenies have the potential to produce the shortest post-quantum blind signatures, it is unclear how we can leverage known approaches to build them. This brings us to the main question of this work:

Can we construct an efficient post-quantum (partially) blind signature from isogenies or even abelian group actions in general?

6.2 Preliminaries

Notations.

We summarize some notations unique to this chapter. First, we use multiplication notation for the known-order effective group action $(G, \mathcal{E}, E_0, \star)$. We assume the group G is cyclic of order N with a public generator $\mathfrak{g}$. Since several subroutines and samplings are used in this chapter, for a set S, we let $s \xleftarrow{\$} S$ represent uniformly sampling from S to distinguish the use of $\leftarrow$ of a subroutine.

We use $\mathbb{Z}_d$ to denote the set $\{0, \ldots, d-1\}$. Moreover, any vector is indexed from 0, e.g., $\mathbf{a} \in \mathbb{Z}_d^\kappa$ is expressed as $(a_0, \ldots, a_{\kappa-1})$. With an overload of notations, for any integer j, we define the bold font $\mathbf{j}$ as the length-κ vector $(j, \ldots, j)$. For any positive integer d and $a \in \mathbb{Z}$ or $\mathbb{Z}_d$, we use $[a]_d$ to denote $(a \mod d) \in \mathbb{Z}_d$. For an element g and vector $\mathbf{a} = (a_1, \ldots, a_n)$, we use $g^{\mathbf{a}}$ as a shorthand for $(g^{a_1}, \ldots, g^{a_n})$. Moreover, for any operation $\star$ defined between two elements g and h and vectors $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n)$, we use $g^{\mathbf{a}} \star h^{\mathbf{b}}$ as a shorthand for $(g^{a_1} \star h^{b_1}, \ldots, g^{a_n} \star h^{b_1})$.

6.2.1 Blind Signatures

We define blind signatures consisting of three moves, e.g., [AO00, KLX22b, KLX22a] by ignoring the tag info for a partial blind signature.

Definition 6.2.1 (Blind Signature Scheme)**.** *A three-move blind signature* $\mathsf{BS} = (\mathsf{BS.KGen}, \mathsf{BS.S}, \mathsf{BS.U}, \mathsf{BS.V})$ *with an efficiently decidable public key space* $\mathcal{PK}$ *consists of the following PPT algorithms:*

$\mathsf{BS.KGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: *On input the security parameter* 1^λ*, the key generation algorithm outputs a public and a secret key* $(\mathsf{pk}, \mathsf{sk})$*.*

$\mathsf{BS.S} = (\mathsf{BS.S}_1, \mathsf{BS.S}_2)$: *The interactive signer algorithm consists of two phases:*

- $\mathsf{BS.S}_1(\mathsf{sk}) \to (\mathsf{st}_\mathsf{S}, \rho_{\mathsf{S},1})$: *On input a secret key* sk*, it outputs an internal signer state* st_S *and a first-sender message* $\rho_{\mathsf{S},1}$*.*[3]
- $\mathsf{BS.S}_2(\mathsf{st}_\mathsf{S}, \rho_\mathsf{U})) \to \rho_{\mathsf{S},2}$: *On input a signer state* st_S *and a user message* ρ_U*, it outputs a second-sender message* $\rho_{\mathsf{S},2}$*.*

$\mathsf{BS.U} = (\mathsf{BS.U}_1, \mathsf{BS.U}_2)$: *The interactive user algorithm consists of two phases:*

- $\mathsf{BS.U}_1(\mathsf{pk}, \mathsf{M}, \rho_{\mathsf{S},1}) \to (\mathsf{st}_\mathsf{U}, \rho_\mathsf{U})$: *On input a public key* $\mathsf{pk} \in \mathcal{PK}$*, a message* M*, and a first-sender message* $\rho_{\mathsf{S},1}$*, it outputs an internal user state* st_U *and a user message* ρ_U*.*
- $\mathsf{BS.U}_2(\mathsf{st}_\mathsf{U}, \rho_{\mathsf{S},2})) \to \sigma$: *On input a user state* st_U *and a second-signer message* $\rho_{\mathsf{S},2}$*, it outputs a signature* σ*.*

[3]We assume without loss of generality that sk includes pk and st_S includes $(\mathsf{pk}, \mathsf{sk})$ and omit it when the context is clear. Therefore, we may assume that st_U includes M.

$\mathsf{BS.V}(\mathsf{pk}, \mathsf{M}, \sigma) \to 1$ **or** 0 : *On input a public key* pk*, a message* M*, and a signature* σ*, the verification algorithm outputs* 1 *to indicate the signature is valid, and* 0 *otherwise.*

We require a blind signature to be complete, blind against malicious signer, and one-more unforgeable. We first define correctness.

Definition 6.2.2 (Perfect Correctness)**.** *A three-move blind signature scheme* BS *is* perfectly correct *if for all public and secret key pairs* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{BS.KGen}(1^\lambda)$ *and every message* M*, we have*

$$\Pr\left[\mathsf{BS.V}(\mathsf{pk}, \mathsf{M}, \sigma) = 1 \;\middle|\; \begin{array}{r} (\mathsf{st}_\mathsf{S}, \rho_{\mathsf{S},1}) \leftarrow \mathsf{BS.S}_1(\mathsf{sk}) \\ (\mathsf{st}_\mathsf{U}, \rho_\mathsf{U}) \leftarrow \mathsf{BS.U}_1(\mathsf{pk}, \mathsf{M}, \rho_{\mathsf{S},1}) \\ \rho_{\mathsf{S},2} \leftarrow \mathsf{BS.S}_2(\mathsf{st}_\mathsf{S}, \rho_\mathsf{U}) \\ \sigma \leftarrow \mathsf{BS.U}_2(\mathsf{st}_\mathsf{U}, \rho_{\mathsf{S},2}) \end{array}\right] = 1$$

The following definitions are taken from [KLX22b, KLX22a]. The blindness roughly requires the transcript to be independent of the signature even if the signer choses the keys maliciously.

Definition 6.2.3 (Blindness Under Chosen Keys)**.** *We define blindness of a three-move blind signature scheme* BS *via the following game between a challenger and an adversary* $\mathcal{A}$*:*

Setup. *The challenger samples* $\mathsf{coin} \in \{0, 1\}$ *and runs* $\mathcal{A}$ *on input* 1^λ*.*

Online Phase. *When* $\mathcal{A}$ *outputs messages* $\widetilde{\mathsf{M}}_0$ *and* $\widetilde{\mathsf{M}}_1$*, and a public key* $\mathsf{pk} \in \mathcal{PK}$*, it assigns* $(\mathsf{M}_0, \mathsf{M}_1) := (\widetilde{\mathsf{M}}_\mathsf{coin}, \widetilde{\mathsf{M}}_{1-\mathsf{coin}})$*.* $\mathcal{A}$ *is then given access to oracles* U_1*,* U_2*, which behave as follows.*

Oracle U_1**.** *On input* $b \in \{0, 1\}$*, and a first-signer message* $\rho_{\mathsf{S},1,b}$*, if the session* b *is not yet open, the oracle marks session* b *as* `opened` *and runs* $(\mathsf{st}_{\mathsf{U},b}, \rho_{\mathsf{U},b}) \leftarrow \mathsf{BS.U}_1(\mathsf{pk}, \mathsf{M}_b, \rho_{\mathsf{S},1,b})$*. It returns* $\rho_{\mathsf{U},b}$ *to* $\mathcal{A}$*.*

Oracle U_2**.** *On input* $b \in \{0, 1\}$ *and a second-signer message* $\rho_{\mathsf{S},2,b}$*, if the session* b *is* `opened`*, the oracle creates a signature* $\sigma_b \leftarrow \mathsf{BS.U}_2(\mathsf{st}_{\mathsf{U},b}, \rho_{\mathsf{S},2,b})$*. It marks session* b *as* `closed`*. Oracle* U_2 *does not output anything.*

Output Determination. *When both sessions are closed and* $\mathsf{BS}.V(\mathsf{pk}, \mathsf{M}_b, \sigma_b) = 1$ *for* $b \in \{0, 1\}$*, the oracle returns the two signatures* $(\sigma_\mathsf{coin}, \sigma_{1-\mathsf{coin}})$ *to* $\mathcal{A}$*, where note that* σ_coin *(resp.* $\sigma_{1-\mathsf{coin}}$*) is a valid signature for* $\widetilde{\mathsf{M}}_0$ *(resp.* $\widetilde{\mathsf{M}}_1$*) regardless of the choice of* coin*.* $\mathcal{A}$ *outputs a guess* coin^* *for* coin*. We say* $\mathcal{A}$ wins *if* $\mathsf{coin}^* = \mathsf{coin}$*.*

We say BS *is* blind under chosen keys *if the advantage of* $\mathcal{A}$ *defined as* $\Pr[\mathcal{A}$ *wins*$]$ *is negligible.*

One-more unforgeability roughly ensures that at most one valid signature is generated after each execution of $\mathsf{BS.Sign}$. Formally, we have the following.

Definition 6.2.4 (One-More-Unforgeability)**.** *We define* ℓ*-one-more unforgeability (*ℓ-OMUF*) for any* $\ell \in \mathbb{N}$ *of a three-move blind signature scheme* BS *via the following game between a challenger and an adversary* $\mathcal{A}$*:*

Setup. *The challenger samples* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{BS.KGen}(1^\lambda)$ *and runs* $\mathcal{A}$ *on input* pk*. It further initializes* $\ell_{\mathsf{closed}} = 0$ *and* $\mathsf{opened}_{\mathsf{sid}} = \mathtt{false}$ *for all* $\mathsf{sid} \in \mathbb{N}$*.*

Online Phase. $\mathcal{A}$ *is given access to oracles* S_1 *and* S_2*, which behave as follows.*

Oracle S_1**:** *The oracle samples a fresh session identifier* sid*. It sets* $\mathsf{opened}_{\mathsf{sid}} \leftarrow$ *true and generates* $(\mathsf{st}_{\mathsf{S},\mathsf{sid}}, \rho_{\mathsf{S},1}) \leftarrow \mathsf{BS.S}_1(\mathsf{sk})$*. Then it returns* sid *and the first-sender message* $\rho_{\mathsf{S},1}$ *to* $\mathcal{A}$*.*

Oracle S_2**:** *On input a user message* ρ_{U} *and a session identifier* sid*, if* $\ell_{\mathsf{closed}} \geq \ell$ *or* $\mathsf{opened}_{\mathsf{sid}} = \mathtt{false}$*, then it returns* $\perp$*. Otherwise, it sets* $\ell_{\mathsf{closed}}++$ *and* $\mathsf{opened}_{\mathsf{sid}} = \mathtt{false}$*. It then computes the second-signer message* $\rho_{\mathsf{S},2} \leftarrow \mathsf{BS.S}_2(\mathsf{st}_{\mathsf{S},\mathsf{sid}}, \rho_{\mathsf{U}})$ *and returns* $\rho_{\mathsf{S},2}$ *to* $\mathcal{A}$*.*

Output Determination. *When* $\mathcal{A}$ *outputs distinct tuples* $(\mathsf{M}_1, \sigma_1), \ldots, (\mathsf{M}_k, \sigma_k)$*, we say* $\mathcal{A}$ wins *if* $k \geq \ell_{\mathsf{closed}} + 1$ *and for all* $i \in [k]$*,* $\mathsf{BS}.V(\mathsf{pk}, \mathsf{M}_i, \sigma_i) = 1$*.*

We say BS *is* ℓ-one-more unforgeable *if the advantage of* $\mathcal{A}$ *defined as* $\Pr[\mathcal{A}$ *wins*$]$ *is negligible.*

6.2.2 Assumption: Ring Group Action Inverse Problem

Recall that $G \cong \mathbb{Z}_N$ and $\mathbb{Z}_N$ is a ring. We introduce a generalized version of the group action inverse problem by considering a *d-th primitive root of unity*, denoted by ζ_d, over $\mathbb{Z}_N$ such that $\zeta_d^d = 1$ and $\zeta_d^j \neq 1$ for any $j \in [d-1]$.

Such a ζ_d exists if d is a divisor of the Carmichael function $\lambda(N)$. Concretely, if $N = \Pi p_i^{e_i}$ where p_i are distinct primes, we have $\lambda(N) = \mathsf{lcm}_i(\lambda(p_i^{e_i}))$ where

$$\lambda(p_i^{e_i}) = \begin{cases} \frac{1}{2}\varphi(p_i^{e_i}) & \text{if } p_i = 2 \wedge e_i \geq 3 \\ \varphi(p_i^{e_i}) & \text{otherwise} \end{cases}$$

where φ is the Euler phi-function.

We define the *ring group action inverse problem* (analogue to the ring variant of a lattice assumption with a cyclotomic structure) [4] with respect to ζ_d as follows.

Definition 6.2.5 (ζ_d-Ring Group Action Inverse Problem (rGAIP))**.** *Given* $(E_0, S) \in \mathcal{E}^{d+1}$ *where* $S = ([\mathfrak{g}^{s\zeta_d^j}] \star E_0)_{j \in \mathbb{Z}_d}$*,* $s \leftarrow [N]$ *and* $d|\lambda(N)$ *(here* λ *is the Carmichael function), the* ζ_d-ring group action inversion problem *(*ζ_d-rGAIP*) is to recover* s*.*

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\zeta_d\text{-}\mathsf{rGAIP}}_{(G,\mathcal{E},E_0,\star)}(\mathcal{A}) = \Pr[\mathcal{A}\text{ wins}]$ where the probability is also taken over the randomness used in the experiment.

When the context is clear, we may remove d from the subscript or remove ζ_d entirely and call it rGAIP for simplicity. This problem is a generalized version of GAIP, which is a ζ_2-rGAIP with $\zeta_2 = -1$ (when the prime $p = 3 \pmod 4$). To see this, by taking the quadratic twist of

[4]Roughly, in the ring LWE setting, the polynomial ring is taken modulo a cyclotomic polynomial $x^d - 1$ for some parameter d.

a GAIP instance $E' = [\mathfrak{g}^s] \star E_0$, we have $(E', E'^{-1}) = ([\mathfrak{g}^s] \star E_0, [\mathfrak{g}^{-s}] \star E_0)$. Similar to GAIP [FIM+14, BN18, CDEL21] having polynomial-time hidden shift problem (HSP) algorithms for insecure group structures, the hardness of an ζ_d-rGAIP also relies on the underlying algebraic structure and the specific choice of ζ_d. In Sec. 6.5.2, we provide a structural analysis on the ζ_d-rGAIP for CSIDH-512 and display a few weak and hard instances depending on ζ_d. We show that for some carefully-chosen d (depending on N), ζ_d-rGAIP is as essentially as hard as the original GAIP.

Finally, when constructing our optimized blind signatures in Sec. 6.4, we require d to satisfy a further requirement other than ζ_d-rGAIP being hard. Informally, we require $\eta_d = \mathsf{lcm}_{i\in[d-1]}(\mathsf{gcd}(\zeta_d^i - 1, N))$ to be small for the extractor of the underlying sigma protocol to be efficient. More details can be found in Sec. 6.4.

6.3 Generic OMUF Proofs for Blind Schnorr-Type Signatures

This section is taken almost verbatim from [KLLQ23]. This section is essential to have a simpler and rigorous *one more unforgeability* proof for our blind signature scheme.

In this section, we review the recent work of Kastner, Loss, and Xu [KLX22a] that provided a proof of the Abe-Okamoto (partially) blind signature [AO00]. The original security proof of the one-more unforgeability in [AO00] contained a leap of logic in the security proof (i.e., the scheme was correct but the security proof was not), and Kastner et al. provided a somewhat generic proof that works for many of the blind Schnorr-type signatures [CP93].[5]

While their focus was on the scheme by Abe and Okamoto, the proof is generic enough to capture other similar schemes (see for instance [KLX22a, Appendix F] that provides a proof sketch of [Abe01]). Indeed, the constructions we propose fall under their generic proofs as well. To this end, we extract the minimal definitions and prerequisite lemmas from [KLX22a] for the OMUF proof. Then, we turn the lemmas, derived from the setting of [AO00], into the *requirements* to argue the security for the blind Schnorr-type signatures. Here, we note that it is likely that one can rewrite [KLX22a] in a more generic fashion by borrowing the tools from [HKL19]. However, we chose not to for better readability and since isogenies do not naturally endow a *linear* identification scheme as required by [HKL19]. In conclusion, we want to stress that we are not claiming any technical originality in this section. Instead, we have compiled the ideas presented in [KLX22a] into an interface that is both accessible and practical. We hope the interface will prove useful for future work in this area.

Below, we provide a brief overview of the proof by Kastner et al. and then introduce the key lemmas that need to be proven in this thesis to apply their proof.

[5]Note that the proof in [KLX22a] relies on the fact that there are two possible signing keys per public key. Therefore, their proof does not work for the original Schnorr blind signature [CP93], which is known to be secure if we further rely on the algebraic group model [KLX22b].

6.3.1 Proof Overview

Loosely speaking, a blind Schnorr-type signature is a type of blind signature that builds on top of a Schnorr-type sigma protocol [Sch90]. The signer of the blind signature is identical to the prover in a sigma protocol, while the user of the blind signature modifies the verifier in the sigma protocol by appropriately adding blindness factors. In the proof of one-more unforgeability, the adversary (i.e., a malicious user) does not care if its forgeries are blind, and thus, how the blindness is achieved can be ignored for now.

At a high level, to argue one-more unforgeability, we would like the reduction to embed a hard problem into the public key of the blind signature and appeal to the special soundness of the underlying sigma protocol to extract a solution from the forgeries. However, unlike standard Fiat-Shamir based signatures, the reduction cannot rely on HVZK to simulate the signatures since the challenge is under the adversary's control. To simulate the interaction between the adversary, we thus allow the public key to have *two* valid secret keys, e.g., $(\mathsf{vk} = (E_0, [\mathfrak{g}^{a_0}] \star E_0, [\mathfrak{g}^{a_1}] \star E_0), \mathsf{sk} = (\delta, a_\delta))$ with $\delta \in \{0, 1\}$. The reduction embeds a hard problem into one of the secret keys while simulating with the other secret key.

What makes the security proof of blind Schnorr-type signatures tricky is that even if the adversary's view is independent of the secret key being used, this alone does not complete the proof. This is because to argue that the secret key extracted via the special soundness of the underlying sigma protocol is unbiased, we need to argue that the algorithm (i.e., reduction) executing the extractor of the special soundness is unbiased. While this holds for standard Fiat-Shamir based signature schemes since the reduction can invoke HVZK, this is not the case for blind signatures. As we discussed above, since the adversary chooses the challenge, the reduction can only try to invoke witness indistinguishability. However, witness indistinguishability breaks when the reduction *rewinds* the adversary since the reduction needs to simulate two transcripts using the same first commitment of the sigma protocol. Thus, the reduction is not compatible with the definition of witness indistinguishability.

That being said since the view of the adversary (in each run) is independent of the secret key being used, intuition tells us that the extraction works: the only thing that's not working is the security proof. To overcome this issue, Kastner et al. [KLX22a] provides a detailed analysis of the probability of the reduction succeeding while implicitly relying on witness indistinguishability. We note that Abe and Okamoto [AO00] also rely on the same proof approach but included a subtle but non-trivially fixable flaw to compute the probability.

6.3.2 Key Definitions, Lemmas, and Theorems

We extract the minimal definitions and lemmas from [KLX22a] in a self-contained manner so that the security of our blind signatures is established through several easy-to-state lemmas. For a more full exposition, we refer the readers to [KLX22a].

Preparation.

Assume the adversary against the one-more unforgeability game is restricted to make only $\ell+1$ distinct hash queries to the random oracle, where $\ell+1$ is the number of forgeries the adversary outputs[6]. Moreover, as with any blind Schnorr-type signature, we assume each signature in the forgery is associated with a distinct hash query. We also assume the public key of the blind signature has exactly two corresponding secret keys. More specifically, we assume the underlying sigma protocol is for the **NP** OR-relation R defined with respect to another **NP** relation R'. That is, $(\mathsf{X} := (\mathsf{X}'_0, \mathsf{X}'_1), \mathsf{W} := (\delta, \mathsf{W}'_\delta) \in R$, where $(\mathsf{X}'_0, \mathsf{W}'_0), (\mathsf{X}'_1, \mathsf{W}'_1)) \in R'$, X is the public key and W is the secret key. Finally, we assume the adversary's user-message ρ_U queried to the signing algorithm $\mathsf{BS.S}_2$ satisfies $\rho_\mathsf{U} \in \mathcal{C}$, where $\mathcal{C}$ is the challenge space of the underlying sigma protocol for relation R (and R').

We first define the notion of *instances*. Roughly, an instance defines the signer's key and randomness. We present a variant of the definition of instances in [KLX22a, Definition 4] that is agnostic to the underlying sigma protocol. We provide an explicit description of instances, analogous to [KLX22a, Definition 4], when we detail our construction of blind signatures.

Definition 6.3.1 (Instances)**.** *Assume the public key of a blind Schnorr-type signature has exactly two corresponding secret keys* $\mathsf{sk}_0 = (0, \mathsf{W}'_0)$ *and* $\mathsf{sk}_1 = (1, \mathsf{W}'_1)$*. We define two types of* instances $\mathbf{I}$*: A **0**-side (resp. **1**-side) instance consists of* sk_0 *(resp.* sk_1*) and the randomness used by the honest signer algorithm when the secret key is fixed to* sk_0 *(resp.* sk_1*), i.e., randomness excluding those used by the key generation algorithm.*

The main argument of Kastner et al. boils down to arguing that the output of the extraction algorithm (i.e., forking algorithm) explained above is independent of the instances.

Let $\overrightarrow{h}$ be the vector of responses returned by the random oracle, where $\left|\overrightarrow{h}\right| = \ell+1$, and let rand be the randomness used by the one-more unforgeability adversary. For a vector $\overrightarrow{h}$, its i-th entry is denoted by h_i, and the vector of its first i entries is denoted by $\overrightarrow{h}_{[i]}$. We define a deterministic wrapper algorithm $\mathcal{W}$ that simulates the interaction between the signer and the adversary given input $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$. $\mathcal{W}$ invokes the signer and the adversary on inputs $\mathbf{I}$ and rand, respectively, and uses $\overrightarrow{h}$ to answer the random oracle queries made by the adversary. We define $\mathcal{W}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ to output $\perp$ if the adversary aborts prematurely or fails to win the one-more unforgeability game, and otherwise, output what the adversary outputs. We then define the notion of *successful tuples* as follows.

Definition 6.3.2 (Successful Tuples)**.** *We define the set of* successful tuples *as follows:*

$$\mathsf{Succ} := \{(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \mid \mathcal{W}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \neq \perp\}.$$

[6] $\ell+1$ distinct hash queries are assumed in [KLX22a]. Their Lemma 1 shows how to turn a general adversary into one with $\ell+1$ queries. Similarly, by using a standard hybrid argument, it is clear that one can also turnurn an adversary with repeated queries into one with distinct queries.

We next define a sufficient condition to invoke the extraction algorithm of the underlying sigma protocol. This is a standard definition (often implicitly) used even for Fiat-Shamir based signatures.

Definition 6.3.3 (Successful Forking. [KLX22a, Definition 7]). *We say two successful input tuples* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in \mathsf{Succ}$ fork *from each other at index* $i \in [\ell+1]$ *if* $\overrightarrow{h}_{[i-1]} = \overrightarrow{h}'_{[i-1]}$ *but* $h_i \neq h'_i$. *We denote the set of hash vector pairs* (h_i, h'_i) *such that* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in \mathsf{Succ}$ fork *at index* i *as* $\mathrm{F}_i(\mathbf{I}, \mathsf{rand})$.

We next define the notion of transcripts. A *query transcript* denotes the user messages queried to the signer. A *full transcript* denotes the entire transcript produced by the signer and the adversary, including the final forgery.

Definition 6.3.4 (Query Transcript. [KLX22a, Definition 5]). *Consider the wrapper* $\mathcal{W}$ *running on input* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$. *The* query transcript, *denoted* $\overrightarrow{e}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$, *is the vector of user message* (ρ_U) *queries made to the signing algorithm* $\mathsf{BS.S}_2$ *(simulated by* $\mathcal{W}$*) by the adversary, ordered by* sid.

Definition 6.3.5 (Full Transcript. [KLX22a, Definition 6]). *Consider the wrapper* $\mathcal{W}$ *running on input* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$. *The* full transcript, *denoted* $\mathsf{trans}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$, *is the transcript produced between the signer and the adversary, i.e., all messages sent between the signer and user played by the adversary, including the forgeries.*

We now define *partners*, which plays a key role in the analysis of [AO00, KLX22a]. Informally, two tuples $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in \mathsf{Succ}$ are partners at i if they fork at this index i and produce the same query transcript. Note that this does not necessarily imply that each tuple results in the same full transcript.

Definition 6.3.6 (Partners. [KLX22a, Definition 8]). *We say two successful tuples* $(\mathbf{I}, rand, \overrightarrow{h})$, $(\mathbf{I}, rand, \overrightarrow{h}')$ *are partners at index* $i \in [\ell+1]$ *if the following hold:*

- $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ *and* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ *fork at index* i.
- $\overrightarrow{e}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = \overrightarrow{e}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$

We denote the set of $(\overrightarrow{h}, \overrightarrow{h}')$ *such that* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ *and* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ *are partners at index* i *by* $\mathrm{prt}_i(\mathbf{I}, \mathsf{rand})$.

A *triangle* is another key tool introduced in [AO00, KLX22a] in order to enhance the standard forking tuples with the nice properties of partners. A triangle consists of three vectors $\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}''$ such that each two vectors fork at the same index, and additionally, $(\overrightarrow{h}, \overrightarrow{h}')$ are partners.

Definition 6.3.7 (Triangles. [KLX22a, Definition 9]). *A triangle at index* $i \in [\ell+1]$ *with respect to* $\mathbf{I}$, rand *is a tuple of three successful tuples in the following set:*

$$\triangle_i(\mathbf{I}, \mathsf{rand}) = \left\{ \begin{array}{c|c} ((\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), & (\overrightarrow{h}, \overrightarrow{h}') \in \mathrm{prt}_i(\mathbf{I}, \mathsf{rand}) \\ (\mathbf{I}, \mathsf{rand}\, \overrightarrow{h}'), & (\overrightarrow{h}, \overrightarrow{h}'') \in \mathrm{F}_i(\mathbf{I}, \mathsf{rand})) \\ (\mathbf{I}, \mathsf{rand}\, \overrightarrow{h}'')) & (\overrightarrow{h}', \overrightarrow{h}'') \in \mathrm{F}_i(\mathbf{I}, \mathsf{rand}) \end{array} \right\}$$

For a triangle $((\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'')) \in \triangle_i(\mathbf{I}, \mathsf{rand})$*, we call the pair of tuples* $((\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'))$ *the* base*, and* $((\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}''))$ *and* $((\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}''))$ *the* sides*.*

We next define a map that transforms a b-side instance into a $(1-b)$-side instance for $b \in \{\mathbf{0}, \mathbf{1}\}$. Roughly, the map allows us to relate the number of triangles with a $\mathbf{0}$-side instance to those with a $\mathbf{1}$-side instance. We present a variant of the definition of instances in [KLX22a, Definition 12] that is agnostic to the underlying sigma protocol. We provide an explicit description of the map, analogous to [KLX22a, Definition 12], when we detail our construction of blind signatures.

Definition 6.3.8 (Mapping Instances via Transcript)**.** *For* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \mathsf{Succ}$*, we define* $\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I})$ *as a function that maps a **0**-side instance* $\mathbf{I}$ *(resp. **1**-side instance* $\mathbf{I}$*) to a **1**-side instance* $\mathbf{I}'$ *(resp. **0**-side instance* $\mathbf{I}'$*).*

In [KLX22a], the mapping is crucial to the counting argument as it maps an instance to the other branch for an adversary to create transcripts using the same hash values and randomness. One of the branches involves using a self-generated secret key while the other utilizes a hard instance as the public key. The purpose of the mapping is to determine the likelihood of the extractor being able to extract the witness for the hard instance (see the explanation of Requirement 1 below for a precise use). Note that the map does not need to be computationally feasible since it is only used in the counting argument rather than in the extraction procedure.

Finally, we formally define the witness extractor used by the reduction. We present a variant of the definition of witness extractor in [KLX22a, Definition 13] that is agnostic to the underlying sigma protocol. This is because the witness extractor's concrete description is defined using the special soundness extractor of the underlying sigma protocol, which we will do when we detail our construction of blind signatures.

Definition 6.3.9 (Witness Extraction)**.** *Fix* $\mathbf{I}, \mathsf{rand}$ *and let* $\overrightarrow{h}, \overrightarrow{h}' \in \mathrm{F}_i(\mathbf{I}, \mathsf{rand})$ *for some* $i \in [\ell + 1]$*. Moreover, denote* σ_i, σ_i' *the signatures that correspond to* h_i*,* h_i'*, respectively. We say deterministic algorithms* $(\mathsf{Ext}_0, \mathsf{Ext}_1)$ *are* witness extractors *if* $(\mathsf{Ext}_0(\sigma_i, \sigma_i'), \mathsf{Ext}_1(\sigma_i, \sigma_i')) \in \{(\mathsf{sk}_0, \perp), (\perp, \mathsf{sk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)\}$*.For* $b \in \{0, 1\}$*, we say that the* b-side witness can be extracted from $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ and $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ at index i *if* $\mathsf{Ext}_b(\sigma_i, \sigma_i')$ *outputs* sk_b*.*

Sufficient Conditions for One-More Unforgeability.

We are now prepared to formally present the main result of Kastner et al. [KLX22a]. First of all, if the map $\Phi_{\mathsf{rand}, \overrightarrow{h}}$ is a bijection that preserves transcripts for any rand and $\overrightarrow{h}$, then a partner tuple with a b-side instance maps to another partner tuple with a $(1-b)$-side instance for the same rand and $\overrightarrow{h}$ (see [KLX22a, Corollary 1 and Lemma 3]). This implies that the extracted witness from a partner tuple is independent of b and the secret key generated in the reduction. However, it is not clear if the reduction is able to obtain a partner tuple by rewinding. To this end, we use the sides of the triangle rather than the base (i.e., partner tuple) to extract a witness, where the main observation is that if a b-side witness can be extracted from the base of a triangle, then a b-side witness can be extracted from at least one of the sides. Then, we argue that the reduction having a b-side witness hits one corner of the base of a triangle in the first run, and then hits the top of the triangle such that it creates side with a $(1-b)$-side witness with a probability of roughly $1/2$.

The main contribution of Kastner et al. [KLX22a] was to make the above high-level argument precise. Their result is mostly purely statistical and it suffices to only prove that our blind signature satisfies the following two lemmas to invoke their main theorem concerning one-more unforgeability. The first lemma shows that the blind signature is perfectly *witness indistinguishable*. This is used to establish the extracted witness from a partner tuple is independent of the reduction's secret key.

Requirement 1 ([KLX22a, Lemma 2]). *Fix* $\mathsf{rand}, \overrightarrow{h}$. *For all tuples* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \mathsf{Succ}$, $\Phi_{\mathsf{rand}, \overrightarrow{h}}$ *is a self-inverse bijection and* $\mathsf{trans}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = \mathsf{trans}(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h})$.

The second lemma states that if a witness can be extracted from a base of a triangle, then the same witness can be extracted from at least one of its sides.

Requirement 2 ([KLX22a, Corollary 3]). *Fix* $\mathbf{I}, \mathsf{rand}$ *and let* $(\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}'') \in \triangle_i(\mathbf{I}, \mathsf{rand})$, *for some* $i \in [\ell{+}1]$. *If the **0**-side (**1**-side) witness can be extracted from the base* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ *of the triangle at index i, then one can also extract the **0**-side (**1**-side) witness from at least one of the sides* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'')$ *or* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'')$ *at index i.*

The following is the main theorem of Kastner et al. [KLX22a, Theorem 1] written slightly generally to be agnostic to the underlying hardness assumption.

Theorem 6.3.10. *Let the blind Schnorr-type signature* $(\mathsf{P})\mathsf{BS}$ *be as defined in the preparation of Sec. 6.3.2. In particular, assume the public key consists of two instances of the **NP** relation R' generated by a corresponding hard instance generator* IG *and the underlying sigma protocol has challenge space $\mathcal{C}$.*

If Requirements 1 and 2 hold, then for all $\ell \in \mathbb{N}$, if there exists an adversary $\mathcal{A}$ that makes Q hash queries to the random oracle and breaks the ℓ-one more unforgeability of $(\mathsf{P})\mathsf{BS}$ *with advantage* $\epsilon_{\mathcal{A}} \geq \frac{C_1}{|\mathcal{C}|} \cdot \binom{Q}{\ell+1}$, *then there exists an algorithm $\mathcal{B}$ that breaks the hard instance generator with advantage* $\epsilon_{\mathcal{B}} \geq C_2 \cdot \frac{\epsilon_{\mathcal{A}}^2}{\binom{Q}{\ell+1}^2 \cdot (\ell+1)^3}$ *for some universal positive constants C_1 and C_2.*

6.4 Our Construction Using Roots of Unity

In this section, we present a novel framework for a known-order EGA to construct blind signatures based on the *ring group action inverse problem* (rGAIP), which is a generalized version of the group action inverse problem (GAIP). When we instantiate it with CSIDH and the prime $p = 3 \pmod 4$, -1-rGAIP is exactly GAIP, which leads to an isogeny-based blind signature basing on the standard assumption.

In Secs. 6.4.5 to 6.4.7, we provide the proofs of the correctness, blindness, and OMUF of the construction under the assumption that rGAIP is hard. In Sec. 6.5, we provide analysis on the hardness of the rGAIP for the CISHD-512 parameter set and show that not all rGAIP instances are *equally difficult*.

6.4.1 Overview

Reviewing the Schnorr Blind Signature.

We first recall the Schnorr sigma/identification protocol between a prover with $(\mathsf{pk}, \mathsf{sk}) = (h = g^a, a) \in \mathbb{G} \times \mathbb{Z}_p$ and a verifier with pk. The prover samples $y \xleftarrow{\$} \mathbb{Z}_p$ and sends $Y = g^y$ to the verifier. The verifier sends a random challenge $c \xleftarrow{\$} \mathbb{Z}_p$ to the prover, where the prover replies with $r = y - a \cdot c$. The verifier is convinced that it was communicating with a prover in possession of $\mathsf{sk} = a$ if $g^r \cdot h^c = Y$. Here, if the verifier sets the challenge as $c = \mathcal{H}(Y \| \mathsf{M})$ for a message M and a hash function $\mathcal{H}$ modeled as a random oracle, then $\sigma = (c, r)$ serves as a signature based on the Fiat-Shamir transform [FS87], where the prover is the *signer* and the verifier is the *user with* M.

The interactive signing protocol described does not meet the requirement of *blindness*, which implies that a signature cannot be linked to a particular signing session. When the user produces the pair (M, σ), the signer can identify the session during which it signed σ. Specifically, the signature σ can be linked back to the user by inspecting the hash value c included in σ which indicates when it was used.

The Schnorr blind signature [CP93] allows the user to randomize the interaction, making the session transcript independent of the final signature. This is achieved by introducing randomness to the signature such that $\sigma' = (c + d, r + z)$, where (d, z) is uniformly sampled from $\mathbb{Z}_p$ with respect to the signer's view.

To accomplish this, when the user receives Y as the first-sender message, it generates $(d, z) \xleftarrow{\$} \mathbb{Z}_p^2$ and sets $Y' := g^z \cdot Y \cdot h^d$. It then computes $c' = \mathcal{H}(Y' \parallel \mathsf{M})$ and sends $c := c' - d$ to the signer, who replies with $r = y - a \cdot c$ as before. Since we have $g^r \cdot h^c = Y$, the user multiplies g^z and h^d on each side to obtain $g^{r+z} \cdot h^{c+d} = Y'$. Thus, $\sigma' = (c', r') := (c + d, r + z)$ is a valid signature for the message M. Moreover, it can be verified that this scheme satisfies perfect blindness since any signature $\sigma' = (c', r')$ can be generated from a transcript (Y, c, r) with equal probability, where the probability is taken over the randomness sampled by the user.

Difficulty with Group Actions.

In the above, the user is implicitly using a specific structure of the underlying Schnorr sigma protocol to randomize the interaction. Specifically, it is using the fact that $\mathbb{G}$ is a $\mathbb{Z}_p$-*module*. This allows the user to randomize the first-signer message $Y \in \mathbb{G}$ by multiplying it with the generator $g \in \mathbb{G}$ raised to the power of $z \in \mathbb{Z}_p$ and the public key $h = g^a \in \mathbb{G}$ lifted to the power of $d \in \mathbb{Z}_p$. This property has been more formally abstracted as a *linear identification protocol* [HKL19, HKLN20]. The linear identification protocol covers schemes based on classical groups and lattices, and it provides a systematic way to construct and analyze identification schemes by leveraging the underlying algebraic structure.

Unfortunately, this does not extend to the isogeny setting since isogenies are only a *group action.* Concretely, the CSIDH group action is defined as $\star : G \times \mathcal{E} \to \mathcal{E}$, where G is an ideal class group and $\mathcal{E}$ is a set of elliptic curves, and we further assume the structure of G is known and can be expressed as $G = \langle [\mathfrak{g}] \rangle \cong \mathbb{Z}_N$ for some $N \in \mathbb{N}$, where $\mathfrak{g}$ is the generator [BKV19].

First, we give a naive attempt to construct an isogeny-based Schnorr-style blind signature with a public key $\mathsf{pk} = A = [\mathfrak{g}^a] \star E_0 \in \mathcal{E}$, where a is a random element sampled from $\mathbb{Z}_N$, and E_0 is a fixed curve. However, unlike the traditional Schnorr blind signature scheme, we face limitations when randomizing the first-signer message $Y = [\mathfrak{g}^y] \star E_0$ for $y \in \mathbb{Z}_N$. While computing $[\mathfrak{g}^z] \star Y$ for a random $z \in G$ is possible, combining Y with $[\mathfrak{g}^d] \star A$ is not feasible, as they are both set elements. This is in contrast to the Schnorr blind signature, where one can randomize the first-signer message Y twice; once with a randomness d to conceal the challenge c and again with a randomness z to hide the second-signer message r. As a result, it is unclear how to use isogenies to construct a blind signature while only having one way to randomize Y. In summary, although we can randomize the first-signer message in one way, we need to explore alternative methods to construct a blind signature scheme.

Novel Use of the Quadratic Twist.

Our main observation to overcome this problem is to rely on the property that isogenies are slightly more expressive than a group action due to the *quadratic twist.* Given any $A = [\mathfrak{g}^a] \star E_0$ for an unknown $a \in \mathbb{Z}_N$, we can efficiently compute its quadratic twist $[\mathfrak{g}^{-a}] \star E_0$, which we denote by A^{-1}.

We first explain the underlying isogeny-based sigma protocol, where we assume for now that the challenge space is $\mathcal{C} = \{-1, 1\}$. Identically to the above, the prover sends $Y = [\mathfrak{g}^y] \star E_0$ for $y \xleftarrow{\$} \mathbb{Z}_N$. The verifier then sends a random challenge $c \xleftarrow{\$} \{-1, 1\}$, and the prover replies with $r = y - a \cdot c$. The verifier then verifies the "signature" $\sigma = (c, r)$ by checking whether $[\mathfrak{g}^r] \star A^c = Y$, where note that A^c is well-defined for $c \in \{-1, 1\}$ even though A comes from the set of elliptic curves. For an honest execution of the protocol, we have $[\mathfrak{g}^r] \star A^c = [\mathfrak{g}^r] \star ([\mathfrak{g}^{a \cdot c}] \star E_0) = [\mathfrak{g}^{r + a \cdot c}] \star E_0 = Y$ as desired.[7]

Our idea is to randomize this sigma protocol so that the signature $\sigma = (c, r)$ becomes $\sigma' = (c \cdot d, r \cdot d + z)$, where (d, z) is uniform over $\{-1, 1\} \times \mathbb{Z}_N$ from the view of the signer. Concretely, given the first-sender message Y, the user randomizes Y by sampling random $(d, z) \xleftarrow{\$} \{-1, 1\} \times \mathbb{Z}_N$ and sets $Y' := [\mathfrak{g}^z] \star Y^d$. It then computes $c' = \mathcal{H}(Y' \| \mathsf{M})$ and sends $c := c' \cdot d$. The signer replies with $r = y - a \cdot c$ as before. Since we have $[\mathfrak{g}^r] \star A^c = Y$, the user can first compute $[\mathfrak{g}^{r \cdot d}] \star A^{c \cdot d} = Y^d$. Namely, it performs nothing if $d = 1$, and computes the quadratic twist of both sides if $d = -1$. It then acts by $[\mathfrak{g}^z]$ to obtain $[\mathfrak{g}^{r \cdot d + z}] \star A^{c \cdot d} = [\mathfrak{g}^z] \star Y^d$. Since the right-hand side is Y', $\sigma' = (c', r') := (c \cdot d, r \cdot d + z)$ is a valid signature for the message

[7]Note that this is a standard (optimized variant of an) isogeny-based sigma protocol where 0 is removed from the challenge space (see for instance [BKV19]).

M as desired. Moreover, it can be checked that we have perfect blindness since c and r are both randomized; the (multiplicative) randomness $d \in \{-1,1\}$ hides the challenge c and the (additive) randomness $z \in \mathbb{Z}_N$ hides the response r. Put differently, any signature $\sigma' = (c', r')$ has an equal chance of being generated from a transcript (Y, c, r), where the probability is taken over the randomness sampled by the user.

Finally, to turn this basic idea into a secure blind signature, we enlarge the challenge space to be exponentially large, i.e., $\mathcal{C} = \{-1,1\}^\lambda$ where λ is the security parameter. All the above arguments naturally extend to this enlarged challenge space by running the protocol λ-times in parallel.

Generalization.

After discussing the preliminary results, it is reasonable to inquire about the following two issues:

1. Is it feasible to apply the approach described above to other group actions that lack twists?
2. Can we increase the challenge space to reduce the need for repetition?

Fortunately, the generalization technique presented in this section provides solutions to both of the questions.

It is a natural attempt to reduce the signature size by considering a larger public key space. Indeed, as shown in [BKV19], such an optimization is possible for standard signature schemes by relaxing GAIP to the multi-target GAIP. As a result, the soundness error of the underlying sigma protocol in a single round decreases to $\frac{1}{2S-1}$ from $\frac{1}{3}$ for a public key size S. Since the number of repetitions is decreased to $\frac{\lambda}{\log_2(2S-1)}$, this technique makes it possible to decrease the signature size, signing, and verification time at the cost of increased public key size. For isogeny-based protocols—which are generally slow but offer small key sizes—this is a very favorable tradeoff.

Unfortunately, a natural adaptation of the same relaxation will not apply to our case because the multi-target GAIP does not offer the particular structure that our blind signature requires. Roughly speaking, a main component of our blind signature requires a user/verifier to compute $[\mathfrak{g}^{z+y^*d}] \star E_0$ while only given $[\mathfrak{g}^{y^*}] \star E_0 \in \mathcal{E}$, $z \in \mathbb{Z}_N$ and d. This is only feasible by using the quadratic twist, which is when $d \in \{-1,1\}$. An unstructured random public key not only fails to benefit the user but also breaches the group structure of the challenge space since d is no longer restricted in $\{-1,1\}$.

To this end, we present a novel technique that allows us to trade off efficiency and the signature size using a structured public key. The high-level idea is fairly simple: to generalize the concept of the quadratic twist in the sense of the group action relation. In the previous section, both parties compute the action of $[\mathfrak{g}^r]$ on a curve E_0 or E_0^{-1} with respect to the challenge $c \in \mathbb{Z}_3^\times = \{-1,1\}$. Recall that $([\mathfrak{g}^r] \star E_0)^{-1} = [\mathfrak{g}^{-r}] \star E_0$. In other words, the challenge $c \in \mathbb{Z}_3^\times = \{-1,1\}$ is encoded into $\mathfrak{g}^c$. Since -1 is a second primitive root of unity over $\mathbb{Z}_N$, the challenge space, as a (multiplicative) group, induces an action on $\mathcal{E}$ by computing the twist.

We generalize the concept by expanding the challenge space to $\langle\zeta\rangle = \{1, \zeta, \zeta^2, \ldots, \zeta^{d-1}\}$, where $d \in \mathbb{N}$ and ζ is a d-th primitive root of unity over $\mathbb{Z}_N^\times$; that is, ζ satisfies $\zeta^d = 1$ and $\zeta^j \neq 1$ for any $j \in [d-1]$. Note that $\langle\zeta\rangle$ is naturally a multiplicative (sub)group, which provides an algebraic operation on the challenge space. The action $(r, c) \in \mathbb{Z}_N \times \mathbb{Z}_d$ on a curve $E_0 \in \mathcal{E}$ is defined to be $[\mathfrak{g}^{r\zeta^c}] \star E_0$. When $k = 2$ and ζ can be taken to be -1, this is identical to the scheme described above (the previous subsubsection). However, unlike the case $d = 2$ where we have the formula derived from the quadratic twist, when $d \geq 3$ the signer is required to compute $[\mathfrak{g}^{y^*_{b,j}\zeta}] \star E_0$ for each $(b, j) \in [2] \times [\kappa]$ in $\mathsf{BS.S}_1$ to aid the user's computation.

6.4.2 Preparation

Our construction requires one more property from the d-th primitive root of unity ζ to be useful.

Looking ahead, when we construct a sigma protocol for the rGAIP relation, the special soundness extractor must solve the secret exponent $a \in \mathbb{Z}_N$, given distinct $c_1, c_2 \in \mathbb{Z}_N^2$ and $r_1 = y + a\zeta^{c_1}$, $r_2 = y + a\zeta^{c_2} \pmod N$ for an unknown a and y. If $\mathbb{Z}_N$ was a finite field, then this is trivial. However, in general when $\mathbb{Z}_N$ is a ring, such a may not be efficiently computable because the equation may not be uniquely solvable. One sufficient condition would be to only use a $d \in \mathbb{Z}_N$ such that $(\zeta^{c_1} - \zeta^{c_2})$ is invertible over $\mathbb{Z}_N$ for all distinct $(c_1, c_2) \in \mathbb{Z}_N^2$. However, this is an overly restrictive requirement and we thus make the following relaxed requirement.

Requirement 3. *We require* $\eta_d = \mathsf{lcm}_{i\in[d-1]}(\mathsf{gcd}(\zeta^i - 1, N)) = \mathsf{poly}$.

The requirement is equivalent to finding a d such that d divides many Euler-phi values of maximal prime power divisors of the group order N (see Sec. 6.5.1 about the existence and finding a root). Informally, when η_d is polynomial in the security parameter n, then we can brute force all $a \in \mathbb{Z}_N$ such that $a \cdot (\zeta^{c_1} - \zeta^{c_2}) = z$ for a given $(c_1, c_2, z) \in \mathbb{Z}_N^3$. Formally, we have the following.

Lemma 6.4.1. *Let* (N, d, ζ) *be a public parameter where the factorization of* N *is known and let* $\eta_d = \mathsf{lcm}_{i\in[d-1]}(\mathsf{gcd}(\zeta^i - 1, N))$. *Then, there exists an extractor* Ext' *that takes as input the public parameter and* $(r_1, r_2, c_1, c_2) \in \mathbb{Z}_N^2 \times \mathbb{Z}_d^2$ *where* c_1, c_2 *are distinct with relations* $r_1 = y + a\zeta_d^{c_1}$, $r_2 = y + a\zeta_d^{c_2} \pmod N$, *and outputs a list containing* $a \in \mathbb{Z}_N$ *of size not greater than* η_d *in time* $\mathsf{poly}(\eta_d)$.

Proof. By calculating $(r_1 - r_2)\zeta_d^{-c_2} = a(\zeta_d^{c_1 - c_2} - 1)$, the extractor solves a by solving the linear equation reduced modulo the prime power factors of N, then using the Chinese remainder theorem to obtain a list of candidates of a. The size of the list is the number of solutions for the linear equation, which is at most η_d. □

6.4.3 Base Sigma Protocol with a Large Challenge Space

We first introduce the base sigma protocol with a larger challenge space assuming Requirement 3. This is depicted in Fig. 6.1 without the boxes. We will show the correctness, HVZK, and, importantly, special soundness of this sigma protocol.

P:		V:
$\mathsf{X} = ((A_0^j)_{j\in\mathbb{Z}_d}, (A_1^j)_{j\in\mathbb{Z}_d})$ $= (([\mathfrak{g}^{a_0\zeta^j}] * E_0)_{j\in\mathbb{Z}_d}, ([\mathfrak{g}^{a_1\zeta^j}] * E_0)_{j\in\mathbb{Z}_d})$ $\mathsf{W} = (\delta, a_\delta) \in \{0,1\} \times \mathbb{Z}_N$		$\mathsf{X} = ((A_0^j)_{j\in\mathbb{Z}_d}, (A_1^j)_{j\in\mathbb{Z}_d})$
$\mathbf{y}_\delta \xleftarrow{\$} \mathbb{Z}_N^\kappa$ $\mathbf{Y}_\delta = [\mathfrak{g}^{\mathbf{y}_\delta}] * E_0$ $\boxed{(\mathbf{Y}_\delta^j = [\mathfrak{g}^{\mathbf{y}_\delta\zeta^j}] * E_0)_{j\in\mathbb{Z}_d}}$ $(\mathbf{c}_{1-\delta}, \mathbf{r}_{1-\delta}) \xleftarrow{\$} \mathbb{Z}_d^\kappa \times \mathbb{Z}_N^\kappa$ $\mathbf{Y}_{1-\delta} = [\mathfrak{g}^{\mathbf{r}_{1-\delta}}] * A_{1-\delta}^{\mathbf{c}_{1-\delta}}$ $\boxed{(\mathbf{Y}_{1-\delta}^j = [\mathfrak{g}^{\mathbf{r}_{1-\delta}\zeta^j}] * A_{1-\delta}^{[\mathbf{c}_{1-\delta}+\mathbf{j}]_d})_{j\in\mathbb{Z}_d}}$	$(\mathbf{Y}_0, \mathbf{Y}_1)$ $\boxed{(\mathbf{Y}_0^j, \mathbf{Y}_1^j)_{j\in\mathbb{Z}_d}}$ $\xrightarrow{\qquad\qquad}$	
	$\mathbf{c}$ $\xleftarrow{\qquad\qquad}$	$\mathbf{c} \xleftarrow{\$} \mathbb{Z}_d^\kappa$
$\mathbf{c}_\delta = \mathbf{c} - \mathbf{c}_{1-\delta}$ $\mathbf{r}_\delta = \mathbf{y}_\delta - a_\delta\zeta^{\mathbf{c}_\delta}$	$(\mathbf{r}_0, \mathbf{r}_1, \mathbf{c}_0, \mathbf{c}_1)$ $\xrightarrow{\qquad\qquad}$	Accept if $\mathbf{c} = \mathbf{c}_0 + \mathbf{c}_1$ and $\forall\ b \in \{0,1\}$, $\boxed{\forall j \in \mathbb{Z}_d}$, $[\mathfrak{g}^{\mathbf{r}_b}] \star A_b^{\mathbf{c}_b} = \mathbf{Y}_b$ $\boxed{[\mathfrak{g}^{\mathbf{r}_b\zeta^j}] \star A_b^{[\mathbf{c}_b+\mathbf{j}]_d} = \mathbf{Y}_b^j}$

Figure 6.1: The base sigma protocol with a large challenge space, where the box is to be ignored. Recall $\mathbb{Z}_d = \{0, 1, \ldots, d-1\}$. A_b^j denotes $[\mathfrak{g}^{a_b\zeta^j}] \star E_0$ for $j \in \mathbb{Z}_d$ and the vector $A_b^{[\mathbf{c}]_d}$ denotes $(A_b^{[c_0]_d}, \ldots, A_b^{[c_{\kappa-1}]_d})$ where $\mathbf{c} = (c_0, \ldots, c_{\kappa-1}) \in \mathbb{Z}^\kappa$. If $\mathbf{c} \in \mathbb{Z}_d^\kappa$, then $A_b^{[\mathbf{c}]_d}$ is simply $A_b^{\mathbf{c}}$. Other notations are explained at the beginning of Sec. 6.2. The base sigma protocol can be made compatible with blind signatures by running the boxed lines instead of the preceding non-boxed lines.

Correctness.

It suffices to show the equation.

$$[\mathfrak{g}^{\mathbf{r}_b}] \star A_b^{\mathbf{c}_b} = \mathbf{Y}_b \tag{6.1}$$

for $b \in \{0,1\}$. For the case $b = 1-\delta$, the equation holds naturally. For the case $b = \delta$, we have

$$\begin{aligned}
[\mathfrak{g}^{\mathbf{r}_\delta}] \star A_\delta^{\mathbf{c}_\delta} &= [\mathfrak{g}^{\mathbf{y}_\delta - a_\delta\zeta^{\mathbf{c}_\delta}}] \star A_\delta^{\mathbf{c}_\delta} \\
&= [\mathfrak{g}^{\mathbf{y}_\delta - a_\delta\zeta^{\mathbf{c}_\delta}}] \star ([\mathfrak{g}^{a_\delta\zeta^{\mathbf{c}_\delta}}] \star E_0) \\
&= \mathbf{Y}_\delta,
\end{aligned}$$

where we use the fact that $A_\delta^c = [\mathfrak{g}^{a_\delta\zeta^c}] \star E_0$ for any $c \in \mathbb{Z}_d$.

HVZK.

Given a challenge $\mathbf{c} \in \mathbb{Z}_d^\kappa$ and the instance $\mathsf{X} = ((A_0^j)_{j\in\mathbb{Z}_d}, (A_1^j)_{j\in\mathbb{Z}_d})$, a zero-knowledge simulator Sim samples random $(\mathbf{c}_0, \mathbf{c}_1) \xleftarrow{\$} \mathbb{Z}_d^\kappa$ conditioned on $\mathbf{c}_0 + \mathbf{c}_1 = \mathbf{c}$. Then, for each $b \in \{0,1\}$, the simulator generates $\mathbf{r}_b \xleftarrow{\$} \mathbb{Z}_N^\kappa$ and $\mathbf{Y}_b = [\mathfrak{g}^{\mathbf{r}_b}] \star A_b^{\mathbf{c}_b}$, and outputs $((\mathbf{Y}_0, \mathbf{Y}_1), \mathbf{c}, (\mathbf{r}_0, \mathbf{r}_1, \mathbf{c}_0, \mathbf{c}_1))$. Since there is a bijection between $\mathbf{r}_b$ and $\mathbf{Y}_b$ once $\mathbf{c}_b$ is fixed, this produces a transcript identically distributed as a real transcript.

Witness Indistinguishable.

This is a direct consequence of the above since perfect HVZK implies perfect witness indistinguishability as explained in Rem. 3.3.5.

Special Soundness.

It suffices to show that special soundness holds for $\kappa = 1$. Let $((Y_0, Y_1), c, (r_0, r_1, c_0, c_1))$, and $((Y_0, Y_1), c', (r_0', r_1', c_0', c_1'))$ be the two valid transcripts. Since $c = c_0 + c_1$, $c = c_0' + c_1'$ and $c \neq c'$, we assume $c_0 \neq c_0'$ without loss of generality. We have $r_0, r_0' \in \mathbb{Z}_N$, and distinct $c_0, c_0' \in \mathbb{Z}_d$ with relations $r_0 = y + a_0\zeta_d^{c_0}$, $r_0' = y + a_0\zeta_d^{c_0'} \pmod N$ where y, a_0 are unknown. Since we assume Requirement 3 holds, we can use the extractor $\mathsf{Ext}'(r_0, r_0', c_0, c_0')$ in Lem. 6.4.1 to obtain a list of size $\eta = \mathsf{lcm}_{i\in[d-1]}(\mathsf{gcd}(\zeta^i - 1, N)) = \mathsf{poly}$ containing $a_0 \in \mathbb{Z}_N$ in polynomial time. We can find a_0 from the list by running through each element in the list and checking if it maps to the statement $(A_0^j)_{j\in\mathbb{Z}_d}$ or $(A_1^j)_{j\in\mathbb{Z}_d}$.

Here, we implicitly assume the statement is honestly generated and that this check always terminates.

6.4.4 Enhancing the Base Sigma Protocol for Blind Signatures

Before explaining our blind signature, we make a subtle but important modification to our base sigma protocol. Looking ahead, this modification enables us to prove perfect blindness for our blind signature (refer to Rem. 6.4.4 for more precise use of this modification). To understand this modification, notice that if we tried to use a similar idea as in the prior sections to blind $\mathbf{Y}_b = [\mathfrak{g}^{\mathbf{y}_b}] \star E_0$ for $b \in \{0,1\}$, the user must randomize it to a value $[\mathfrak{g}^{\mathbf{z}_b}] \star ([\mathfrak{g}^{\mathbf{y}_b\zeta^{\mathbf{d}_b}}] \star E_0)$, where $(\mathbf{z}_b, \mathbf{d}_b) \xleftarrow{\$} \mathbb{Z}_N^\kappa \times \mathbb{Z}_d^\kappa$. This was doable when $d = 2$, since $\zeta = -1$ and $[\mathfrak{g}^{\mathbf{y}_b\zeta^{\mathbf{d}_b}}] \star E_0$ is simply the quadratic twist of $\mathbf{Y}_b$. However, in general, such a computation cannot be performed. To this end, we let the prover include components that will later help the user in the blind signature. This extension to our basic sigma protocol is illustrated in Fig. 6.1, where the box represents the modification. The prover sends $[\mathfrak{g}^{\mathbf{y}_b\zeta^j}] \star E_0$ for all $j \in \mathbb{Z}_d$ so that the user in the blind signature can choose whichever one based on the $\mathbf{d}_d$ it samples. We also modify the verifier of the base sigma protocol to check that $[\mathfrak{g}^{\mathbf{y}_b\zeta^{\mathbf{d}_b}}] \star E_0$ were generated correctly.

Below, we show that the extended sigma protocol satisfies correctness and HVZK. Since the extended sigma protocol includes the transcript of the base sigma protocol, special soundness is inherited.

Correctness.

It suffices to show that

$$[\mathfrak{g}^{\mathbf{r}_b \zeta^j}] \star A_b^{[\mathbf{c}_b + \mathbf{j}]_d} = \mathbf{Y}_b^j$$

for any $(b, j) \in \{0,1\} \times \mathbb{Z}_d$. For the case $b = 1 - \delta$, the equation holds by definition. For the case $b = \delta$, we have

$$\begin{aligned} [\mathfrak{g}^{\mathbf{r}_\delta \zeta^j}] \star A_\delta^{[\mathbf{c}_\delta + \mathbf{j}]_d} &= [\mathfrak{g}^{\mathbf{y}_\delta \zeta^j - a_\delta \zeta^{\mathbf{c}_\delta + \mathbf{j}}}] \star A_\delta^{[\mathbf{c}_\delta + \mathbf{j}]_d} \\ &= [\mathfrak{g}^{\mathbf{y}_\delta \zeta^j - a_\delta \zeta^{\mathbf{c}_\delta + \mathbf{j}}}] \star ([\mathfrak{g}^{a_\delta \zeta^{\mathbf{c}_\delta + \mathbf{j}}}] \star E_0) \\ &= \mathbf{Y}_\delta^j, \end{aligned}$$

where we use the fact that $A_\delta^{[c]_d} = [\mathfrak{g}^{a_\delta \zeta^c}] \star E_0$ for any $c \in \mathbb{Z}$.

HVZK.

Given a challenge $\mathbf{c} \in \mathbb{Z}_d^\kappa$ and the instance $\mathsf{X} = ((A_0^j)_{j \in \mathbb{Z}_d}, (A_1^j)_{j \in \mathbb{Z}_d})$, a zero-knowledge simulator Sim samples random $(\mathbf{c}_0, \mathbf{c}_1) \xleftarrow{\$} \mathbb{Z}_d^\kappa$ conditioned on $\mathbf{c}_0 + \mathbf{c}_1 = \mathbf{c}$. Then, for each $(b, j) \in \{0,1\} \times \mathbb{Z}_d$, the simulator generates $\mathbf{r}_b \xleftarrow{\$} \mathbb{Z}_N^\kappa$ and $\mathbf{Y}_b^j = [\mathfrak{g}^{\mathbf{r}_b \zeta^j}] \star A_b^{[\mathbf{c}_b + \mathbf{j}]_d}$, and outputs $((\mathbf{Y}_0^j, \mathbf{Y}_1^j)_{j \in \mathbb{Z}_d}, \mathbf{c}, (\mathbf{r}_0, \mathbf{r}_1, \mathbf{c}_0, \mathbf{c}_1))$ Since for every $j \in \mathbb{Z}_d$, there is a bijection between $\mathbf{r}_b$ and $\mathbf{Y}_b^j$ once $\mathbf{c}_b$ is fixed, this produces a transcript identically distributed as a real transcript.

6.4.5 Description of Our Blind Signature

We present our isogeny-based/group-action-based blind signature – Otters [8] – building on top of the enhanced base sigma protocol in Sec. 6.4.3. Let $(G, \mathcal{E}, E_0, \star)$ be the public parameter as the known-order effective group Def. 2.3.4 where G is cyclic of order N with a public generator $\mathfrak{g}$. Let ζ to be a d-th root of unity. We assume these parameters are provided to all algorithms. The parameter $\kappa \in \mathbb{N}$ indicates the number of repetitions of the underlying sigma protocol such that $d^\kappa \geq 2^\lambda$. Let $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_d^\kappa$ be a hash function modeled as a random oracle in the security proof.

The following algorithms are summarized in Fig. 6.2.

$\mathsf{BS.KGen}\left(1^\lambda\right)$: On input the security parameter 1^λ, it samples a bit $\delta \xleftarrow{\$} \{0,1\}$, $(a_0, a_1) \xleftarrow{\$} \mathbb{Z}_N^2$, and outputs a public key $\mathsf{pk} = ((A_0^j)_{j \in \mathbb{Z}_d}, (A_1^j)_{j \in \mathbb{Z}_d}) \subseteq \mathcal{PK} = \mathcal{E}^{2d}$ where $A_b^j = [\mathfrak{g}^{a_b \zeta^j}] * E_0$ for $(b, j) \in \{0,1\} \times \mathbb{Z}_d$, and secret key $\mathsf{sk} = (\delta, a_\delta)$.

[8] <u>O</u>r-proof <u>T</u>wisted <u>T</u>hre<u>E</u>-<u>R</u>ound <u>S</u>ignature.

$\mathsf{BS.S}_1(\mathsf{sk})$: The signer first samples $\mathbf{y}^*_\delta \xleftarrow{\$} \mathbb{Z}_N^\kappa$ and sets $\mathbf{Y}^{j*}_\delta = [\mathfrak{g}^{\mathbf{y}^*_\delta \zeta^j}] \star E_0$ for $j \in \mathbb{Z}_d$. It then samples $(\mathbf{c}^*_{1-\delta}, \mathbf{r}^*_{1-\delta}) \xleftarrow{\$} \mathbb{Z}_d^\kappa \times \mathbb{Z}_N^\kappa$ and sets $\mathbf{Y}^{j*}_{1-\delta} = [\mathfrak{g}^{\mathbf{r}^*_{1-\delta}\zeta^j}] \star A_{1-\delta}^{\mathbf{c}^*_{1-\delta}+\mathbf{j}}$ for $j \in \mathbb{Z}_d$. It then outputs the signer state $\mathsf{st}_\mathsf{S} = (\mathbf{y}^*_\delta, \mathbf{c}^*_{1-\delta}, \mathbf{r}^*_{1-\delta})$ and the first-sender message $\rho_{\mathsf{S},1} = (\mathbf{Y}_0^{j*}, \mathbf{Y}_1^{j*})_{j \in \mathbb{Z}_d}$.

$\mathsf{BS.U}_1(\mathsf{pk}, \mathsf{M}, \rho_{\mathsf{S},1})$: The user parses $(\mathbf{Y}_0^{j*}, \mathbf{Y}_1^{j*})_{j \in \mathbb{Z}_d} \leftarrow \rho_{\mathsf{S},1}$, samples $(\mathbf{d}_b, \mathbf{z}_b) \xleftarrow{\$} \mathbb{Z}_d^\kappa \times \mathbb{Z}_N^\kappa$, and computes $\mathbf{Z}_b = [\mathfrak{g}^{\mathbf{z}_b}] \star (Y^{*\,d_{b,0}}_{b,0}, \ldots, Y^{*\,d_{b,\kappa-1}}_{b,\kappa-1})$ for $b \in \{0,1\}$. Here, note that $Y_{b,j}^{d_{b,j}*}$ denotes the j-th ($j \in \mathbb{Z}_d$) element of $\mathbf{Y}_b^{d_{b,j}*} \in \mathcal{E}^\kappa$ and $d_{b,j}$ is the j-th element of $\mathbf{d}_b \in \mathbb{Z}_d^\kappa$. It then computes $\mathbf{c} = \mathsf{H}(\mathbf{Z}_0 \| \mathbf{Z}_1 \| \mathsf{M}) \in \mathbb{Z}_d^\kappa$ and outputs the user state $\mathsf{st}_\mathsf{U} = (\mathbf{d}_0, \mathbf{d}_1, \mathbf{z}_0, \mathbf{z}_1)$ and user message $\rho_\mathsf{U} = \mathbf{c}^* = \mathbf{c} - \mathbf{d}_0 - \mathbf{d}_1$.

$\mathsf{BS.S}_2(\mathsf{st}_\mathsf{S}, \rho_\mathsf{U})$: The signer parses $(\mathbf{y}^*_\delta, \mathbf{c}^*_{1-\delta}, \mathbf{r}^*_{1-\delta}) \leftarrow \mathsf{st}_\mathsf{S}$, $\mathbf{c}^* \leftarrow \rho_\mathsf{U}$, sets $\mathbf{c}^*_\delta = \mathbf{c}^* + \mathbf{c}^*_{1-\delta} \in \mathbb{Z}_d^\kappa$, and computes $\mathbf{r}^*_\delta = \mathbf{y}^*_\delta - a_\delta \zeta^{\mathbf{c}^*_\delta} \in \mathbb{Z}_N^\kappa$.It then outputs the second-signer message $\rho_{\mathsf{S},2} = (\mathbf{c}^*_0, \mathbf{c}^*_1, \mathbf{r}^*_0, \mathbf{r}^*_1)$.

$\mathsf{BS.U}_2(\mathsf{st}_\mathsf{U}, \rho_{\mathsf{S},2})$: The user parses $(\mathbf{d}_0, \mathbf{d}_1, \mathbf{z}_0, \mathbf{z}_1) \leftarrow \mathsf{st}_\mathsf{U}$, $(\mathbf{c}^*_0, \mathbf{c}^*_1, \mathbf{r}^*_0, \mathbf{r}^*_1) \leftarrow \rho_{\mathsf{S},2}$ and checks if $[\mathfrak{g}^{\mathbf{r}^*_b \zeta^j}] \star A_b^{[\mathbf{c}^*_b + \mathbf{j}]_d} = \mathbf{Y}_b^{j*}$ holds for all $(b, j) \in \{0,1\} \times \mathbb{Z}_d$. If not, it outputs $\perp$. Otherwise, it sets $(\mathbf{c}_b, \mathbf{r}_b) = (\mathbf{c}^*_b + \mathbf{d}_b, \mathbf{z}_b + \mathbf{r}^*_b \zeta^{\mathbf{d}_b}) \in \mathbb{Z}_d^\kappa \times \mathbb{Z}_N^\kappa$ for $b \in \{0,1\}$. It then checks if

$$\mathbf{c}_0 + \mathbf{c}_1 = \mathsf{H}\Big([\mathfrak{g}^{\mathbf{r}_0}] \star A_0^{\mathbf{c}_0} \| [\mathfrak{g}^{\mathbf{r}_1}] \star A_1^{\mathbf{c}_1} \| \mathsf{M}\Big). \tag{6.2}$$

If it holds, it outputs a signature $\sigma = (\mathbf{c}_0, \mathbf{c}_1, \mathbf{r}_0, \mathbf{r}_1) \in (\mathbb{Z}_d^\kappa)^2 \times (\mathbb{Z}_N^\kappa)^2$, and otherwise $\perp$.

$\mathsf{BS.}V(\mathsf{pk}, \mathsf{M}, \sigma)$: The verifier outputs 1 if Eq. (6.2) holds, and otherwise 0.

6.4.6 Proof of Correctness and Blindness

The subsection shows that our blind signature presented in Sec. 6.4.5 has (perfect) correctness and blindness.

Theorem 6.4.2. *The blind signature scheme in Figure 6.2 is (perfectly) correct.*

Proof. To show correctness, it suffices to show the equation

$$\mathbf{c}_0 + \mathbf{c}_1 = \mathsf{H}\left([\mathfrak{g}^{\mathbf{r}_0}] \star A_0^{\mathbf{c}_0} \;\|\; [\mathfrak{g}^{\mathbf{r}_1}] \star A_1^{\mathbf{c}_1} \;\|\; \mathsf{M}\right)$$

holds when both the signer and user follow the protocol.

From the description of $\mathsf{BS.U}_1, \mathsf{BS.S}_2$ and $\mathsf{BS.U}_2$, we have $\mathbf{c} = \mathbf{c}^* + \mathbf{d}_0 + \mathbf{d}_1$, $\mathbf{c}^* = \mathbf{c}^*_1 + \mathbf{c}^*_2$, and $\mathbf{c}_b = \mathbf{c}^*_b + \mathbf{d}_b$ for $b \in \{0,1\}$. Therefore, we have $\mathbf{c} = \mathbf{c}_0 + \mathbf{c}_1$, which shows the l.h.r. equation. It remains to show $\mathbf{Z}_b = [\mathfrak{g}^{\mathbf{r}_b}] \star A_b^{\mathbf{c}_b}$ for each $b \in \{0,1\}$. Following the definition of $\mathbf{Z}_b$ computed

```
$\mathsf{BS.KGen}(1^\lambda)$
101 :  $(a_0, a_1, \delta) \xleftarrow{\$} \mathbb{Z}_N^2 \times \{0,1\}$
102 :  $(A_0^j)_{j\in\mathbb{Z}_d} \leftarrow ([\mathfrak{g}^{a_0\zeta^j}] * E_0)_{j\in\mathbb{Z}_d}$
103 :  $(A_1^j)_{j\in\mathbb{Z}_d} \leftarrow ([\mathfrak{g}^{a_1\zeta^j}] * E_0)_{j\in\mathbb{Z}_d}$
104 :  $\mathsf{pk} \leftarrow ((A_0^j)_{\in\mathbb{Z}_d}, (A_1^j)_{j\in\mathbb{Z}_d})$
105 :  $\textbf{return}\ (\mathsf{pk}, \mathsf{sk} = (\delta, a_\delta))$
```

```
$\mathsf{BS.S}_1(\mathsf{sk})$
201 :  $\textbf{parse}\ (\delta, a_\delta) \leftarrow \mathsf{sk}$
202 :  $\mathbf{y}_\delta^* \xleftarrow{\$} \mathbb{Z}_N^\kappa$
203 :  $(\mathbf{c}_{1-\delta}^*, \mathbf{r}_{1-\delta}^*) \xleftarrow{\$} \mathbb{Z}_d^\kappa \times \mathbb{Z}_N^\kappa$
204 :  $\textbf{for}\ j \in \mathbb{Z}_d$
205 :      $\mathbf{Y}_\delta^{j*} = [\mathfrak{g}^{\mathbf{y}_\delta^*\zeta^j}] \star E_0$
206 :      $\mathbf{Y}_{1-\delta}^{j*} = [\mathfrak{g}^{\mathbf{r}_{1-\delta}^*\zeta^j}] * A_{1-\delta}^{\mathbf{c}_{1-\delta}^*+\mathbf{j}}$
207 :  $\mathsf{st}_\mathsf{S} = (\mathbf{y}_\delta^*, \mathbf{c}_{1-\delta}^*, \mathbf{r}_{1-\delta}^*)$
208 :  $\rho_{\mathsf{S},1} = (\mathbf{Y}_0^{j*}, \mathbf{Y}_1^{j*})_{j\in\mathbb{Z}_d}$
209 :  $\textbf{return}\ (\mathsf{st}_\mathsf{S}, \rho_{\mathsf{S},1})$
```

```
$\mathsf{BS.U}_1(\mathsf{pk}, \mathsf{M}, \rho_{\mathsf{S},1})$
301 :  $\textbf{parse}\ (\mathbf{Y}_0^{j*}, \mathbf{Y}_1^{j*})_{j\in\mathbb{Z}_d} \leftarrow \rho_{\mathsf{S},1}$
302 :  $\textbf{for}\ b \in \{0,1\}\ \textbf{do}$
303 :      $(\mathbf{d}_b, \mathbf{z}_b) \xleftarrow{\$} \mathbb{Z}_d^\kappa \times \mathbb{Z}_N^\kappa$
304 :      $\mathbf{Z}_b = [\mathfrak{g}^{\mathbf{z}_b}] \star (Y^{*d_{b,0}}_{b,0}, \ldots, Y^{*d_{b,\kappa-1}}_{b,\kappa-1})$
305 :  $\mathbf{c}\ = \mathsf{H}(\mathbf{Z}_0 || \mathbf{Z}_1 || \mathsf{M})$
306 :  $\mathbf{c}^* = \mathbf{c} - \mathbf{d}_0 - \mathbf{d}_1 \in \mathbb{Z}_d^\kappa$
307 :  $\mathsf{st}_\mathsf{U} \leftarrow (\mathbf{d}_0, \mathbf{d}_1, \mathbf{z}_0, \mathbf{z}_1)$
308 :  $\textbf{return}\ (\mathsf{st}_\mathsf{U}, \rho_\mathsf{U} = \mathbf{c}^*)$
```

```
$\mathsf{BS.S}_2(\mathsf{st}_\mathsf{S}, \rho_\mathsf{U})$
401 :  $\textbf{parse}\ (\mathbf{y}_\delta^*, \mathbf{c}_{1-\delta}^*, \mathbf{r}_{1-\delta}^*) \leftarrow \mathsf{st}_\mathsf{S}$
402 :  $\textbf{parse}\ \mathbf{c}^* \leftarrow \rho_\mathsf{U}$
403 :  $\mathbf{c}_\delta^* \leftarrow \mathbf{c}^* - \mathbf{c}_{1-\delta}^*$
404 :  $\mathbf{r}_\delta^* \leftarrow \mathbf{y}_\delta^* - a_\delta \zeta^{\mathbf{c}_\delta^*}$
405 :  $\textbf{return}\ \rho_{\mathsf{S},2} = (\mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{r}_0^*, \mathbf{r}_1^*)$
```

```
$\mathsf{BS.U}_2(\mathsf{st}_\mathsf{U}, \rho_{\mathsf{S},2})$
501 :  $\textbf{parse}\ (\mathbf{d}_0, \mathbf{d}_1, \mathbf{z}_0, \mathbf{z}_1) \leftarrow \mathsf{st}_\mathsf{U}$
502 :  $\textbf{parse}\ (\mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{r}_0^*, \mathbf{r}_1^*) \leftarrow \rho_{\mathsf{S},2}$
503 :  $\textbf{for}\ (b, j) \in \{0,1\} \times \mathbb{Z}_d$
504 :      $\textbf{if}\ [\mathfrak{g}^{\mathbf{r}_b^*\zeta^j}] \star A_b^{[\mathbf{c}_b^*+\mathbf{j}]_d} \neq \mathbf{Y}_b^{j*}$
505 :          $\textbf{return}\ \sigma = \perp$
506 :  $\textbf{for}\ b \in \{0,1\}$
507 :      $\mathbf{c}_b \leftarrow \mathbf{c}_b^* + \mathbf{d}_b$
508 :      $\mathbf{r}_b \leftarrow \mathbf{z}_b + \mathbf{r}_b^* \zeta^{\mathbf{d}_b}$
509 :  $\mathbf{c}' = \mathsf{H}([\mathfrak{g}^{\mathbf{r}_0}] \star A_0^{\mathbf{c}_0} \parallel [\mathfrak{g}^{\mathbf{r}_1}] \star A_1^{\mathbf{c}_1} \parallel \mathsf{M})$
510 :  $\textbf{if}\ \mathbf{c}_0 + \mathbf{c}_1 = \mathbf{c}'$
511 :      $\textbf{return}\ \sigma = (\mathbf{c}_0, \mathbf{c}_1, \mathbf{r}_0, \mathbf{r}_1)$
512 :  $\textbf{return}\ \sigma = \perp$
```

```
$\mathsf{BS.V}(\mathsf{pk}, \mathsf{M}, \sigma)$
601 :  $\textbf{parse}\ (\mathbf{c}_0, \mathbf{c}_1, \mathbf{r}_0, \mathbf{r}_1) \leftarrow \sigma$
602 :  $\mathbf{c}' = \mathsf{H}([\mathfrak{g}^{\mathbf{r}_0}] \star A_0^{\mathbf{c}_0} \parallel [\mathfrak{g}^{\mathbf{r}_1}] \star A_1^{\mathbf{c}_1} \parallel \mathsf{M})$
603 :  $\textbf{if}\ \mathbf{c}_0 + \mathbf{c}_1 = \mathbf{c}'$
604 :      $\textbf{return}\ 1$
605 :  $\textbf{return}\ 0$
```

Figure 6.2: Our blind signature, Otters, where H is a hash function and ζ is a d-th primitive root of unity. Recall $\mathbb{Z}_d = \{0, 1, \ldots, d-1\}$ and that we use the notations $\mathbf{d} = (d_0, \ldots, d_{\kappa-1}) \in \mathbb{Z}_d^\kappa$ and $\mathbf{Y}^j = (Y_0^j, \ldots, Y_{\kappa-1}^j) \in \mathcal{E}^\kappa$. Moreover, if $\mathbf{c} \in \mathbb{Z}_d^\kappa$, then $A_b^{[\mathbf{c}]_d}$ is simply $A_b^{\mathbf{c}}$ for $b \in \{0,1\}$. See the caption of Fig. 6.1 for further explanation on the notations.

by $\mathsf{BS.U}_1$, we have

$$\begin{aligned}
\mathbf{Z}_b &= [\mathfrak{g}^{\mathbf{z}_b}] \star (Y^{*d_{b,0}}_{b,0}, \dots, Y^{*d_{b,\kappa-1}}_{b,\kappa-1}) \\
&= [\mathfrak{g}^{\mathbf{z}_b}] \star ([\mathfrak{g}^{r^*_{b,0}\zeta^{d_{b,0}}}] \star A_b^{[c^*_{b,0}+d_{b,0}]_d}, \dots, [\mathfrak{g}^{r^*_{b,\kappa-1}\zeta^{d_{b,\kappa-1}}}] \star A_b^{[c^*_{b,\kappa-1}+d_{b,\kappa-1}]_d}) \qquad (6.3) \\
&= ([\mathfrak{g}^{z_{b,0}+r^*_{b,0}\zeta^{d_{b,0}}}] \star A_b^{[c^*_{b,0}+d_{b,0}]_d}, \dots, [\mathfrak{g}^{z_{b,\kappa-1}+r^*_{b,\kappa-1}\zeta^{d_{b,\kappa-1}}}] \star A_b^{[c^*_{b,\kappa-1}+d_{b,\kappa-1}]_d}) \\
&= [\mathfrak{g}^{\mathbf{r}_b}] \star A_b^{\mathbf{c}_b}, \qquad (6.4)
\end{aligned}$$

where Eq. (6.3) follows from the check performed by $\mathsf{BS.U}_2$ and the correcntess of the underlying sigma protocol shown in Sec. 6.4.4, and Eq. (6.4) follows from the definition of $(\mathbf{c}_b, \mathbf{r}_b)$. □

Next, we will show the generalized blind signature has perfect blindness. Notably, blindness holds even under chosen keys. This is a strong property since if a malicious signer uses malformed supersingular curves in $\mathcal{E}$ without the ring structure as the public key, the user cannot detect this. The main reason why we can argue perfect blindness is that if the public key is malformed, then the pair of curves in the first message $(\mathbf{Y}_0^{j*}, \mathbf{Y}_1^{j*})_{j\in\mathbb{Z}_d}$ is also malformed in a controlled manner. If there exists one user state that leads to a valid signature, then we can argue that the first message must be in a specific (but possibly incorrect) form regardless of the user state. Using this, we are able to establish a bijection between an arbitrary user state and a valid signature conditioning on a fixed first and second signature messages and a user message. Namely, any valid signature could have been produced with an equal probability.

Theorem 6.4.3. *The blind signature scheme in Figure 6.2 is (perfectly) blind under chosen keys.*

Proof. Let $(\rho_{\mathsf{S},1,0}, \rho_{\mathsf{S},2,0})$ and $(\rho_{\mathsf{S},1,1}, \rho_{\mathsf{S},2,1})$ be the two sets of first and second-signer message pairs the adversary $\mathcal{A}$ queries to oracles U_1 and U_2. Moreover, let $\rho_{\mathsf{U},b}$ be the user message returned by oracle U_1 when $\mathcal{A}$ queries with $\rho_{\mathsf{S},1,b}$ for $b \in \{0,1\}$, and let $(\sigma_{\mathsf{coin}}, \sigma_{1-\mathsf{coin}})$ be the two signatures $\mathcal{A}$ sees at the end, where note that these two corresponds to $\widetilde{\mathsf{M}}_0$ and $\widetilde{\mathsf{M}}_1$, respectively, regardless of the choice of $\mathsf{coin} \in \{0,1\}$. We call $(\rho_{\mathsf{S},1,b}, \rho_{\mathsf{U},b}, \rho_{\mathsf{S},2,b})_{b\in\{0,1\}}$ the *view* of $\mathcal{A}$. To prove perfect blindness, it suffices to prove that the view is independent of $\mathsf{coin} \in \{0,1\}$. In other words, since the randomness used by oracle U_1 is defined by $(\mathsf{st}_{\mathsf{U},b})_{b\in\{0,1\}}$ and oracle U_2 is deterministic, we prove that there exist two sets of states $(\mathsf{st}^{(0)}_{\mathsf{U},b})_{b\in\{0,1\}}$ and $(\mathsf{st}^{(1)}_{\mathsf{U},b})_{b\in\{0,1\}}$ that can be sampled by oracle U_1 with an equal probability such that they generate the same view to $\mathcal{A}$ but produce a different pair of signatures (σ_0, σ_1) and (σ_1, σ_0), respectively. Considering that the set of valid signature space and user randomness/state space is identical, we prove a stronger statement that for any non-aborting (partial) view $(\rho_{\mathsf{S},1,0}, \rho_{\mathsf{U},0}, \rho_{\mathsf{S},2,0})$ of $\mathcal{A}$, there is a bijection between a valid signature σ_0 on message M_0 and a state $\mathsf{st}_{\mathsf{U},0}$ of the oracle U_1. Below, we drop the subscript 0 for readability.

Let us denote the first and second-signer message as $\rho_{\mathsf{S},1} = (\mathbf{Y}_0^{j*}, \mathbf{Y}_1^{j*})_{j\in\mathbb{Z}_d}$, $\rho_{\mathsf{S},2} = (\mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{r}_0^*, \mathbf{r}_1^*)$, a user message as $\rho_{\mathsf{U}} = \mathbf{c}^*$, and a valid signature for message M as $\sigma = (\mathbf{c}_0, \mathbf{c}_1, \mathbf{r}_0, \mathbf{r}_1) \in (\mathbb{Z}_d^\kappa)^2 \times (\mathbb{Z}_N^\kappa)^2$. Here, note that any public key $\mathsf{pk} = ((A_0^j)_{j\in\mathbb{Z}_d}, (A_1^j)_{j\in\mathbb{Z}_d})$ output by the adversary (i.e., malicious signer) $\mathcal{A}$ can be efficiently checked to be valid elliptic curves (i.e., supersingularity) but cannot be checked if it has the correct cyclic structure.

We define a map between the signature $\sigma = (\mathbf{c}_0, \mathbf{c}_1, \mathbf{r}_0, \mathbf{r}_1)$ and user state $\mathsf{st}_\mathsf{U} = (\mathbf{d}_0, \mathbf{d}_1, \mathbf{z}_0, \mathbf{z}_1)$ by $\mathbf{d}_b = \mathbf{c}_b - \mathbf{c}_b^*$ and $\mathbf{z}_b = \mathbf{r}_b - \mathbf{r}_b^* \cdot \zeta^{\mathbf{d}_b}$ for $b \in \{0,1\}$. It is easy to check that once the view (or $\rho_{\mathsf{S},2} = (\mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{r}_0^*, \mathbf{r}_1^*)$) is fixed, then this mapping is indeed a bijection. It remains to prove that this st_U is a state that produces σ.

Observe that when $\mathsf{BS.U}_1(\mathsf{pk}, \mathsf{M}, \rho_{\mathsf{S},1})$ runs with the state st_U, it computes $\mathbf{Z}_b = [\mathfrak{g}^{\mathbf{z}_b}] \star (Y^{*d_{b,0}}_{b,0}, \ldots, Y^{*d_{b,\kappa-1}}_{b,\kappa-1})$ for $b \in \{0,1\}$ using $\rho_{\mathsf{S},1}$. It then sets $\mathbf{c}' = \mathsf{H}(\mathbf{Z}_0 || \mathbf{Z}_1 || \mathsf{M})$ and defines $\rho'_\mathsf{U} = \mathbf{c}'^* = \mathbf{c}' - \mathbf{d}_0 - \mathbf{d}_1$. Moreover, due to restrictions on the blindness game, the view is non-aborting for at least one state st_U. Combining this with the fact that the first check performed by $\mathsf{BS.U}_2(\mathsf{st}_\mathsf{U}, \rho_{\mathsf{S},2})$ only depends on $\rho_{\mathsf{S},2}$, and in particular is independent of st_U, we have $[\mathfrak{g}^{\mathbf{r}_b^* \zeta^j}] \star A_b^{[\mathbf{c}_b^* + \mathbf{j}]_d} = \mathbf{Y}_b^{j^*}$ for $j \in \mathbb{Z}_d$ and any state st_U. Therefore, $\mathsf{BS.U}_2$ always outputs σ as desired since the signature σ is assumed to be valid.

It remains to check that $\rho'_\mathsf{U} = \mathbf{c}'^*$ generated by $\mathsf{BS.U}_1$ is the desired $\rho_\mathsf{U} = \mathbf{c}^*$ to complete the proof. Since σ is valid and due to the definition of st_U, we have $\mathbf{c}_0^* + \mathbf{c}_1^* + \mathbf{d}_0 + \mathbf{d}_1 = \mathsf{H}\left([\mathfrak{g}^{\mathbf{r}_0}] \star A_0^{\mathbf{c}_0} \ \| \ [\mathfrak{g}^{\mathbf{r}_1}] \star A_1^{\mathbf{c}_1} \ \| \ \mathsf{M}\right)$. Moreover, since the view is non-aborting, we are guaranteed that $\mathbf{c}^* = \mathbf{c}_0^* + \mathbf{c}_1^*$. Therefore, if $\mathbf{Z}_b = [\mathfrak{g}^{\mathbf{r}_b}] \star A_b^{\mathbf{c}_b}$ for $b \in \{0,1\}$, then we can conclude that $\mathbf{c}^* = \mathbf{c}'^*$ as desired. This can be checked as follows, where we use the fact that $[\mathfrak{g}^{\mathbf{r}_b^* \zeta^j}] \star A_b^{[\mathbf{c}_b^* + \mathbf{j}]_d} = \mathbf{Y}_b^{j^*}$ for $j \in \mathbb{Z}_d$ in the second equality:

$$
\begin{aligned}
\mathbf{Z}_b &= [\mathfrak{g}^{\mathbf{z}_b}] \star (Y^{*d_{b,0}}_{b,0}, \ldots, Y^{*d_{b,\kappa-1}}_{b,\kappa-1}) \\
&= [\mathfrak{g}^{\mathbf{z}_b}] \star ([\mathfrak{g}^{r^*_{b,0}\zeta^{d_{b,0}}}] \star A_b^{[c^*_{b,0}+d_{b,0}]_d}, \ldots, [\mathfrak{g}^{r^*_{b,\kappa-1}\zeta^{d_{b,\kappa-1}}}] \star A_b^{[c^*_{b,\kappa-1}+d_{b,\kappa-1}]_d}) \\
&= ([\mathfrak{g}^{z_{b,0}+r^*_{b,0}\zeta^{d_{b,0}}}] \star A_b^{[c^*_{b,0}+d_{b,0}]_d}, \ldots, [\mathfrak{g}^{z_{b,\kappa-1}+r^*_{b,\kappa-1}\zeta^{d_{b,\kappa-1}}}] \star A_b^{[c^*_{b,\kappa-1}+d_{b,\kappa-1}]_d}) \\
&= [\mathfrak{g}^{\mathbf{r}_b}] \star A_b^{\mathbf{c}_b}.
\end{aligned}
$$

This completes the proof.

□

Remark 6.4.4. *As can be observed from the second equation right above, the verification of the base sigma protocol (indicated by the boxed lines in Fig. 6.1) is crucial for achieving perfect blindness. Without this additional verification step, perfect blindness would not be guaranteed.*

6.4.7 Proof of One-More Unforgeability

Our proof of OMUF consists of preparing the necessary tools present in Sec. 6.3 to invoke Thm. 6.3.10. Specifically, we define instances $\mathbf{I}_0, \mathbf{I}_1$ (see Def. 6.3.1), the map $\Phi_{\mathsf{rand},\vec{h}}$ (see Def. 6.3.8), the witness extractors $(\mathsf{Ext}_0, \mathsf{Ext}_1)$ (see Def. 6.3.9) and prove that Requirements 1 and 2 hold.

Preparation: Instances.

Let us first define the **0**-side instance $\mathbf{I}_0$ and the **1**-side instance $\mathbf{I}_1$. Below, we assume the adversary against the one-more unforgeability game makes ℓ-signing queries in total.

A **0**-side instance $\mathbf{I}_0 = (0, a_0, \mathbf{A}_1, \overrightarrow{\mathbf{y}_0^*}, \overrightarrow{\mathbf{r}_1^*}, \overrightarrow{\mathbf{c}_1^*})$ is defined as follows:

- $(0, a_0)$: The secret key sk when $\delta = 0$.
- $\mathbf{A}_1$: The part of the public key $\mathsf{pk} = (\mathbf{A}_0 = (A_0^j)_{j\in\mathbb{Z}_d}, \mathbf{A}_1 = (A_1^j)_{j\in\mathbb{Z}_d})$ whose secret key (i.e. the rGAIP solution) is unknown.
- $\mathbf{y}_0^{*(k)}$: The exponent of the commitment $(\mathbf{Y}_0^{j*(k)})_{j\in\mathbb{Z}_d}$ in the k-th ($k \in [\ell]$) first-sender message when $\delta = 0$ such that $\mathbf{Y}_0^{j*(k)} = [\mathfrak{g}^{\mathbf{y}_0^{*(k)}\zeta^j}] \star E_0$ for each $j \in \mathbb{Z}_d$.
- $\mathbf{c}_1^{*(k)}$: The simulated challenge (i.e. $\mathbf{c}_1^*$ in Fig. 6.2) in the k-th ($k \in [\ell]$) first-sender message when $\delta = 0$.
- $\mathbf{r}_1^{*(k)}$: The exponent of the commitment $\mathbf{Y}_1^{j*(k)}$ in the k-th ($k \in [\ell]$) first-sender message when $\delta = 0$ such that $\mathbf{Y}_1^{j*(k)} = [\mathfrak{g}^{\mathbf{r}_1^{*(k)}\zeta^j}] \star A_1^{[\mathbf{c}_1^{*(k)}+\mathbf{j}]_d}$ for each $j \in \mathbb{Z}_d$.

A **1**-side instance $\mathbf{I}_1 = (1, a_1, \mathbf{A}_0, \overrightarrow{\mathbf{y}_1^*}, \overrightarrow{\mathbf{r}_0^*}, \overrightarrow{\mathbf{c}_0^*})$ is defined as follows:

- $(1, a_1)$: The secret key sk when $\delta = 1$.
- $\mathbf{A}_0$: The part of the public key $\mathsf{pk} = (\mathbf{A}_0 = (A_0^j)_{j\in\mathbb{Z}_d}, \mathbf{A}_1 = (A_1^j)_{j\in\mathbb{Z}_d})$ whose secret key (i.e. the rGAIP solution) is unknown.
- $\mathbf{y}_1^{*(k)}$: The exponent of the commitment $(\mathbf{Y}_1^{j*(k)})_{j\in\mathbb{Z}_d}$ in the k-th ($k \in [\ell]$) first-sender message when $\delta = 1$ such that $(\mathbf{Y}_1^{j*(k)})_{j\in\mathbb{Z}_d} = [\mathfrak{g}^{\mathbf{y}_1^{*(k)}\zeta^j}] \star E_0$ for each $j \in \mathbb{Z}_d$.
- $\mathbf{c}_0^{*(k)}$: The simulated challenge (i.e. $\mathbf{c}_0^*$ in Fig. 6.2) in the k-th ($k \in [\ell]$) first-sender message when $\delta = 1$.
- $\mathbf{r}_0^{*(k)}$: The exponent of the commitment $\mathbf{Y}_0^{j*(k)}$ in the k-th ($k \in [\ell]$) first-sender message when $\delta = 1$ such that $\mathbf{Y}_0^{j*(k)} = [\mathfrak{g}^{\mathbf{r}_0^{*(k)}\zeta^j}] \star A_0^{[\mathbf{c}_0^{*(k)}+\mathbf{j}]_d}$ for each $j \in \mathbb{Z}_d$.

Preparation: Map $\Phi_{\mathsf{rand},\overrightarrow{h}}$.

We next define the map $\Phi_{\mathsf{rand},\overrightarrow{h}}$ that maps a **0**-side instance $\mathbf{I}_0$ into a **1**-side instance $\mathbf{I}_1$ and vice versa. Concretely, a **0**-side instance $\mathbf{I}_0 = (0, a_0, \mathbf{A}_1, \overrightarrow{\mathbf{y}_0^*}, \overrightarrow{\mathbf{r}_1^*}, \overrightarrow{\mathbf{c}_1^*})$ maps to a **1**-side instance $\mathbf{I}_1$ such that

$$\begin{aligned}\mathbf{I}_1 = (\ 1, a_1,\ & A_0 = (A_0^j)_{j\in\mathbb{Z}_d} = ([\mathfrak{g}^{a_0\zeta^j}] * E_0)_{j\in\mathbb{Z}_d},\\ & \overrightarrow{\mathbf{y}_1^*} = \overrightarrow{\mathbf{r}_1^*} + a_1\zeta^{\overrightarrow{\mathbf{c}_1^*}},\ \overrightarrow{\mathbf{c}_0^*} = \overrightarrow{\mathbf{c}^*} - \overrightarrow{\mathbf{c}_1^*},\ \overrightarrow{\mathbf{r}_0^*} = \overrightarrow{\mathbf{y}_0^*} - a_0\zeta^{\overrightarrow{\mathbf{c}_0^*}}),\end{aligned}$$

where $a_1 \in \mathbb{Z}_N$ is such that $A_1^0 = [\mathfrak{g}^{a_1}] \star E_0$ and recall that $\overrightarrow{\mathbf{c}^*} = \overrightarrow{e}(\mathbf{I}_0, \mathsf{rand}, \overrightarrow{h})$. On the other hand, a **1**-side instance $\mathbf{I}_1 = (1, a_1, \mathbf{A}_0, \overrightarrow{\mathbf{y}_1^*}, \overrightarrow{\mathbf{r}_0^*}, \overrightarrow{\mathbf{c}_0^*})$ maps to a **0**-side instance $\mathbf{I}_0$ such that

$$\begin{aligned}\mathbf{I}_0 = (\ 0, a_0,\ & \mathbf{A}_1 = (A_1^j)_{j\in\mathbb{Z}_d} = ([\mathfrak{g}^{a_1\zeta^j}] * E_0)_{j\in\mathbb{Z}_d},\\ & \overrightarrow{\mathbf{y}_0^*} = \overrightarrow{\mathbf{r}_0^*} + a_0\zeta^{\overrightarrow{\mathbf{c}_0^*}},\ \overrightarrow{\mathbf{c}_1^*} = \overrightarrow{\mathbf{c}^*} - \overrightarrow{\mathbf{c}_0^*},\ \overrightarrow{\mathbf{r}_1^*} = \overrightarrow{\mathbf{y}_1^*} - a_1\zeta^{\overrightarrow{\mathbf{c}_1^*}})\end{aligned}$$

```
$\mathsf{Ext}_0(\sigma, \sigma')$
101 : if $\exists t \in [\kappa]$ s.t. $c_{0,t} \neq c'_{0,t}$
102 :   $L \leftarrow \mathsf{Ext}'(r_{0,t}, r'_{0,t}, c'_{0,t}, c_{0,t})$
103 :   for $a' \in L$
104 :     if $[\mathfrak{g}^{a'}] \star E_0 = A_0^0$
105 :       return $a'$
106 : return $\perp$
```

```
$\mathsf{Ext}_1(\sigma, \sigma')$
101 : if $\exists t \in [\lambda]$ s.t. $c_{1,t} \neq c'_{1,t}$
102 :   $L \leftarrow \mathsf{Ext}'(r_{1,t}, r'_{1,t}, c'_{1,t}, c_{1,t})$
103 :   for $a' \in L$
104 :     if $[\mathfrak{g}^{a'}] \star E_0 = A_1^0$
105 :       return $a'$
106 : return $\perp$
```

Figure 6.3: Witness extractors for our generalized blind signature for σ, σ'. In the above, $\sigma = (\mathbf{c}_0, \mathbf{c}_1, \mathbf{r}_0, \mathbf{r}_1)$ and $\sigma' = (\mathbf{c}'_0, \mathbf{c}'_1, \mathbf{r}'_0, \mathbf{r}'_1)$, where $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}'_0, \mathbf{c}'_1$ live in $\mathbb{Z}_d^\kappa$ and $\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}'_0, \mathbf{r}'_1$ live in $\mathbb{Z}_N^\kappa$. Ext' is the extractor in Lem. 6.4.1. Non-bold font indicates the entries of a vector.

where $a_0 \in \mathbb{Z}_N$ is such that $A_0^0 = [\mathfrak{g}^{a_0}] \star E_0$ and recall that $\overrightarrow{\mathbf{c}^*} = \overrightarrow{e}(\mathbf{I}_1, \mathsf{rand}, \overrightarrow{h})$.

Preparation: Witness Extractors ($\mathsf{Ext}_0, \mathsf{Ext}_1$).

Fix $\mathbf{I}, \mathsf{rand}$ and let $(\overrightarrow{h}, \overrightarrow{h}') \in F_i(\mathbf{I}, \mathsf{rand})$ for some $i \in [\ell + 1]$. Let $\sigma = (\mathbf{c}_0, \mathbf{c}_1, \mathbf{r}_0, \mathbf{r}_1), \sigma' = (\mathbf{c}'_0, \mathbf{c}'_1, \mathbf{r}'_0, \mathbf{r}'_1)$ be the signatures that correspond to $\mathbf{c}^{(i)}$ and $\mathbf{c}'^{(i)}$, respectively, where recall $\mathbf{c}^{(i)}$ (resp. $\mathbf{c}'^{(i)}$) is the i-th entry of $\overrightarrow{h}$ (resp. $\overrightarrow{h}'$) and the hash value of form $\mathsf{H}(\cdot \parallel \cdot \parallel \mathsf{M})$ (i.e., $\mathbf{c}$ in Fig. 6.2). In particular, we have $\mathbf{c}_0 + \mathbf{c}_1 = \mathbf{c}^{(i)}$ and $\mathbf{c}'_0 + \mathbf{c}'_1 = \mathbf{c}'^{(i)}$. We define the witness extractors $(\mathsf{Ext}_0, \mathsf{Ext}_1)$ as in Fig. 6.3.

The following lemma establishes the correctness of the witness extractors.

Lemma 6.4.5. $(\mathsf{Ext}_0, \mathsf{Ext}_1)$ *in Fig. 6.3 satisfy the definition of witness extractors in Def. 6.3.9.*

Proof. By the definition of $\mathrm{F}_i(\mathbf{I}, \mathsf{rand})$ (see Def. 6.3.3), we have $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in \mathsf{Succ}$ and $\mathbf{c}^{(i)} \neq \mathbf{c}'^{(i)}$. The former implies that the two signatures σ and σ' are valid. Concretely, we have

$$\mathbf{c}^{(i)} = \mathbf{c}_0 + \mathbf{c}_1 = \mathsf{H}\Big([\mathfrak{g}^{\mathbf{r}_0}] \star A_0^{\mathbf{c}_0} \parallel [\mathfrak{g}^{\mathbf{r}_1}] \star A_1^{\mathbf{c}_1} \parallel \mathsf{M}\Big)$$
$$\mathbf{c}'^{(i)} = \mathbf{c}'_0 + \mathbf{c}'_1 = \mathsf{H}\Big([\mathfrak{g}^{\mathbf{r}'_0}] \star A_0^{\mathbf{c}'_0} \parallel [\mathfrak{g}^{\mathbf{r}'_1}] \star A_1^{\mathbf{c}'_1} \parallel \mathsf{M}'\Big).$$

Moreover, since $\overrightarrow{h}$ and $\overrightarrow{h}'$ agree up to the i-th entry and the challenger and adversary's randomness are fixed, the input to the hash functions agree. Namely, we have

$$[\mathfrak{g}^{\mathbf{r}_0}] \star A_0^{\mathbf{c}_0} = [\mathfrak{g}^{\mathbf{r}'_0}] \star A_0^{\mathbf{c}'_0} \quad \wedge \quad [\mathfrak{g}^{\mathbf{r}_1}] \star A_1^{\mathbf{c}_1} = [\mathfrak{g}^{\mathbf{r}'_1}] \star A_1^{\mathbf{c}'_1} \quad \wedge \quad \mathsf{M} = \mathsf{M}'$$

Since $\mathbf{c}^{(i)} \neq \mathbf{c}'^{(i)}$, we must have $\mathbf{c}_0 \neq \mathbf{c}'_0$ or $\mathbf{c}_1 \neq \mathbf{c}'_1$. Thus, due to the special soundness of the underlying sigma protocol (see Sec. 6.4.3) one of Ext_0 or Ext_1 always outputs a valid secret key. This completes the proof. □

Proof of One-More Unforgeability.

We have the following two lemmas required to invoke the main theorem Thm. 6.4.8.

Lemma 6.4.6. *Requirement 1 holds for our definition of the map* $\Phi_{\mathsf{rand},\overrightarrow{h}}$ *above.*

Proof. Since the proof for the $\mathbf{0}$-side and $\mathbf{1}$-side instances $\mathbf{I}_0$ and $\mathbf{I}_1$ are analogous, we only consider the $\mathbf{0}$-side instance. For any $\mathsf{rand}, \overrightarrow{h}$, let us consider the query transcript $\overrightarrow{e}(\mathbf{I}_0, \mathsf{rand}, \overrightarrow{h}) = \overrightarrow{\mathbf{c}^*}$, i.e., the vector of user message ρ_U queries made by the adversary to the signing algorithm $\mathsf{BS.S}_2$. Since the underlying sigma protocol is perfectly witness indistinguishable due to perfect HVZK (see Rem. 3.3.5), for each $i \in [\ell]$ and $\mathbf{c}^{*(i)}$, there is a set of randomness that the signer with a secret key $(1, a_1)$ (i.e., a $\mathbf{1}$-side witness) could have used to produce the same view (i.e., first and second-signer messages) to the adversary. Concretely, this set of randomness is exactly those defined by $\Phi_{\mathsf{rand},\overrightarrow{h}}(\mathbf{I}_0)$. Hence, we have $\mathsf{trans}(\mathbf{I}_0, \mathsf{rand}, \overrightarrow{h}) = \mathsf{trans}(\Phi_{\mathsf{rand},\overrightarrow{h}}(\mathbf{I}_0), \mathsf{rand}, \overrightarrow{h})$ as desired. Moreover, it is easy to check that $\Phi_{\mathsf{rand},\overrightarrow{h}}(\Phi_{\mathsf{rand},\overrightarrow{h}}(\mathbf{I}_0))$ from the definition of $\Phi_{\mathsf{rand},\overrightarrow{h}}$. Hence, it is a bijection as desired. This completes the proof. □

Lemma 6.4.7. *Requirement 2 holds for our definition of the witness extractors* $(\mathsf{Ext}_0, \mathsf{Ext}_1)$ *Fig. 6.3.*

Proof. Since the proof of $\mathbf{0}$-side and $\mathbf{1}$-side is analogous, we only consider the $\mathbf{0}$-side case. We prove the lemma by contradiction. Suppose the $\mathbf{0}$-side witness can be extracted from the base $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ at index i, but cannot be extracted from either of the sides $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'')$ or $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'')$. By Lem. 6.4.5, the assumption holds if and only if $\mathbf{c}_0 = \mathbf{c}_0''$ and $\mathbf{c}_0' = \mathbf{c}_0''$. As a result, $\mathbf{c}_0 = \mathbf{c}_0'$. By Lem. 6.4.5, the $\mathbf{0}$-side witness cannot be extracted from $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$. However, this contradicts our assumption. □

Combining everything together, we obtain the following.

Theorem 6.4.8 (One-more Unforgeability)**.** *The partially blind signature scheme in Figure 6.2 is one-more unforgeable. More precisely, for all* $\ell \in \mathbb{N}$*, if there exists an adversary* $\mathcal{A}$ *that makes* Q *hash queries to the random oracle and breaks the* ℓ*-one more unforgeability of our scheme with advantage* $\epsilon_{\mathcal{A}} \geq \frac{C_1}{d^\kappa} \cdot \binom{Q}{\ell+1}$*, then there exists an algorithm* $\mathcal{B}$ *that breaks the* ζ-rGAIP *problem with advantage* $\epsilon_{\mathcal{B}} \geq C_2 \cdot \frac{\epsilon_{\mathcal{A}}^2}{\binom{Q}{\ell+1}^2 \cdot (\ell+1)^3}$ *for some universal positive constants* C_1 *and* C_2*. Note we use a* d*-th primitive root of unity* ζ *and* κ *denotes the number of parallel repetitions of the underlying sigma protocol.*

Proof. Upon receiving an rGAIP instance, the wrapper proceeds as described in Sec. 6.3.2. The proof follows from the above Lems. 6.4.6 and 6.4.7 and Thm. 6.3.10 (i.e., [KLX22a, Theorem 1]) and the result follows.

□

6.5 Analysis of Ring GAIP

This section analyzes the ζ_d-ring group action inverse problem (ζ_d-rGAIP) over CSIDH-512. Sec. 6.5.1 discusses the existence of ζ_d and the finding method. Sec. 6.5.2 recalls the most efficient classical and quantum algorithms against GAIP and presents a structural attack on ζ_d-rGAIP which effectively reduces ζ_d-rGAIP for a few choices of d to a GAIP instance with a smaller group size compared to the original group considered by ζ_d-rGAIP. In Sec. 6.5.3, we complement our cryptanalysis by proving that ζ_d-rGAIP for a few choices of d is as hard as GAIP defined over the same group. This shows optimality of our structural attack for ζ_d-rGAIP for some choices of d. We note that the concrete value of d's that admit an attack or a reduction depends on the concrete CSIDH-512 parameter sets.

6.5.1 Finding a Root of Unity Satisfying Requirement 3

We briefly discuss the existence of and a process for finding a primitive d-th root of unity $\zeta_d \in \mathbb{Z}_N^\times$ which satisfies Requirement 3. Firstly, it is a straightforward consequence of the fundamental theorem of finitely-generated abelian groups and the definition of $\lambda(N)$ that $\mathbb{Z}_N^\times \cong \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \times \cdots \times \mathbb{Z}_{n_r}$ where $n_1 \,|\, n_2 \,|\, \cdots \,|\, n_r$ and $n_r = \lambda(N)$, so that a d-th root of unity exists if and only if d is a divisor of $\lambda(N)$—here, $\lambda(\cdot)$ is the Carmichael function, whose formula is given in Sec. 6.2.2.

To find such a root for a given valid d, the most intuitive method, perhaps, is to start with a primitive $\lambda(N)$-th root of unity $\zeta_{\lambda(N)}$ where $\lambda(\cdot)$ is the Carmichael function, and compute $\zeta = \zeta_{\lambda(N)}^{\frac{\lambda(N)}{d}}$, which will have order exactly d. Unfortunately, this may result in a d-th root of unity that does not meet Requirement 3 (even when one exists which satisfies Requirement 3). In particular, we have to ensure that ζ is a generator modulo all but small prime power divisors of N to conclude $\eta_d = \mathsf{lcm}_{i\in[d-1]}(\mathsf{gcd}(\zeta^i - 1, N)) = \mathsf{poly}$. To this end, in every Sylow subgroup of $\mathbb{Z}_N^\times$, we find a generator of a cyclic subgroup of order d (if one exists) and use the Chinese remainder theorem to obtain a d-th root of unity. If a root meeting Requirement 3 exists, this method ensures finding such a root.

Concretely, for the CSIDH-512 parameter sets we have

$$\begin{aligned} N &= 3 \times 37 \times 1407181 \times 51593604295295867744293584889 \\ &\quad \times 31599414504681995853008278745587832204909, \\ \lambda(N) &= 2^3 \times 3^2 \times 5 \times 7^2 \times 47 \times 71 \times 499 \times 4387211249599988753766461 3 \\ &\quad \times 111265544030570407933127742061928986637, \end{aligned}$$

and we can construct the following primitive d-th roots of unity with respect to CSIDH-512 following this method for $2 \le d \le 9$, 47 and 499:

$$\begin{aligned}
\zeta_2 &= -1 \\
\zeta_3 &= 2477699437908495650371104512535948994004956355404732770089878647330138924903 49 \\
\zeta_4 &= 84724991146787019937735534381733959212289361891396363362098648465646877579 45 \\
\zeta_5 &= 72453024324688395187181869396509941269039951262689579224914692627674819998175 \\
\zeta_7 &= 72860468942899689738460495171518121784504211848373863183929808636917555788857 \\
\zeta_8 &= 17968081027951002862127994231802972521244754950515032766640065054960810210290 \\
\zeta_9 &= 14453246721132891293831442989793035798362245406527607825524292109425810395270 4 \\
\zeta_{47} &= 62847811803796095830053712564080163474854470329795797448561296882359337268 20 \\
\zeta_{499} &= 27716990710015300853542735667675665633828171067931279717294182935872148507972.
\end{aligned}$$

Remark 6.5.1. *In the list above, we only display d that is a prime power. For other composite divisors of $\lambda(N)$, one can obtain the corresponding root by multiplication. For instance, we can obtain $\zeta_{23453} = \zeta_{47}\zeta_{499}$.*

Concretely, for the CSIDH-512 parameter set, the totients of the small prime divisors of N have the following (maximal) small prime power divisors:

$$\begin{aligned}
\varphi(3) &: 2 \\
\varphi(37) &: 2^2,\ 3^2 \\
\varphi(1407181) &: 2^2,\ 3,\ 5,\ 47,\ 499 \\
\varphi(51593604295295867744293584889) &: 2^3,\ 3,\ 7^2 \\
\varphi(31599414504681995853008278745587832204909) &: 2^2,\ 71.
\end{aligned}$$

This implies that for the CSIDH-512 parameter we can only find a 4th root of unity meeting Requirement 3 (with $\eta_4 = 3$) since only $\mathbb{Z}_3^\times$ has no cyclic subgroups of order 4. For example, for any 3rd root of unity ζ_3, we always have a 134-bit divisor of $\mathsf{gcd}(\zeta_3 - 1, N)$. Therefore, ζ_4-rGAIP over CSIDH-512 is the candidate hardness assumption that can be used for our optimized blind signature construction.

In the next subsection, we show that the hardness of ζ_d-rGAIP varies with the choice of ζ_d. Since we believe ζ_d-rGAIP has independent interest outside our optimized blind signature application, we waive Requirement 3 when considering the cryptanalysis.

6.5.2 Cryptanalysis and Structural Attack on rGAIP

In the previous section, we show how to choose a root ζ_d according to the decomposition of the multiplication group of $\mathbb{Z}_N^\times$. In this section, we show that the underlying structure of ζ_d in each component is related to the security of ζ_d-rGAIP by presenting a concrete cryptanalysis on the overstretched ζ_d-rGAIP with respect to the CSIDH-512 parameter.

Generic Attacks on GAIP.

We start from recalling the most efficient classical and quantum algorithms against a GAIP. The best known classical algorithm against GAIP is the meet-in-the-middle/Pollard-rho attack [GHS02, GS13] with time complexity $O(\sqrt{|G|}) = O(\sqrt[4]{p})$ against GAIP.

The best-known quantum algorithm against GAIP is Kuperberg's algorithm [Kup05, Reg04, Kup11, Pei20, BS20]. Typically, given a challenge E to find $a \in \mathbb{Z}_N$ such that $E_0 = [\mathfrak{g}^a] \star E$, we have a hidden shift problem by defining $f(x) = [\mathfrak{g}^x] \star E_0$ and $g(x) = [\mathfrak{g}^x] \star E$, the permutations f, g over $\mathcal{E}$ are hidden shifted by a. By applying Kuperberg's algorithm, one can solve GAIP in time complexity $2^{O(\sqrt{\log(|G|)})}$. It is not clear whether the additional structure can give an advantage to the adversary by reducing the group size *in general*. The subset $\{1, \zeta_d, \ldots, \zeta_d^{d-1}\}$ forms a group with multiplication instead of addition. Modifying the group action by restricting to the multiplication subgroup of $\mathbb{Z}_N^\times$ does not give a feasible g with a hidden shift a. Also, ζ generates the entire $\mathbb{Z}_N$ under the addition due to being a unit so that the quotient group does not help in this case.

Structural Attack on rGAIP.

Let ζ_d be a d-th primitive root of unity and N be the class number. We show that the underlying structure of the root in each component of $\mathbb{Z}_N^\times$ is related to security by displaying a structural attack against ζ_d-rGAIP and the efficacy of the attack is related to each $\mathsf{gcd}(\zeta_d^i - 1, N)$.

The high-level strategy of our structural attack is to break down a ζ_d-rGAIP instance into several GAIP instances over smaller subgroups or quotient groups. The idea is to exploit the differential information of any two curves in the instance and launch a Pohlig-Hellman-type attack. Recall that the instance is of the form $(X_0 = [\mathfrak{g}^a] \star E_0, X_1 = [\mathfrak{g}^{a\zeta_d}] \star E_0, \ldots, X_{d-1} = [\mathfrak{g}^{a\zeta_d^{d-1}}] \star E_0)$. For any two curves X_i, X_j in the instance, there exists a unique group element $[\mathfrak{g}_{ij}] = [\mathfrak{g}^{a\zeta_d^j - a\zeta_d^i}] \in G$ such that $[\mathfrak{g}_{ij}] \star X_i = X_j$. Therefore, recovering differential action $[\mathfrak{g}_{ij}]$ gives the information of a. Typically, it is difficult to recover such $[\mathfrak{g}_{ij}]$ due to the size of the group and considering the GAIP of (X_i, X_j). However, depending on the knowledge of η_d derived from the public ζ_d, the hardness of the GAIP of the structural (X_i, X_j) can be reduced. This is because $G_{ij} := \{[\mathfrak{g}^{n(\zeta_d^j - \zeta_d^i)}] \,|, n \in \mathbb{Z}_N\}$ possibly constitutes a proper subgroup of G up to i and j. For any $[\mathfrak{g}'] \in G$, we have $[\mathfrak{g}'] \star X_i = [\mathfrak{g}_{ij}] \star X_i = X_j$ if and only if $[\mathfrak{g}']G_{ij} = [\mathfrak{g}_{ij}]G_{ij}$. As a result, recovering $[\mathfrak{g}']G_{ij}$ is exactly a GAIP problem of (X_i, X_j) over the quotient group G/G_{ij}. Then, after obtaining such $[\mathfrak{g}'] \in G$ such that $[\mathfrak{g}']G_{ij} = [\mathfrak{g}^{a\zeta_d^j - a\zeta_d^i}]G_{ij} = [\mathfrak{g}^a]G_{ij}$, we can recover $[\mathfrak{g}^a]$ by solving $(E_0, (\mathfrak{g}')^{-1} \star X_0)$ over G_{ij} for ${\mathfrak{g}'}^{-1}[\mathfrak{g}^a]$. Therefore, the main strength against our structural attack depends on the GAIP hardness with the group size of $\max(|G_{ij}|, |G/G_{ij}|)$.

By extending upon the above idea and choosing a proper subsequence of (i, j), the root ζ_d gives the following ascending chain:

$$\{1\} = G_1 < G_2 < \ldots < G_k = G,$$

where for each $\ell \in [k]$, $G_\ell = G_{ij}$ for some distinct $i, j \in [d]$. Using the aforementioned structural attack, the hardness of ζ_d-rGAIP is determined by the size of the largest quotient group $G_{\ell+1}/G_\ell$ for some $\ell \in [k-1]$.

We may summarize the attack (reduction) described above as the following theorem.

Theorem 6.5.2. *Let $(G, \mathcal{E}, E_0, \star)$ be KO-EGA. Let ζ be a d-th root of unity for G. Let*

$$\{1\} = G_1 < G_2 < \ldots < G_k = G$$

*be a chain for G where $G_\ell = \{[\mathfrak{g}^{n(\zeta^j - \zeta^i)}] \,|, n \in \mathbb{Z}_N\}$ for some $i, j \in \{0, \cdots, d-1\}$. Given a GAIP adversary $\mathcal{A}$ over the KO-EGA model, there exists an ζ-*rGAIP *adversary $\mathcal{B}$ running in polynomial time with $O(k)$ queries to $\mathcal{A}$ such that*

$$\mathsf{Adv}^{\zeta\text{-rGAIP}}_{(G,\mathcal{E},E_0,\star)}(\mathcal{B}) \leq \sum_{\ell=1}^{k-1} \mathsf{Adv}^{\mathsf{GAIP}}_{(G_{\ell+1}/G_\ell,\mathcal{E},X_i',\star)}(\mathcal{A}).[9]$$

where X_i' is some element in $\mathcal{E}$ choosen in the reduction.

Proof. Given ζ-rGAIP challenge $(X_0 = [\mathfrak{g}^a] \star E_0, X_1 = [\mathfrak{g}^{a\zeta}] \star E_0, \ldots, X_{d-1} = [\mathfrak{g}^{a\zeta^{d-1}}] \star E_0)$, the adversary $\mathcal{B}$ with access to $\mathcal{A}$ proceeds as follows:

1. Set $\mathfrak{g}_k = \mathfrak{g}^{a_k} = 1$ where $a_k = 0$ and write $G_\ell = G_{ij} = \{[\mathfrak{g}^{n(\zeta^j - \zeta^i)}] \,|, n \in \mathbb{Z}_N\}$ for some $i, j \in \{0, \cdots, d-1\}$. For ℓ starting from $k-1$ to 1, solve the GAIP instance $(\mathfrak{g}^{-a_{\ell+1}\zeta^i} \star X_i, \mathfrak{g}^{-a_{\ell+1}\zeta^j} \star X_j)$ over the group $G_{\ell+1}/G_\ell$ using $\mathcal{A}$, then obtain $\mathfrak{g}_\ell = \mathfrak{g}^{a_\ell}$ for some a_ℓ.

2. Output $\mathfrak{g}_1 \cdots \mathfrak{g}_k$.

In the first iteration, we obtain $\mathfrak{g}^{a_{k-1}} \in G/G_{k-1}$ such that $\mathfrak{g}^{a_{k-1}} \star G_{k-1} = \mathfrak{g}^{a_{k-1}} \star G_{k-1}$. We therefore have $(\mathfrak{g}^{a_{k-1}})^{-1}\mathfrak{g}^a \in G_{k-1}$. This reduces to finding a ζ-rGAIP solution over G_{k-1} where the instance can be produced by $(\mathfrak{g}^{-a_{k-1}} \star X_0, \mathfrak{g}^{-a_{k-1}} \star X_1, \cdots, \mathfrak{g}^{-a_{k-1}\zeta^{d-1}} \star X_{d-1})$.

In the each iteration afterward, we obtain $\mathfrak{g}^{a_\ell} \in G_{\ell+1}/G_\ell$ such that $\mathfrak{g}^{a_\ell} \star G_\ell = \mathfrak{g}^{a_\ell} \star G_\ell$. We therefore have $(\mathfrak{g}^{a_\ell})^{-1}\mathfrak{g}^a \in G_\ell$. This reduces to finding a ζ-rGAIP solution over G_ℓ where the instance can be produced by $(\mathfrak{g}^{-a_\ell} \star X_0, \mathfrak{g}^{-a_\ell\zeta} \star X_1, \cdots, \mathfrak{g}^{-a_\ell\zeta^{d-1}} \star X_{d-1})$. By using the mathematical induction, we have $\mathfrak{g}_1^{-1} \cdots \mathfrak{g}_k^{-1}\mathfrak{g}^a \in \{1\}$ in the end and therefore $\mathfrak{g}_1 \cdots \mathfrak{g}_k = \mathfrak{g}^a$. $\square$

Remark 6.5.3. *We note that* $\mathsf{gcd}(\zeta^i - 1, N)$ *is divisible by a prime divisor p of N if and only if $\zeta^{\frac{d}{\mathsf{gcd}(i,d)}} \equiv 1 \pmod{p}$. Thus we only need to calculate* $\mathsf{gcd}(\zeta^{d'} - 1, N)$ *for every divisor d' of d to find η_d. In particular, when d is prime, we need only compute* $\mathsf{gcd}(\zeta - 1, N)$ *to find η_d. Therefore, we only need to consider* $\mathsf{gcd}(\zeta - 1, N)$ *for* $d = 3, 5, 7, 11, 47, 499$ *for the CSIDH-512 parameter set.*

We use the methods of Remark 6.5.3 and Thm. 6.5.2 to analyze the strength of ζ_d-rGAIP with respect to d for a few parameters over CSIDH-512 as follows.

[9]Over KOEGA $(G, \mathcal{E}, E_0, \star)$, the group G can be represented by $\oplus_{i=1}^d \mathbb{Z}_{m_i}$. Therefore, it induces the KO-EGA $(G_{\ell+1}/G_\ell, \mathcal{E}, E_0, \star)$ for any ℓ in a cananical way by using the operation from $(G, \mathcal{E}, E_0, \star)$.

$$
\begin{aligned}
\mathsf{gcd}(\zeta_2 - 1, N) &= \mathsf{gcd}(\zeta_4 - 1, N) = 1 \\
\mathsf{gcd}(\zeta_3 - 1, N) &= 3 \times 1407181 \times 51593604295295867744293584889 \\
&\quad \times 31599414504681995853008278745587832204909 \\
\mathsf{gcd}(\zeta_4^2 - 1, N) &= 3 \\
\mathsf{gcd}(\zeta_5 - 1, N) &= 3 \times 37 \times 51593604295295867744293584889 \\
&\quad \times 31599414504681995853008278745587832204909 \\
\mathsf{gcd}(\zeta_7 - 1, N) &= 3 \times 37 \times 1407181 \times 31599414504681995853008278745587832204909 \\
\mathsf{gcd}(\zeta_8 - 1, N) &= 31599414504681995853008278745587832204909 \\
\mathsf{gcd}(\zeta_8^2 - 1, N) &= \mathsf{gcd}(\zeta_8 - 1, N) \times 3 \\
\mathsf{gcd}(\zeta_8^4 - 1, N) &= \mathsf{gcd}(\zeta_8^2 - 1, N) \times 37 \times 1407181 \\
\mathsf{gcd}(\zeta_9 - 1, N) &= \mathsf{gcd}(\zeta_9^2 - 1, N) = \mathsf{gcd}(\zeta_3 - 1, N) \\
\mathsf{gcd}(\zeta_{47} - 1, N) &= \mathsf{gcd}(\zeta_{499} - 1, N) = \mathsf{gcd}(\zeta_5 - 1, N).
\end{aligned}
$$

As a consequence, we reduce each ζ_d-rGAIP instance to a GAIP instance with a group size determined by ζ_d. This is summarized in Tab. 6.1. For ζ_8, we have a chain $\{1\} = G_1 < G_2 < G_3 < G_4 < G_5 = G$ where G_2, G_3, G_4 is of size $\mathsf{gcd}(\zeta_8 - 1, N), \mathsf{gcd}(\zeta_8^2 - 1, N), \mathsf{gcd}(\zeta_8^4 - 1, N)$, respectively, and the largest quotient group is $|G_2/G_1| \approx 2^{134}$, which demonstrates the invulnerability of ζ_8-rGAIP. For instance, for ζ_3 we have a chain $\{1\} = G_1 < G_2 < G_3 = G$ where G_2 is of size 37 and the largest quotient group is $|G_3/G_2| \approx 2^{251}$. For ζ_4, ζ_{47} and ζ_{499} we have a chain $\{1\} = G_1 < G_2 < G_3 = G$ where G_2 is of size 1407181 with the largest quotient group $|G_3/G_2| \approx 2^{236}$. Our cryptanalysis gives an upper bound of ζ_d-rGAIP from the perspective of GAIP. Importantly, ζ_4-rGAIP which we use for our optimized blind signature only seems to lose 2 bits of security compared with ζ_2-rGAIP, or equivalently, GAIP over CSIDH-512.

ζ_d-rGAIP	ζ_2	ζ_3	ζ_4	ζ_5	ζ_7	ζ_8	ζ_9	ζ_{47}	ζ_{499}
GAIP with Group Size in $\log_2$	257	251	255	236	161	134	251	236	236

Table 6.1: The upper row denotes ζ_d-rGAIP over CSIDH-512. Using our cryptanalysis in Sec. 6.5.2 and Thm. 6.5.2, we reduce each ζ_d-rGAIP instance into a GAIP instance with a group size summarized in the lower row. Note that GAIP over CSIDH-512 is equivalent to ζ_2-rGAIP over CSIDH-512.

6.5.3 Equivalence between GAIP and rGAIP

We complement our cryptanalysis by showing that our attack is optimal for some parameters. Although a few instances of ζ_d-rGAIP were shown to be significantly weaker than the original GAIP over CSIDH-512, we present a surprising condition that allows to reduce ζ_d-rGAIP to the original GAIP. This shows that the attack in Tab. 6.1 is optimal for those specific choices of ζ_d. We note that although the condition does not cover all cases (including ζ_4 which meets Requirement 3), the result gives us some guidance of the hardness of ζ_d-rGAIP.

Large $\mathsf{gcd}(\zeta_d - 1, N) \approx N$.

We begin by noting that in this case we do not know how to have an efficient extractor in our optimized sigma protocol due to the large value of η_d (see Lem. 6.4.1). In other words, it does not satisfy Requirement 3.

It is clear that GAIP is never easier than ζ_d-rGAIP. The key insight of the reverse reduction is that when $\mathsf{gcd}(\zeta_d - 1, N) \approx N$ (or $\mathsf{gcd}(\zeta_d - 1, N) = N/\mathsf{poly}$ to be precise), given a GAIP instance we can generate a ζ_d-rGAIP instance by trial and error. Additionally, the success rate can also be amplified by repetitively invoking the GAIP oracle and testing the correctness.

Concretely, given $X_0 = [\mathfrak{g}^a] \star E_0$ and access to an ζ_d-rGAIP adversary $\mathcal{A}$ for a d-th primitive root of unity ζ_d, we can construct a GAIP adversary $\mathcal{B}$ which invokes $\mathcal{A}$ on input $(X_0, [a'] \star X_0, [a'^{\zeta_d}] \star X_0, \ldots, [a'^{\zeta_d^{d-1}}] \star X_0)$ where a' is sampled uniformly at random from the subgroup $\{r^{\zeta_d - 1} | r \in G\}$. Then, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. Since the subgroup is of size $N/\mathsf{gcd}(\zeta_d - 1, N) = \mathsf{poly}$, the adversary $\mathcal{B}$ invokes $\mathcal{A}$ on a well-formed instance with probability $\mathsf{gcd}(\zeta_d - 1, N)/N$, which is non-negligible. This is because the instance is well-formed if and only if $[a'] = [\mathfrak{g}^{a(\zeta_d - 1)}]$, where $\langle \mathfrak{g}^{\zeta_d - 1} \rangle$ is of size $N/\mathsf{gcd}(\zeta_d - 1, N)$.

Therefore, the reduction described above leads to the following theorem.

Theorem 6.5.4. *Given any ζ_d-rGAIP adversary $\mathcal{A}$ for a known-order effective group action of the group size N, there exists a GAIP adversary $\mathcal{B}$ in time d over the same action such that* $\mathsf{Adv}^{\zeta_d\text{-rGAIP}}(\mathcal{A}) \leq \frac{N}{\mathsf{gcd}(\zeta_d - 1, N)} \cdot \mathsf{Adv}^{\mathsf{GAIP}}(\mathcal{B})$.

As a consequence, we know that for CSIDH-512 we have ζ_3, ζ_9, ζ_5, ζ_{47}, ζ_{499}-rGAIPs are as hard as the original GAIP with a reduction loss of factors $37, 37, 1407181, 1407181, 1407181$ respectively. Similarly, $\zeta_{117265} = \zeta_5 \zeta_{47} \zeta_{499}$ also has a reduction loss of a factor 1407181.

6.6 Performance

We present an overall performance in Tab. 6.2 for our Otters protocol (Fig. 6.2) instantiated using CSIDH-512 of CSI-FiSh. The group action from CSIDH-512 has been estimated to have 128 bits of classical security and over 60 bits of quantum security [Pei20]. We instantiate the blind signature schemes with ζ_2 and ζ_4 chosen in Sec. 6.5, denoted by CSI-Otters and CSI-Otters-ζ_4. When instantiated with $\zeta_2 = -1$, the blind signature scheme is based on the standard group action inverse problem. As explained in Sec. 6.5, ζ_4 is the only parameter that satisfies Requirement 3 while being presumably as hard as GAIP over CSIDH-512.

In CSIDH-512, it takes 512 bits to represent a set element for $\mathcal{E}$ and 256 bits to represent a group element for G. In the context of CSI-Otters, the public key of the signer is one element of $\mathcal{E}$, and the bandwidth is also 2λ elements of $\mathcal{E}$. On the other hand, the user's bandwidth contains a λ-bit string, and the resulting signature is composed of 2λ group elements along with 2λ bits. In the context of $\mathsf{CSI\text{-}Otters}\text{-}\zeta_4$, the number of repetitions is halved. The public key of the signer is 4 elements of $\mathcal{E}$, and the bandwidth is also 4λ elements of $\mathcal{E}$. On the other hand, the user's bandwidth contains a λ-bit string, and the resulting signature is composed of λ group elements along with 2λ bits.

	BW.S	BW.U	$\lvert\mathsf{sk}\rvert$	$\lvert\mathsf{pk}\rvert$	$\lvert\sigma\rvert$	Assumption
CSI-Otters	16 KB	16 B	16 B	128 B	8.2 KB	GAIP
CSI-Otters-ζ_4	64 KB	16 B	16 B	512 B	4.1 KB	ζ_4-rGAIP

Table 6.2: The overall performance of our blind signature family regarding the bandwidth (BW), the secret key size, the public size, and the signature size using CSIDH-512. We take $\lambda = 128$ and sk is generated by a seed of λ bits.

6.7 Discussion

We would like to suggest two interesting questions for this topic. Firstly, an intriguing open question pertains to whether the ROS (Random inhomogeneities in a Overdetermined Solvable system) attack, as introduced by [BLL+21], can be applied to our blind signature scheme. In essence, the ROS problem, originally formulated by Schnorr [Sch01], involves adaptively making signature queries and computing linear combinations of provided signatures to produce a new valid signature. However, in general isogeny-based cryptosystems do not appear to offer the operations (merging curves). Consequently, it raises an intriguing inquiry into whether there exists a variant of the ROS attack that can be adapted to our scheme.

Secondly, the work conducted by Tessaro and Zhu [TZ22] presents a few blind signature schemes which are provably concurrently secure. The security can be proven independently of the ROS problem under the abstract models (GGM and AGM). This prompts the question of whether it is possible to demonstrate or devise a new blind signature scheme under abstract models [DHK+23, BGZ23] that can be proven secure against adaptive attacks.

Chapter 7

Verifiable Random Functions

This chapter presents the work carried out in [Lai23], which the author of the thesis individually worked on. The chapter is almost verbatim of the original work.

Abstract. In this chapter, we introduce the first provably secure verifiable random functions (VRFs) based on isogenies and the standard DDH assumption, which offer a competitive proof size among the post-quantum VRFs. To prove their security, we introduce a generalized variant called the master DDH problem and demonstrate its equivalence to the standard DDH problem. Furthermore, we demonstrate a novel application of the quadratic twist to expand the input space and reduce the size of the verification key and the proof.

7.1 Introduction

Verifiable random functions (VRFs) are a cryptographic primitive that were first introduced by Micali, Rabin, and Vadhan [MRV99]. They are a more advanced form of pseudorandom functions (PRFs) that not only generate pseudorandom outputs, but also provide a non-interactive and publicly verifiable proof to validate the output. The security of VRFs is maintained even when numerous copies of the input, output, and proof are made public. In particular, the notion of residual pseudorandomness for VRFs ensures that the pseudorandomness remains for inputs that have not been evaluated and the unique provability guarantees that it is computationally infeasible for an attacker to generate distinct outputs for the same input with valid proofs.

The versatility of VRFs has been demonstrated through their applications in DNSSEC protocols [GNP+15] and, especially, blockchain technology [GHM+17, HMW18, EKS+21]. The growth of cryptocurrencies such as Bitcoin and Algorand has spurred significant interest in blockchain technology, which is being fueled by its potential. Early blockchain systems, such as Bitcoin, utilized the Proof-of-Work (PoW) consensus mechanism, where miners compete to solve a cryptographic puzzle and the winner is rewarded. In contrast, the Proof-of-Stake (PoS) consensus protocol provides a more environmentally sustainable solution by allowing validators to stake their tokens and conducting an online lottery. Due to the cryptographic properties, VRFs play a critical role in PoS blockchain applications for their applications in cryptographic sortition and Byzantine consensus [GHM+17, DGKR18, HMW18].

In practice, most existing VRFs are based on elliptic curve cryptography (ECC), pairing-based BLS-type signatures or other Diffie-Hellman-type assumptions [BGLS03, BMR10, ACF14, Jag15, PWH+17]. However, these VRFs are vulnerable to quantum computing attacks, as they rely on underlying assumptions that can be broken by a quantum adversary in polynomial time [Sho99]. Despite their versatility and significance, post-quantum VRFs are underdeveloped, with only five constructions from three works to date [EKS+21, BDE+22, ESLR22]. The preliminary result of the lattice-based LB-VRF [EKS+21] provides limited residual pseudorandomness and requires updating the public key after, at most, five evaluations. Though it is sufficient in some scenarios, it cannot serve for long-term applications or on a large scale. Currently, only SL-VRF [BDE+22] from LowMC and the lattice-based LaV [ESLR22] offer full VRF capabilities. Regardless of the existence of Naor–Reingold-type PRFs (pseudorandom synthesizers [NR99]) [BPR12, Mon18] in lattices, the most versatile post-quantum branch, it seems challenging to push them forward to VRFs from this direction in a practical manner. Therefore, post-quantum VRFs have limited development, with only two full VRF proposals relying on a well-known assumption from symmetric primitives and lattices. Further research is necessary to address this challenge and further advance the capabilities of VRFs.

Despite a known subexponential vulnerability [Reg04, Kup05, Kup11, Pei20, BS20], recent research continues to demonstrate the competitiveness of isogeny cryptography as a post-quantum branch, including signature schemes [BKV19, EKP20, DG19], UC-secure oblivious transfers [LGD21, BMM+22], theshold signatures [DM20], (linkable/accountable) ring and group signatures [BKP20, BDK+22], and PAKE [AEK+22]. Due to the less rich algebraic structure offered by the isogenies, translating classical constructions has shown to be a non-trivial task in general [BKP20, MOT20, LGD21, BDK+22] from the perspective of viable and practical tools and the reliable and versatile assumptions, both of which very limited. For instance, the most practical classical counterpart ECVRF [PWH+17], based on a signature scheme with the unique signature property, requires hashing a string to a supersingular elliptic curve with unknown endomorphism ring, which is known to be a notorious bottleneck in isogeny-based cryptography [BBD+22, MMP22]. Additionally, the use of pairings, [BGLS03, BLS01], could lead to a "partially post-quantum" only result [DMPS19].

Regardless of the prior failure and the difficulties, it is still natural to ask:

Can we have a post-quantum verifiable random functions from isogenies with a competitive performance and without compromising security notions?

7.1.1 Related Works

To the best of our knowledge, there are only five constructions from three works to date [EKS+21, BDE+22, ESLR22] related to post-quantum VRFs. Of these, the lattice-based LB-VRF, X-VRF in [EKS+21, BDE+22] have limited capabilities but with compact proof sizes

(0.6-7.3KB). They provide only limited residual pseudorandomness, requiring the public key to be updated after a limited number of evaluations, making it unsuitable for long-term applications or large-scale use. The SL-VRF and LaV in [BDE+22, ESLR22] provide full VRFs from LowMC and the hybrid MSIS/MLWR respectively. They have proof sizes of 40KB,12KB, and the secret key sizes of 24B and 6.4KB respectively.

In the field of isogeny cryptography, various protocols have been proposed that relate to random functions. For instance, Naor-Reingold type pseudorandom functions (PRF) have been proposed in [ADMP20, MOT20]. Additionally, there have been proposals for oblivious random functions using oblivious transfers with a Naor–Reingold-type PRF or one-more type assumptions [BKW20]. The latter one has been shown insecure [BKM+21]. Currently, a provably secure isogeny-based VRF has yet to be introduced in the literature.

7.1.2 Contributions

In this study, we present two VRFs, CAPYBARA and TSUBAKI[1], which provide an affirmative solution to the above question through the following three contributions.

1. Inspired and based on the Naor–Reingold pseudorandom function as in [ADMP20, BKW20, MOT20], we construct a proof system where the prover can demonstrate the knowledge of the action factorization of a set element based on a distinguished base point (see R_{fac} defined below). We use the technique from [BDK+22] to make the proof system online-extractable, providing tightly-secure unique provability. Additionally, we utilize the approach in [BKP20] to reduce the proof size. As a result, our VRFs have an exponentially large input space ($\{0,1\}^\lambda$) and expected proof sizes of 39KB and 34KB using CSIDH-512, which is comparable to the symmetric-primitive-based VRF [BDE+22]. The secret key can also be strored (compressed) as a 32B seed and generated efficiently usign PRNG on input of the seed.

2. We introduce a new decisional assumption, known as the master decisional Diffie-Hellman problem, which implies a variety of decisional problems. We show that it is as hard as the original DDH problem.

3. We show a new use of the quadratic twists (see Footnote 2) to expand the input space to be ternary ($\{-1,0,1\}^\kappa$). By using a similar method, we prove that this variant is as secure as the decisional square Diffie-Hellman problem, whose computational version is as hard as the group action inverse problem.

As a result, we introduce the first group action and isogeny-based VRFs in literature with a competitive performance. Additionally, our CAPYBARA construction is based on the standard DDH assumption. Our method of construction and the techniques utilized are versatile and can be applied to other number-theoretic pseudorandom functions, demonstrating the promising potential of incorporating group actions and isogeny cryptography in the field of VRF research.

[1]Compact Action factorization Proofs Yielded By A RAndom function and Twist-SqaUre-BAsed tweaK from Isogenies.

7.1.3 Technical Overview

The ideas beneath this work are fairly simple. First, given a transitive and free (effective) group action $(G, \mathcal{E}, \star, h_0)$ for some distinguished element $h_0 \in \mathcal{E}$, we start from a Naor–Reingold-type pseudorandom function on input $x = (x_1 \cdots x_\kappa) \in \{0,1\}^\kappa$:

$$f(\mathsf{sk}, x) = (c_0 c_1 g_1^{x_1} \cdots g_\kappa^{x_\kappa}) \star h_0$$

where the secret key $\mathsf{sk} = (c_0, c_1, g_1, \cdots, g_\kappa)$ with the public key $\mathsf{vk} = (c_0 \star h_0, c_1 \star h_0, g_1 \star h_0, \cdots, g_\kappa \star h_0)$. Remark that without c_1, it is a secure pseudorandom random function but not a secure verifiable random function since the adversary is given vk so that the evaluation at 0 is known.

Second, the factorization over the group $g = \Pi g_i$ (not necessarily unique) gives the factorization of $g \star h_0$ over the set with respect to h_0. We construct an action factorization proof system to prove the correctness of the evaluation of $f(\mathsf{sk}, x)$. Formally, let $h \leftarrow f(\mathsf{sk}, x)$ on input $x \in \{0,1\}^\kappa$. We consider the action factorization relation

$$R_{\mathsf{fac}} = \left\{ ((h_0, X_0, X_1, \{h_i\}_{i\in I}, h), (c_0, c_1, \{g_i\}_{i\in I})) \;\middle|\; \begin{array}{r} X_j = c_j \star h_0 \ \forall j \in \{0,1\} \\ g_i \star h_0 = h_i \ \forall i \in I \\ (c_0 c_1 \Pi_{i\in I} g_i) \star h_0 = h \end{array} \right\},$$

where $I = \{i \in [\kappa] | x_i = 1\}$. Notice that without h in the statement and the constraint, the proof system is trivial using a standard graph-isomorphism-type proof of knowledge in parallel. We show that a user with the corresponding witness can prove a set element $h \in \mathcal{E}$ can be "factorized" through $\{h_i\}_{i\in I}$ and h_0 when the action is over an abelian group.

The three-move public-coin proof system starts from the prover who generates random $r, r_0, r_i \leftarrow G$ for $i \in I$, computes $(r \star X_0, r_0 \star X_1, \{r_i \star h_i\}_{i\in I}, (r r_0 \Pi_{i\in I} r_i) \star h) = (X_0', X_1', \{h_i'\}_{i\in I}, h')$, and sends it to the verifier. The verifier returns a random challenge $b \in \{0,1\}$ to the prover. Depending on b, the prover reveals $(r c_0^b, r_0 c_1^b, \{g_i^b r_i\}_{i\in I})$ to the verifier. Upon receiving $(r', r_0', \{r_i'\}_{i\in I})$, if $b = 0$, the verifier checks whether $(r' \star X_0, r_0' \star X_1, \{r_i' \star h_i\}_{i\in I}, (r' r_0' \Pi_{i\in I} r_i') \star h) = (X_0', X_1', \{h_i'\}_{i\in I}, h')$. If $b = 1$, the verifier checks whether $(r' \star h_0, r_0' \star h_0, \{r_i' \star h_0\}_{i\in I}, (r' r_0' \Pi_{i\in I} r_i') \star h_0) = (X_0', X_1', \{h_i'\}_{i\in I}, h')$. The verifier accepts if it is the case or rejects otherwise. By λ times repetitions and applying the Fiat-Shamir transform, one can obtain NIZK for the relation R_{fac}. For the sake of clarity, we present the construction by assuming the group structure is known. We show in Rem. 7.4.1 that the construction is also feasible in the unknown group structure setting.

Third, instead of resorting to an ad-hoc assumption, we prove the residual pseudorandomness of our VRF is as hard as the decisional Diffie-Hellman problem. We first introduce a generalized decisional problem – the master decisional Diffie-Hellman problem. The problem starts with the challenger giving the adversary an instance $(g_1 \star h_0, \cdots, g_N \star h_0)$. The adversary can make queries for an arbitrary combination of $(g_{s_1} \cdots g_{s_k}) \star h_0$ for any $\{s_1, \cdots, s_k\} \subseteq [N]$, and also sends a challenge query, which has not been queried before. The challenger returns as instructed or a random set element from $\mathcal{E}$, and the adversary's task is to determine which is the case. The problem covers a variety of variants of group-action-based decisional problems. Then, we prove the problem is as hard as the original DDH problem.

Fourth, we make the proof compact and achieve online extractability. The latter notion gives a tight reduction for the full uniqueness where the adversary cannot forge two valid proofs on the same input for two distinct evaluations for any malicious generated keys without using a rewinding argument. To achieve online extractability, one can consider using Unruh's transform [Unr15] (or Pass' transform [Pas03] by hashing both responses and appending them to the commitment). This, however, will result in costly overhead. Instead, while running the proof above, the prover uses a seed and a pseudorandom number generator (PRNG) to generate the group elements $r, r_0, \{r_i\}_{i\in I}$. By employing the proof technique developed in [BDK+22], the modification leads to an online-extractable proof system with much more compact proofs.

Fifth, as an independent interest in the CSIDH setting, we show a new use of the quadratic twists (see Footnote 2) and reduce the sizes of the public and secret keys and the computational cost for the user by relaxing the assumptions. In this way, the public key can be naturally expanded twice $(c_0 \star h_0, c_1 \star h_0, g_1 \star h_0, \cdots, g_\kappa \star h_0, (g_1 \star h_0)^t, \cdots, (g_\kappa \star h_0)^t)$.[2] The modification reduces 37% of the key size, 15% the evaluation cost, and 12% the maximal proof size. We prove that the underlying assumption for the residual pseudorandomness is as hard as the decisional square Diffie-Hellman problem in the appendix, of which the computational version is as hard as the group action inverse problem (i.e. Dlog).

Finally, we optimize the proof size again using the unbalanced challenge space and the seed trees introduced in [BKP20], which reduces the proof sizes of both constructions by a factor of 3. The proof sizes of our final VRFs are expected to be 39KB and 34KB when using CSIDH-512.

Roadmap. We begin in Sec. 7.2.1 with a brief introduction to VRFs and hardness assumptions (Secs. 2.3, 7.2.2 and 7.2.3). We then introduce our action factorization proof system in Sec. 7.4. We present our VRF constructions, CAPYBARA, in Sec. 7.5 and its variant, TSUBAKI, in Sec. 7.6. We show the underlying assumption of CAPYBARA (resp. TSUBAKI) is as hard as the DDH problem in Sec. 7.3 (resp. the decisional square DDH problem in Sec. B.1). Finally, we give the final optimization for both constructions and the performance comparison in Sec. 7.7.

[2] Note the reduction of the key size comes in different flavors in contrast to [BKV19, EKP20] where the twist reduces the public key size by decreasing the soundness error of the sigma protocol. Here, the twist decreases the key size by expanding a binary input to a ternary input instead of benefiting the proof system. The proof system is still BINARY challenge in this construction.

7.2 Preliminaries

Notations.

We summarize some notations unique to this chapter. First, we use multiplication notation for the effective group action $(G, \mathcal{E}, E_0, \star)$. We give a brief discussion for translating the constructions in this chapter to a restricted effective group model in Rem. 7.4.1. For $\mathbf{v} = (a_1, \cdots, a_N) \in G^N$ and $\mathbf{e} = (E_1, \cdots, E_N) \in \mathcal{E}^N$, we extend the action to an arbitrary dimension by writing $\mathbf{v}\star\mathbf{e} = (a_1\star E_1, \cdots, a_N\star E_N) \in \mathcal{E}^N$. We also abuse the notation $\mathbf{v}\star E = (a_1\star E, \cdots, a_N\star E) \in \mathcal{E}^N$ when the context is clear. Also, $\mathbf{e}_i$ represents the i-th elementary vector where the i-th entry is 1 and the others are zeros. For an array $\mathbf{v} = (v_1, \cdots, v_N)$, we may denote the i-th entry v_i as $\mathbf{v}_i$. For a subset $I \subseteq [N]$, we let $\mathbf{v}_I$ denote the sub-array $(v_i)_{i\in I}$.

Remark 7.2.1 (**Additional Requirement.**). *We have an additional requirement for our group actions. For the security parameter λ, we require the group size $|G|$ to be larger than 2^λ. The requirement naturally holds due to the known quantum subexponential attacks $2^{O(\sqrt{|G|})}$ [Reg04, Kup05, Kup11, Pei20, BS20]. This is necessary to ensure that we have adequate min-entropy for our proof system in Sec. 7.4.*

7.2.1 Verifiable Random Functions

In this subsection, we give a brief introduction to the verifiable random functions, and its notions [MRV99].

Definition 7.2.2. *(Verifiable Random Function) A* verifiable random function (VRF) *consists of four probabilistic polynomial-time algorithms $\Pi_{\mathsf{VRF}} = \{\mathsf{ParGen}, \mathsf{KeyGen}, \mathsf{VRFEval}, \mathsf{Ver}\}$ where:*

- $\mathsf{ParGen}(1^\lambda)$*: On input a security parameter 1^λ, this probabilistic algorithm outputs some global, public parameter* pp*.*
- $\mathsf{KeyGen}(\mathsf{pp})$*: On input public parameter* pp*, this probabilistic algorithm outputs two binary strings, a secret key* sk *and a public key* vk*.*
- $\mathsf{VRFEval}(\mathsf{sk}, x)$*: On input a secret key sk and an input $x \in \{0,1\}^{\ell(\lambda)}$, this algorithm outputs (v, π) for the VRF value $v \in \{0,1\}^{m(\lambda)}$ and the corresponding proof π proving the correctness of v.*
- $\mathsf{Ver}(\mathsf{vk}, v, x, \pi)$*: On input (vk, v, x, π), this probabilistic algorithm outputs either 1 or 0.*

The residual pseudorandomness guarantees the pseudorandomness of the function even if the user has revealed many evaluations together with the proofs. In some applications, it is sufficient to have a few-times relaxed notion where the pseudorandomness is ensured for only limited copies of evaluations are revealed [EKS+21]. In this work, we consider the original version of the notion.

Definition 7.2.3. *((Residual) Pseudorandomness) Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT adversary. The pseudorandomness experiment $\mathsf{ExpVRF}^{\mathsf{PR}}_{\mathcal{A},\Pi_{\mathsf{VRF}}}(\lambda)$ of a VRF scheme Π_{VRF} proceeds as follows.*

```
1. $Q \leftarrow \emptyset$
2. $\mathsf{pp} \leftarrow \mathsf{ParGen}(1^\lambda)$
3. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$
4. $(\tilde{x}, st) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{VRFEval}}(\cdot)}(\mathsf{vk})$
5. $(v_0, \pi_0) \leftarrow \mathsf{VRFEval}(\mathsf{sk}, \tilde{x})$
6. $v_1 \leftarrow \{0,1\}^{m(\lambda)}$
7. $b \leftarrow \{0,1\}$
8. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{VRFEval}}(\cdot)}(v_b, st)$
9. The output of the experiment is defined to
   be 1 if $b' = b$ and $\tilde{x} \notin Q$, and 0 otherwise.
```

```
$\mathcal{O}_{\mathsf{VRFEval}}(x)$ :
    1. $Q \leftarrow Q \cup \{x\}$
    2. Return $\mathsf{VRFEval}(\mathsf{sk}, x)$
```

We say $\mathcal{A}$ wins if $\mathsf{ExpVRF}^{\mathsf{PR}}_{\mathcal{A},\Pi_{\mathsf{VRF}}}(\lambda) = 1$. The advantage of $\mathcal{A}$ is defined to be

$$\mathsf{Adv}^{\mathsf{PR}}_{\Pi_{\mathsf{VRF}}}(\mathcal{A}) := \left|\Pr\left[\mathcal{A} \textit{ wins}\right] - 1/2\right|,$$

where the probability is taken over the randomness used by $\mathcal{A}$ and the randomness used in the experiment. A VRF protocol Π_{VRF} is said to be pseudorandom *if for any PPT adversary $\mathcal{A}$ there exists a negligible function* negl *such that*

$$\mathsf{Adv}_{\mathcal{A},\Pi_{\mathsf{VRF}}} \leq \mathsf{negl}(\lambda).$$

Definition 7.2.4. *(Complete Provability) Let $\Pi_{\mathsf{VRF}} = \{\mathsf{ParGen}, \mathsf{KeyGen}, \mathsf{VRFEval}, \mathsf{Ver}\}$ be a VRF scheme. Π_{VRF} is said to have* provability *if for any $\mathsf{pp} \leftarrow \mathsf{ParGen}(1^\lambda)$ and $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, the output $(v, \pi) \leftarrow \mathsf{VRFEval}(\mathsf{sk}, x)$ satisfies*

$$\mathsf{Ver}(\mathsf{vk}, v, x, \pi) = 1.$$

The following notion, unique provability, implies that for any adversary (possibly computationally unbounded with at most polynomial public coin queries) it is difficult to generate a malicious public key such that the adversary can produce two valid proofs for two distinct evaluations of the same input.

Definition 7.2.5. *(Unique Provability) Let* $\Pi_{\mathsf{VRF}} = \{\mathsf{ParGen}, \mathsf{KeyGen}, \mathsf{VRFEval}, \mathsf{Ver}\}$ *be a VRF scheme and* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *be an adversary. A* uniqueness provability *experiment proceeds as follows.*

1. $\mathsf{pp} \leftarrow \mathsf{ParGen}(1^\lambda)$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathcal{A}_1(\mathsf{pp})$
3. $(\mathsf{vk}, x, v_1, v_2, \pi_1, \pi_2) \leftarrow \mathcal{A}_2(\mathsf{vk})$

We say an adversary $\mathcal{A}$ *wins if* $v_1 \neq v_2$ *and* $\mathsf{Ver}(\mathsf{vk}, v_1, x, \pi_1) = \mathsf{Ver}(\mathsf{vk}, v_2, x, \pi_2) = 1$. *The advantage of* $\mathcal{A}$ *is defined to be* $\mathsf{Adv}^{\mathsf{UP}}_{\Pi_{\mathsf{VRF}}}(\mathcal{A}) := \Pr[\mathcal{A}$ *wins*$]$ *where the probability is taken over the randomness used by* $\mathcal{A}$ *and in the experiment.*

7.2.2 Hardness Assumptions of Group Actions for CAPYBARA

In this subsection, we introduce a few standard assumptions in group actions. We start from two computational assumptions, which we will not use in our construction, but it is helpful to understand the hierarchy of the decisional versions.

The following is the core hardness assumption for our first VRF in Sec. 7.5.

We introduce a generalized version of the decisional problem – the master decisional problem, analogue to the generalized DDH assumption [BLMW07] and similar to the Uber-family assumptions [Boy08]. In the master decisional problem, the starting instance consists of several random set elements, and the adversary can query any combination of them with respect to the group elements. We will show that the generalized version is as hard as the DDH problem using a hybrid argument.

Problem 10 (Master Decisional Diffie-Hellman (MDDH) Problem)**.** *Let* $(G, \mathcal{E}, \star, E_0)$ *be a group action,* $n \in \mathbb{N}$*, and* $b \in \{0, 1\}$*. The decisional master Diffie-Hellman problem experiment* $\mathsf{Exp}^{\mathsf{MDDH}}(n, b)$ *on input* (n, b) *proceeds as follows.*

1. *The challenger* $\mathcal{C}$ *generates a tuple* $(g_1 \star E_0, \cdots, g_n \star E_0)$ *where* $g_1, \cdots, g_n \leftarrow G$*, and sends the tuple to the adversary* $\mathcal{A}$*.*
2. $\mathcal{A}$ *is given access to a Diffie-Hellman (DH) oracle on input* $(x_1, \cdots, x_n) \in \{0, 1\}^n$ *returning* $\prod_i^n g_i^{x_i} \star E_0$*.*
3. $\mathcal{A}$ *sends a string* $v = (v_1, \cdots, v_n) \in \{0, 1\}^n$ *to* $\prod_i^n g_i^{v_i} \star E_0$ *to the challenge oracle* $\mathcal{C}$*.*
4. $\mathcal{C}$ *ignores if* v *has been queried before or is of Hamming weight less than* 2*. Otherwise,* $\mathcal{C}$*, depending on the input* b*, computes* $X_0 = \prod_i^n g_i^{v_i} \star E_0$ *or* $X_1 = r \star E_0$ *for some* $r \leftarrow G$*, and send* X_b *to* $\mathcal{A}$*. This process will only output for one time.*
5. $\mathcal{A}$ *outputs* $b' \in \{0, 1\}$ *to guess* b*.*

We denote the advantage of a decisional master Diffie-Hellman problem adversary $\mathcal{A}$ by

$$\mathsf{Adv}^{\mathsf{MDDH}}(\mathcal{A}) = \left|\Pr[\mathcal{A}(\mathsf{Exp}^{\mathsf{MDDH}}(n, b=0)) \to 1] - \Pr[\mathcal{A}(\mathsf{Exp}^{\mathsf{MDDH}}(n, b=1)) \to 1]\right|,$$

where b is the randomness in the experiment, and the probability is taken over the randomness used by $\mathcal{A}$ and the randomness used in the experiment. The group action $(G, \mathcal{E}, \star, E_0)$ is implicitly parameterized in the experiment. We say the MDDH problem is hard, if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl such that $\mathsf{Adv}^{\mathsf{MDDH}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$.

The assumption implies a variety of forms of decisional problems. For instance, given $(a \star x, b \star x, c \star x, ab \star x, bc \star x, cd \star x)$ to distinguish between $abc \star x$ or a random element in $\mathcal{E}$ is an instance of the problem. The interactivity of the assumption appears to be strange at a glance. It is, however, very reasonable. Otherwise, when n is linear in λ, giving all combinations implies revealing almost the entire set $\mathcal{E}$. Looking ahead, we will use this problem to show our verifiable random function has residual pseudorandomness. Unlike pseudorandomness, where the adversary has access to either the pseudorandom function or a random function, the MDDH experiment allows the adversary to learn the evaluations of any combination of the instances adaptively. We show in Sec. 7.3 the equivalence of the master DDH and the original DDH.

7.2.3 Relaxed Assumptions for TSUBAKI

This section introduces a few relaxed decisional assumptions that allow us to construct a more efficient verifiable random function variant. We use the quadratic twists in this section, and for a group action $(G, \mathcal{E}, \star, E_0)$ we let $E_0 \in \mathcal{E}$ denote the element that has the property that $E_0^t = E_0$. Also, for any $(g, E) \in G \times \mathcal{E}$, we have $(g \star E)^t = g^{-1} \star E^t$.

Firstly, we relax the DDH problem by introducing the standard square variant problem. The problem has been used to construct some cryptographic protocols [DM20, AEK+22]. A very recent work [DHK+23] justifies the hardness of the assumption in a generic model for group actions.

Definition 7.2.6 (Decisional Square CSIDH (sDDH) Problem). *Let $(G, \mathcal{E}, \star, E_0)$ be a group action. The decisional square CSIDH problem is that the adversary $\mathcal{A}$ is given $T_b = (g_1 \star E_0, h_b \star E_0)$ where $h_0 = g_1^2, h_1 = g_2$ and $(g_1, g_2, b) \leftarrow G^2 \times \{0,1\}$ and return $b' \in \{0,1\}$.*

We denote the advantage of an sDDH adversary $\mathcal{A}$ by

$$\mathsf{Adv}^{\mathsf{sDDH}}(\mathcal{A}) = |\Pr[\mathcal{A}(T_0) \to 1] - \Pr[\mathcal{A}(T_1) \to 1]|,$$

where b is the randomness in the experiment, and the probability is taken over the randomness used by $\mathcal{A}$ and the randomness used in the experiment. The group action $(G, \mathcal{E}, \star, E_0)$ is implicitly parameterized in the experiment. We say the sDDH problem is hard, if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl such that $\mathsf{Adv}^{\mathsf{sDDH}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$.

The computational version of the problem is quantum equivalent to the computational problem [LGD21], and quantum equivalent to the GAIP problem [GPSV21]. A full quantum equivalence is given in [MZ22].
One can reduce the sDDH problem to the DDH problem by mapping the instance $(g_1 \star E_0, h_b \star_0)$ to $(g_1 \star E_0, (gg_1) \star E_0, (gh_b) \star E_0)$ where $g \leftarrow G$. Though the reverse reduction is not known, sDDH is still believed to be a hard problem.

We introduce the decisional assumptions for our VRF variant where the input is ternary from $\{-1, 0, 1\}$, naturally corresponding to the following queries.

Definition 7.2.7 (Twisted Master Decisional CSIDH (tMDDH) Problem)**.** *Let $(G, \mathcal{E}, \star, E_0)$ be a group action, $n \in \mathbb{N}$, and $b \in \{0, 1\}$. The twisted master DDH problem experiment $\mathsf{Exp}^{\mathsf{tMDDH}}(n, b)$ on input (n, b) proceeds as follows.*

1. *The challenger $\mathcal{C}$ computes $E = g \star E_0$ where $g \leftarrow G$.*
2. *$\mathcal{C}$ generates a tuple $(g_1 \star E, \cdots, g_n \star E)$ where $g_1, \cdots, g_n \leftarrow G$, and sends the tuple to the adversary $\mathcal{A}$.*
3. *$\mathcal{A}$ is given access to a Diffie-Hellman (DH) oracle on input $(x_1, \cdots, x_n) \in \{0, \pm 1\}^n$ returning $\prod_i^n g_i^{x_i} \star E$.*
4. *$\mathcal{A}$ sends a string $v = (v_1, \cdots, v_n) \in \{0, \pm 1\}^n$ to the challenge oracle $\mathcal{C}$.*
5. *$\mathcal{C}$ ignores if v has been queried before or is of Hamming weight less than 2. Otherwise, $\mathcal{C}$, depending on b, computes $X_0 = \prod_i^n g_i^{v_i} \star E$ or $X_1 = r \star E$ for some $r \leftarrow G$, and sends X_b to $\mathcal{A}$. This process will only output for one time.*
6. *$\mathcal{A}$ outputs $b' \in \{0, 1\}$ to guess b.*

We denote the advantage of the decisional problem adversary $\mathcal{A}$ by

$$\mathsf{Adv}^{\mathsf{tMDDH}}(\mathcal{A}) = \left| \Pr[\mathcal{A}(\mathsf{Exp}^{\mathsf{tMDDH}}(n, b = 0)) \to 1] - \Pr[\mathcal{A}(\mathsf{Exp}^{\mathsf{tMDDH}}(n, b = 1)) \to 1] \right|,$$

where b is the randomness in the experiment, and the probability is taken over the randomness used by $\mathcal{A}$ and the randomness used in the experiment. The group action $(G, \mathcal{E}, \star, E_0)$ is implicitly parameterized in the experiment. We say the tMDDH problem is hard, if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl such that $\mathsf{Adv}^{\mathsf{tMDDH}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$.

We show in Sec. B.1 that the twisted decisional master CSIDH problem is not easier than the decisional square CSIDH problem. To see this, we are introducing a non-standard intermediate assumption, which will make the proof easier to follow. The assumption coincides with a decisional version of a problem proposed in [LGD21].

Definition 7.2.8 (Decisional Reciprocal CSIDH (rDDH) Problem)**.** *Let $(G, \mathcal{E}, \star, E_0)$ be a group action. The decisional reciprocal CSIDH problem is that the adversary $\mathcal{A}$ is given $T_b = (g_1 \star E_0, g_2 \star E_0, h_b \star E_0, h'_b \star E_0)$ where $h_0 = g_1 g_2, h_1 = g_3, h'_0 = g_1 g_2^{-1}, h'_1 = g_4$ and $(g_1, g_2, g_3, g_4, b) \leftarrow G^4 \times \{0, 1\}$, and return $b' \in \{0, 1\}$.*

We denote the advantage of an rDDH adversary $\mathcal{A}$ by

$$\mathsf{Adv}^{\mathsf{rDDH}}(\mathcal{A}) = |\Pr[\mathcal{A}(T_0) \to 1] - \Pr[\mathcal{A}(T_1) \to 1]|,$$

where b is the randomness in the experiment, and the probability is taken over the randomness used by $\mathcal{A}$ and the randomness used in the experiment. The group action $(G, \mathcal{E}, \star, E_0)$ is implicitly parameterized in the experiment. We say the rDDH problem is hard, if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl such that $\mathsf{Adv}^{\mathsf{rDDH}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$.

The computational version proposed in [LGD21] has been proven to be equivalent to the computation square CDH problem, which is equivalent to the GAIP problem. The following proposition shows that the decisional reciprocal problem is not easier than the decisional square problem. In the appendix Sec. B.1, we will use the multi-challenge version of the decisional reciprocal problem to show the hardness of the twisted decisional master problem.

Proposition 7.2.9. *Let $(G, \mathcal{E}, \star, E_0)$ be a group action. Given an adversary $\mathcal{A}$ against the* rDDH *problem, there exist an* sDDH *adversary $\mathcal{B}_1$ and a decisional CSIDH problem $\mathcal{B}_2$ such that*

$$\mathsf{Adv}^{\mathsf{rDDH}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{sDDH}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{DDH}}(\mathcal{B}_2).$$

Proof. We prove this by introducing a sereis of hybrid games $\mathsf{Game}_1, \mathsf{Game}_2, \mathsf{Game}_3$ by gradually changing the experiment, where Game_1 corresponds to the case of $b = 0$ in the experiment (Def. 7.2.8) and Game_3 corresponds to the case $b = 1$.

Game_2 : the same as Game_1 except that the pair $(g_1 \star E_0, g_2 \star E_0, g_1 g_2 \star E_0, g_1 g_2^{-1} \star E_0)$ given to $\mathcal{A}$ is modified as $(g_1 \star E_0, g_2 \star E_0, g_1 g_2 \star E_0, g_4 \star E_0)$ where $g_4 \leftarrow G$. Claim $\mathsf{Game}_1 \approx_c \mathsf{Game}_2$ thanks to the sDDH problem. Concretely, we build an sDDH adversary $\mathcal{B}_1$ using $\mathcal{A}$. Upon receiving a square CSIDH challenge $(s \star E_0, X)$, the reduction $\mathcal{B}_1$ proceeds as follows

1. Generate $a \leftarrow G$.
2. Forward $(a \star (s \star E_0), (s \star E_0)^t, a \star E_0, a \star X)$ to $\mathcal{A}$.
3. Output whatever $\mathcal{A}$ returns.

Note that $(s \star E_0)^t = s^{-1} \star E_0$ and $a = (as)s^{-1}$. Therefore, when the challenge is the second case in the sDDH experiment (i.e. a random curve), $\mathcal{B}_1$ generates Game_2. On the other hand, if the challenge is the first case in the experiment (i.e. $X = s^2 \star E_0$), then $\mathcal{B}_1$ generates Game_1 since $a \star X = as^2 \star E_0$ and $as^2 = as(s^{-1})^{-1}$. Therefore, $\mathsf{Adv}^{\mathsf{sDDH}}(\mathcal{B}_1) = |\Pr[\mathcal{A}(\mathsf{Game}_1) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_2) \to 1]|$.

Game_3 : the same as Game_2 except that the pair $(g_1 \star E_0, g_2 \star E_0, g_1 g_2 \star E_0, g_4 \star E_0)$ given to $\mathcal{A}$ is modified as $(g_1 \star E_0, g_2 \star E_0, g_3 \star E_0, g_4 \star E_0)$ where $g_3 \leftarrow G$. This is exactly the second case in the rDDH problem. Claim $\mathsf{Game}_2 \approx_c \mathsf{Game}_3$ thanks to the DDH problem. Concretely, we build an DDH adversary $\mathcal{B}_2$ using $\mathcal{A}$. Upon receiving a square CSIDH challenge $(g_1 \star E_0, g_2 \star E_0, X)$, the reduction $\mathcal{B}_2$ proceeds as follows

1. Generate $g_4 \leftarrow G$.
2. Forward $(g_1 \star E_0, g_2 \star E_0, X, g_4 \star E_0)$ to $\mathcal{A}$.
3. Output whatever $\mathcal{A}$ returns.

Note that when the challenge is the second case in the DDH experiment (i.e. a random curve), $\mathcal{B}_2$ generates Game_3. On the other hand, if the challenge is the first case in the experiment (i.e. $X = g_1g_2 \star E_0$), then $\mathcal{B}_1$ generates Game_2. Hence, $\mathsf{Adv}^{\mathsf{DDH}}(\mathcal{B}_2) = |\Pr[\mathcal{A}(\mathsf{Game}_2) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_3) \to 1]|$.

Therefore, we have

$$\mathsf{Adv}^{\mathsf{rDDH}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{sDDH}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{DDH}}(\mathcal{B}_2).$$

□

7.3 Hardness of Master Decisional Diffie-Hellman Problem

The following theorem shows that the MDDH problem is as hard as the DDH problem. It is worth highlighting the reduction is inspired by the pseudorandomness treatment in the literature [BMR10, ADMP20, BKW20, MOT20].

Theorem 7.3.1. *The* MDDH *problem is not easier than the* mcDDH *problem. Concretely, let* $(G, \mathcal{E}, \star, E_0)$ *be a group action,* $\mathcal{A}$ *be a* MDDH *problem adversary with parameter* $n \in \mathbb{N}$*. If at most* $q_{\mathsf{DH}} = poly(\lambda)$ *queries are made in the experiment by* MDDH $\mathcal{A}$ *then there exists* mcDDH *problem adversaries* $\mathcal{B}_2, \cdots \mathcal{B}_n$ *such that*

$$\mathsf{Adv}^{\mathsf{MDDH}}(\mathcal{A}) \leq \sum_{i=2}^{n} \mathsf{Adv}^{\mathsf{mcDDH}}(\mathcal{B}_i).$$

Proof. We prove the theorem via a hybrid argument by introducing a series of games $\mathsf{Game}_1, \cdots,$ Game_n by modifying the responses of the DH oracle and the challenge oracle in the MDDH experiment gradually. Among the games, Game_1 is the original MDDH experiment, We will modify the response of the challenge oracle and the DH oracle together, which will be explained later. For $i \in [n]$ where $b \in \{0, 1\}$, let $\mathcal{A}(\mathsf{Game}_i(b))$ represent $\mathcal{A}$ running the Game_i, the modified MDDH experiment with the random coin b used in the experiment, and $\mathcal{A}$ will return 0 or 1. Therefore, by definition,

$$\mathsf{Adv}^{\mathsf{MDDH}}(\mathcal{A}) = |\Pr[\mathcal{A}(\mathsf{Game}_1(b = 1)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_1(b = 0)) \to 1]|. \tag{7.1}$$

Looking ahead, Game_n is the modified MDDH experiment where both the DH oracle and the challenger reply with random elements in $\mathcal{E}$. Therefore, since b is information theoretically hidden from $\mathcal{A}$,

$$|\Pr[\mathcal{A}(\mathsf{Game}_n(b=1)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_n(b=0)) \to 1]| = 0. \tag{7.2}$$

Game_1 : the original MDDH experiment starting with a tuple $(g_1 \star E_0, \cdots, g_n \star E_0)$ where $g_1, \cdots, g_n \leftarrow G$ and the oracle responds as specified.

Game_2 to Game_n: for $j \in \{2, \cdots, n\}$, Game_j is the same as Game_{j-1} except that the response of the DH oracle and the challenge oracle is modified as follows. The modification starts with a list L which is initially

$$\{(\mathbf{0}, E_0), (\mathbf{e}_1, g_1 \star E_0), \cdots, (\mathbf{e}_j, g_j \star E_0)\} \subseteq \{0,1\}^j \times \mathcal{E}.$$

On the query $x = (x_1, \cdots, x_n) \in \{0,1\}^n$, if $((x_1, \cdots, x_j), X) \in L$ for some $X \in \mathcal{E}$, the oracle returns $(\prod_{i=j+1}^{n} g_i^{x_i}) \star X$; otherwise, it draws $g' \leftarrow G$, computes $X = g' \star E_0$, adds $((x_1, \cdots, x_j), X)$ to the list L, and returns $(\prod_{i=j+1}^{n} g_i^{x_i}) \star X$ to $\mathcal{A}$. The reply for the challenge query is modified in the same way if the random coin $b = 0$.

Claim that $\mathsf{Game}_{j-1} \approx_c \mathsf{Game}_j$ for $\mathcal{A}$ for any $2 \le j \le n$. Concretely, a reduction $\mathcal{B}_j$ to the mcDDH problem proceeds as follows

1. Obtain $(g' \star E_0, \{(X_i, X_i')\}_{i \in [q_{\mathsf{DH}}+j-1]})$ from the mcDDH oracle.
2. Then, $\mathcal{B}_j$ initializes with a list

$$L = \left\{ \begin{matrix} (\mathbf{e}_1, X_1), \cdots, (\mathbf{e}_{j-1}, X_{j-1}), (\mathbf{0}, E), \\ (\mathbf{e}_1 + \mathbf{e}_j, X_1'), \cdots, (\mathbf{e}_{j-1} + \mathbf{e}_j, X_{j-1}'), (\mathbf{e}_j, g' \star E_0) \end{matrix} \right\} \subset \{0,1\}^j \times \mathcal{E},$$

   where $\mathbf{e}_i$ is the i-th elementary vector in $\{0,1\}^j$, and set a counter $\mathsf{ct} = j$ to record the number of the pairs (X_i, X_i') taken into the list L.
3. Invoke $\mathcal{A}$ on input $(E, X_1, \cdots, X_{j-1}, g' \star E_0, g_{j+1} \star E_0, \cdots, g_n \star E_0)$ where $g_{j+1}, \cdots, g_n \leftarrow G$.
4. Upon receiving the oracle query $(x_1, \cdots, x_n) \in \{0,1\}^n$, check whether $((x_1, \cdots, x_j), X) \in L$ for some $X \in \mathcal{E}$. If so, return $\prod_{i=j+1}^{n} g_i^{x_i} \star X$. Otherwise, update

$$L \leftarrow \{((x_1, \cdots, x_{j-1}, 0), X_{\mathsf{ct}}),\ ((x_1, \cdots, x_{j-1}, 1), X_{\mathsf{ct}}')\} \cup L,$$

   and set $\mathsf{ct} \leftarrow \mathsf{ct} + 1$, and rerun this step again.
5. Output whatever $\mathcal{A}$ returns.

Note that in Step 1. if $\mathcal{B}_j$ is in the experiment $\mathsf{Exp}^{\mathsf{mcDDH}}(0)$ in the mcDDH problem (Prob. 5 Item 1) then $\mathcal{B}_j$ generates Game_{j-1}. In contrast, if it is in the experiment $\mathsf{Exp}^{\mathsf{mcDDH}}(1)$ in the mcDDH problem (Prob. 5 Item 2), then $\mathcal{B}_j$ generates Game_j. It follows that for $b \in \{0,1\}$,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{mcDDH}}(\mathcal{B}_j) =& \mid \Pr[\mathcal{B}_j(\mathsf{Exp}^{\mathsf{mcDDH}}(0)) \to 1] - \Pr[\mathcal{B}_j(\mathsf{Exp}^{\mathsf{mcDDH}}(1)) \to 1]\mid \\ =& \mid \Pr[\mathcal{A}(\mathsf{Game}_{j-1}(b)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_j(b)) \to 1]\mid. \end{aligned} \tag{7.3}$$

Therefore, we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{MDDH}}(\mathcal{A}) &= \mid \Pr[\mathcal{A}(\mathsf{Game}_1(b=1)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_1(b=0)) \to 1]\mid \qquad \text{(By Eq. (7.1))} \\ &\le \sum_{j=2}^{n}(|\Pr[\mathcal{A}(\mathsf{Game}_{j-1}(b=1)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_j(b=0)) \to 1]| \\ &\quad + |\Pr[\mathcal{A}(\mathsf{Game}_{j-1}(b=1)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_1(b=0)) \to 1]|) \\ &\quad + \mid \Pr[\mathcal{A}(\mathsf{Game}_n(b=1)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_{j-1}(b=1)) \to 1]\mid \\ &\qquad\qquad \text{(Union bounds.)} \\ &= \sum_{j=2}^{n-1} \mathsf{Adv}^{\mathsf{mcDDH}}(\mathcal{B}_j). \qquad \text{(By Eqs. (7.2) and (7.3))}\end{aligned}$$

The result follows. □

7.4 Proof Systems

7.4.1 The Action Factorization Relation and Its Sigma-Protocol

We consider the following action factorization relation R_{fac} for our verifiable random functions.

$$R_{\mathsf{fac}} = \left\{ \mathsf{st} = (E_0, \{E_i\}_{i\in[N]}, E), \mathsf{wt} = \{s_i\}_{i\in[N]} \middle| \begin{array}{l} E_i = s_i \star E_0 \ \forall\ i \in [N] \\ E = (\Pi_{i=1}^{N} s_i) \star E_0 \end{array} \right\}.$$

Sigma Protocol for R_{fac}. We give a basic sigma protocol for R_{fac} as described in Fig. 7.1. Let $N \in \mathbb{N}$ and a statement $(\mathsf{st} = E_0, \{E_i\}_{i\in[N]}, E)$. Say the prover has the witness $(\mathsf{wt} = \{s_i\}_{i\in[N]})$ such that $E_i = s_i \star E_0$ for any $i \in [N]$ and $E = (\Pi_{i=1}^{N} s_i) \star E_0$.

To prove the knowledge, the prover firstly generates $r_1, \cdots, r_N$, computes $E'_i = r_i \star E_i$ for all $i \in [N]$ and $E' = (\Pi_{i=1}^{N} r_i) \star E$, and sends those $N+1$ set elements to the verifier. The verifier returns a random challenge c from $\{0,1\}$ and sends it to the prover. If the challenge is 0, the prover reveals r_i for all $i \in [N]$ to the verifier. Otherwise, the prover reveals $s_i r_i$ for every $i \in [N]$. When $c = 0$, with received $\{r'_i\}_{i\in[N]}$ the verifier checks whether $r'_i \star E_i = E'_i$ for all $i \in [N]$ and whether $E' = (\Pi_{i=1}^{N} r_i) \star E$. When $c = 1$, with received $\{r'_i\}_{i\in[N]}$ the verifier checks whether $r'_i \star E_0 = E'_i$ for all $i \in [N]$ and also $(\Pi_{i=1}^{N} r'_i) \star E_0 = E'$.

In each case, if all equalities hold, the verifier returns 1 to represent the acceptance. Otherwise, the verifier returns 0 to represent the rejection.

Remark 7.4.1. *Constructing the same proof system in a restricted EGA model or the original CSIDH setting [CLM⁺18] with an unknown structure group is feasible. In these settings, the group elements are represented as a linear combination of a given generating set where the coefficients are chosen from a small interval $[-t, t]$. In this case, revealing the addition $s + r$ if both $s, r \in [-t, t]$ will leak the information of the secret s. Therefore, using Fiat-Shamir with aborts [Lyu09, DG19] can circumvent this by sampling r from a larger $[-(T + 1)t, (T + 1)t]$ for some $T \in \mathbb{N}$ and, then, aborting the session while required to reveal $r + s$ and $r + s \notin [-Tt, Tt]$. With a straightforward application to our case of $s_i r_i$ and $\Pi(s_i r_i)$ for $i \in [N]$ and $T = 2\lambda^2$, the abort rate will be larger than $1/3$ (see [DG19, Lemma 2.]). The rejection sampling method can also be improved using [DPV19].*

To reduce the size of the overall response, the prover uses a pseudorandom number generator to generate $r_1, \cdots, r_N \in G$ with a seed, seed_0, picked uniformly at random from $\{0, 1\}^\lambda$. Also, the prover uses the Merkle tree to reduce the communication cost of the first message by producing a root of $\{\{E'_i\}_{i\in[N]}, E'\}$ over $\{0, 1\}^{2\lambda}$.

Theorem 7.4.2. *The sigma protocol $\Pi_\Sigma^{\mathsf{base}}$ described in Fig. 7.1 has correctness.*

Proof. When the challenge is $c = 0$, the prover sends the seed, seed_0, to the verifier. The computation of the verifier will result in the same Merkle root in this case.

When $c = 1$, the prover sends $r'_i = s_i r_i$ for every $i \in [N]$ to the verifier. Recall that for any $i \in [N]$, we have $E_i = s_i \star E_0$, $E'_i = r_i \star E_i$, $E = (\Pi_{i=1}^N s_i) \star E_0$, and $E' = (\Pi_{i=1}^N r_i) \star E$. Also, $E'_i = r_i \star E_i$. Hence, due to commutative G, we have

$$\begin{aligned}(E'_1, \cdots, E'_N, E') &= (r_1 s_1 \star E_0, \cdots, r_N s_N \star E_0, (\Pi_{i=1}^N r_i s_i) \star E_0)\\ &= (r'_1 \star E_0, r'_N \star E_N, (\Pi_{i=1}^N r'_i) \star E_0).\end{aligned}$$

The Merkle tree will result in the same root and correctness follows. □

Theorem 7.4.3. *Let $|G| \geq 2^\lambda$ (see Rem. 7.2.1). The sigma protocol $\Pi_\Sigma^{\mathsf{base}}$ described in Fig. 7.1 has 2-special soundness for the relation R_{fac} if the Merkle tree hash function $\mathcal{O}(\mathsf{MT} \parallel \cdot)$ is collision-resistant. Concretely, for a fixed statement st, there exists an extractor Ext on input two valid transcripts returning either a valid witness wt or a pair $(\mathsf{wt}_1, \mathsf{wt}_2)$ such that $(\mathsf{st}, \mathsf{wt}) \in R_{\mathsf{fac}}$ or $\mathcal{O}(\mathsf{MT} \parallel \mathsf{wt}_1) = \mathcal{O}(\mathsf{MT} \parallel \mathsf{wt}_2)$, respectively.*

Proof. Let $\{\mathsf{root}, 0, \mathsf{resp}_0\}$ and $\{\mathsf{root}, 1, \mathsf{resp}_1\}$ be the two valid transcripts for the same first-message root. Write $r_1, \cdots, r_N \leftarrow \mathcal{O}(\mathsf{PRNG} \parallel \mathsf{resp}_0)$ and $\{r'_1, \cdots, r'_N\} = \mathsf{resp}_1$, the extractor Ext proceeds as follows.

1. Compute $\mathsf{wt}_1 = (r_1 \star E_1, \cdots, r_N \star E_N, (\Pi_{i=1}^N r_i) \star E)$.

```
round 1: $P_1'^{\mathcal{O}}(\mathsf{st} = (E_0, \{E_i\}_{i\in[N]}, E), \mathsf{wt} = \{s_i\}_{i\in[N]})$
1: $\mathsf{seed}_0 \xleftarrow{\$} \{0,1\}^\lambda$
2: $(r_1, \cdots, r_N) \leftarrow \mathcal{O}(\mathsf{PRNG} \parallel \mathsf{seed}_0)$        ▷ Generate $r_i \in G$
3: $E' \leftarrow E$
4: for $i$ from 1 to $N$ do
5:     $E_i' \leftarrow r_i \star E_i$
6:     $E' \leftarrow r_i \star E'$
7: $\mathsf{root} \leftarrow \mathcal{O}(\mathsf{MT} \parallel E_1', \cdots, E_N', E')$        ▷ Produce $\mathsf{root} \in \{0,1\}^{2\lambda}$
8: Prover sends $\mathsf{com} \leftarrow \mathsf{root}$ to Verifier.
```

```
round 2: $V_1'(\mathsf{com})$
1: $c \xleftarrow{\$} \{0,1\}$
2: Verifier sends $\mathsf{ch} \leftarrow c$ to Prover.

round 3: $P_2'(\mathsf{st}, \mathsf{com}, \mathsf{ch})$
1: $c \leftarrow \mathsf{ch}$
2: if $c = 1$ then
3:     for $i$ from 1 to $N$ do
4:         $r_i' \leftarrow s_i r_i$
5:     $\mathsf{resp} \leftarrow \{r_i'\}_{i\in[N]}$
6: else
7:     $\mathsf{resp} \leftarrow \mathsf{seed}_0$
8: Prover sends resp to Verifier
```

```
Verification: $V_2'^{\mathcal{O}}(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$
1: $(\mathsf{root}, c) \leftarrow (\mathsf{com}, \mathsf{ch})$
2: if $c = 1$ then
3:     $(\{r_i'\}_{i\in[N]}) \leftarrow \mathsf{resp}$
4:     $\widetilde{E}' \leftarrow E_0$
5:     for $i$ from 1 to $N$ do
6:         $\widetilde{E}_i' \leftarrow r_i' \star E_0$
7:         $\widetilde{E}' \leftarrow r_i' \star \widetilde{E}'$
8:     $\widetilde{\mathsf{root}} \leftarrow \mathcal{O}(\mathsf{MT} \parallel \widetilde{E}_1', \cdots, \widetilde{E}_N', \widetilde{E}')$
9:     return $\top$ if $\widetilde{\mathsf{root}} = \mathsf{root}$; otherwise, return $\bot$.
10: else
11:     Repeat round 1 with $\mathsf{seed}_0 \leftarrow \mathsf{resp}$.
12:     return $\top$ if results in root; otherwise, return $\bot$.
```

Figure 7.1: Construction of the base sigma protocol $\Pi_\Sigma^{\mathsf{base}} = (P' = (P_1', P_2'), V' = (V_1', V_2'))$ for the relation R where $O(\mathsf{PRNG}\|\cdot)$ and $O(\mathsf{Com}\|\cdot)$ are a PRNG and a commitment scheme instantiated by the random oracle, respectively.

2. Compute $\mathsf{wt}_2 = (r_1' \star E_0, \cdots, r_N' \star E_0, (\Pi_{i=1}^N r_i') \star E_0)$.

3. If $\mathsf{wt}_1 \neq \mathsf{wt}_2$, then return $(\mathsf{wt}_1, \mathsf{wt}_2)$.

4. Else, return $(r_1^{-1} r_1', \cdots, r_N^{-1} r_N')$.

Since $V_2'^{\mathcal{O}}(\{\mathsf{root}, b, \mathsf{resp}_b\}) \to 1$ for $i \in \{0,1\}$, we know have

$$\mathsf{root} = \mathcal{O}(\mathsf{MT} \parallel r_1 \star E_1, \cdots, r_N \star E_N, (\Pi_{i=1}^N r_i) \star E),$$
$$\mathsf{root} = \mathcal{O}(\mathsf{MT} \parallel r_1' \star E_0, \cdots, r_N' \star E_0, (\Pi_{i=1}^N r_i') \star E_0)$$

where $r_1, \cdots, r_N \leftarrow \mathcal{O}(\mathsf{PRNG} \parallel \mathsf{resp}_0)$ and $\{r_1', \cdots, r_N'\} = \mathsf{resp}_1$. If $\mathsf{wt}_1 \neq \mathsf{wt}_2$, then they form a collision for the Merkle tree hash function.

If $\mathsf{wt}_1 = \mathsf{wt}_2$, we have $r_i \star E_1 = r_i' \star E_0$ for any $i \in [N]$ and $(\Pi_{i=1}^N r_i) \star E = (\Pi_{i=1}^N r_i') \star E_0$. If follows that $E_i = (r_i^{-1} r_i') \star E_0$ for all $i \in [N]$. Moreover, since the group is commutative and $(\Pi_{i=1}^N r_i)^{-1}(\Pi_{i=1}^N r_i') \star E_0 = E$, we have $(\Pi_{i=1}^N (r_i^{-1} r_i')) \star E_0 = E$. □

Theorem 7.4.4. *The sigma protocol* $\Pi_\Sigma^{\mathsf{base}}$ *described in Fig. 7.1 is statistically HVZK where the pseudorandom number generator and the Merkle tree hash function are modeled as random oracles* $\mathcal{O}(\mathsf{PRNG} \parallel \cdot)$ *and* $\mathcal{O}(\mathsf{MT} \parallel \cdot)$*, respectively. Concretely, for any* $(\mathsf{st}, \mathsf{wt}) \in R_{\mathsf{fac}}$ *and a computationally-unbounded adversary* $\mathcal{A}$ *with at most* q_H *queries of* $\mathcal{O}(\mathsf{PRNG} \parallel \cdot)$*, there exists a simulator* Sim *such that*

$$\left|\Pr[\mathcal{A}^{\mathcal{O}}(P^{\mathcal{O}}(\mathsf{st}, \mathsf{wt}, c)) = 1] - \Pr[\mathcal{A}^{\mathcal{O}}(\mathsf{Sim}^{\mathcal{O}}(\mathsf{st}, c)) = 1]\right| \leq q_H / 2^\lambda.$$

Proof. Let $(\mathsf{st} = (E_0, \{E_i\}_{i \in [N]}, E), \mathsf{wt} = \{s_i\}_{i \in [N]}) \in R_{\mathsf{fac}}$. Given a st and $c \in \{0, 1\}$, the simulator $\mathsf{Sim}^{\mathcal{O}}(\mathsf{st}, \mathsf{wt}, c)$ proceeds as follows.

1. If $c = 0$, then execute P_1' and generate $(\mathsf{root}, 0, \mathsf{seed}_0)$ where the witness is not required in this process.
2. If $c = 1$, then
   (1.) Generate $r_1', \cdots, r_N' \leftarrow G$ and let $\mathsf{resp} \leftarrow \{r_1', \cdots, r_N'\}$.
   (2.) Compute $E_i' = r_i' \star E_0$ for every $i \in [N]$.
   (3.) Compute $E' = (\Pi_{i=1}^N r_i') \star E_0$.
   (4.) Compute $\mathsf{root} \leftarrow \mathcal{O}(\mathsf{MT} \parallel E_1', \cdots, E_N', E')$.
   (5.) Return $(\mathsf{root}, c, \mathsf{resp})$.

The simulated transcripts are identical to ones produced by the prover with the witness executing the protocol $\Pi_\Sigma^{\mathsf{base}}$. For the case $c = 0$, the procedure is the same since the witness is not involved.

For the case $c = 1$, one can observe that the simulator returns a valid transcript and each element in the response follows the uniform distribution over G. The distribution is the same as the uniform distribution over the coset $(s_i)^{-1}G$ for any $i \in [N]$ used by the prover, since $\mathcal{O}(\mathsf{PRNG} \parallel \cdot)$ is modeled as a random oracle, except for those queries has been made before. Concretely, the difference of two distribution is

$$\begin{aligned}
&1/2 \sum \left|\Pr[(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \leftarrow \widetilde{P}^{\mathcal{O}}(\mathsf{st}, \mathsf{wt}, c))] - \Pr[(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \leftarrow \mathsf{Sim}^{\mathcal{O}}(\mathsf{st}, c)]\right| \\
=&1/2 \sum \left|\Pr[(\mathsf{com}, 1, \mathsf{resp}) \leftarrow \widetilde{P}^{\mathcal{O}}(\mathsf{st}, \mathsf{wt}, c))] - \Pr[(\mathsf{com}, 1, \mathsf{resp}) \leftarrow \mathsf{Sim}^{\mathcal{O}}(\mathsf{st}, c)]\right| \\
=&\frac{q_H}{2}(1/2^\lambda - 1/|G|^N) \\
\leq&\frac{q_H}{2^\lambda},
\end{aligned}$$

so is the advantage of the adversary $\mathcal{A}$. □

Theorem 7.4.5. *Let $|G| \geq 2^\lambda$ (see Rem. 7.2.1). The sigma protocol $\Pi_\Sigma^{\mathsf{base}}$ in Fig. 7.1 has λ min-entropy where $\mathcal{O}(\mathsf{PRNG} \| \cdot)$ and $\mathcal{O}(\mathsf{MT} \| \cdot)$ are instantiated by by a random oracle.*

Proof. When the challenge $\mathsf{ch} = 0$, the seed is drawn uniformly at random from $\{0,1\}^\lambda$, and then r_i are drawn uniformly at random from G for any $i \in [N]$. Note that $|G| \geq 2^\lambda$. Since the action is free and transitive, $r_i \star E_i$ follows the uniform distribution over $\mathcal{E}$ for every i. Then, $\mathsf{com} \in \{0,1\}^{2\lambda}$ is produced by $\mathcal{O}(\mathsf{MT} \| \cdot)$. Throughout the procedure, every random element is drawn from a set larger than 2^λ. Therefore, we have $\Pr\left[\mathsf{com} = \mathsf{com}' \middle| \mathsf{com} \leftarrow P_1^{\mathcal{O}}(\mathsf{st}, \mathsf{wt}), \mathsf{com}' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{st}, \mathsf{wt})\right] \leq 2^{-\lambda}$. □

7.4.2 Online-extractable NIZK

Repeating λ times and using the Fiat-Shamir transform, we turn the sigma protocol Fig. 7.1 into a proof system for the relation R_{fac}. The description is displayed in Fig. 7.2.

```
Prove^O(st = (E_0, {E_i}_{i∈[N]}, E), wt = {s_i}_{i∈[N]})
1: for $i \in [\lambda]$ do
2:     $\mathsf{com}_i \leftarrow P_1'^{\mathcal{O}}(\mathsf{st}, \mathsf{wt})$
3: $\mathsf{com} \leftarrow (\mathsf{com}_1, \cdots, \mathsf{com}_\lambda)$
4: $\mathsf{ch} = (c_1, \cdots, c_\lambda) \leftarrow \mathcal{O}(\mathsf{FS} \| \mathsf{st} \| \mathsf{com})$
5: for $i \in [\lambda]$ do
6:     $\mathsf{resp}_i \leftarrow P_2'^{\mathcal{O}}(\mathsf{st}, \mathsf{com}_i, c_i)$
7: $\mathsf{resp} \leftarrow (\mathsf{resp}_1, \cdots, \mathsf{resp}_\lambda)$
8: return $\pi \leftarrow (\mathsf{com}, \mathsf{ch}, \mathsf{resp})$
```

```
Verify^O(st = (E_0, {E_i}_{i∈[N]}, E), π)
1: $(\mathsf{com} = (\mathsf{com}_1, \cdots, \mathsf{com}_\lambda), \mathsf{ch} = (c_1, \cdots, c_\lambda), \mathsf{resp} = (\mathsf{resp}_1, \cdots, \mathsf{resp}_\lambda)) \leftarrow \pi$
2: $\mathsf{output} = 1$
3: for $i \in [\lambda]$ do
4:     $r \leftarrow V_2'(\mathsf{com}_i, c_i, \mathsf{resp}_i)$
5:     $\mathsf{output} \leftarrow \mathsf{output} \cdot r$
6: $\mathsf{output} \leftarrow \mathsf{output} \cdot (\mathsf{ch} == \mathcal{O}(\mathsf{FS} \| \mathsf{st} \| \mathsf{com}))$
7: return output
```

Figure 7.2: NIZK Π_{NIZK} for the relation $\mathsf{R}_{\mathsf{fac}}$ by applying the Fiat-Shamir transform to $\Pi_\Sigma^{\mathsf{base}} = (P' = (P_1', P_2'), V' = (V_1', V_2'))$ in Fig. 7.1 with λ repetitions.

Theorem 7.4.6 (Completeness)**.** *The proof system Π_{NIZK} for the relation R_{fac} in Fig. 7.2 is complete.*

Proof. In each iteration of $i \in [\lambda]$ in Fig. 7.2, the prover and the verifier execute P' and V' in $\Pi_\Sigma^{\mathsf{base}} = (P', V')$ respectively. By Def. 3.3.2, each execution of $\Pi_\Sigma^{\mathsf{base}} = (P', V')$ has correctness, and the completeness of Π_{NIZK} follows. □

Theorem 7.4.7 (Zero-knowledge)**.** *Let $|G| \geq 2^\lambda$ (see Rem. 7.2.1). The proof system Π_{NIZK} for the relation R_{fac} in Fig. 7.2 is zero-knowledge in the random oracle model. Concretely, for any $(\mathsf{st}, \mathsf{wt}) \in R_{\mathsf{fac}}$ and a computationally-unbounded adversary $\mathcal{A}$ with at most q_{PRNG} queries of $\mathcal{O}(\mathsf{PRNG} \| \cdot)$ and q_{FS} queries of $\mathcal{O}(\mathsf{FS} \| \cdot)$, there exists a simulator Sim such that*

$$\left|\Pr\left[\mathcal{A}^{\mathcal{O}}(P^{\mathcal{O}}(\mathsf{st}, \mathsf{wt})) = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{O}}(\mathsf{Sim}^{\mathcal{O}}(\mathsf{st})) = 1\right]\right| \leq \frac{q_{\mathsf{PRNG}}}{2^\lambda} + \frac{q_{\mathsf{FS}}}{2^{\lambda N}},$$

Proof. Let Sim' be the simulator in Thm. 7.4.4. The simulator Sim firstly simulates the oracle of $\mathcal{O}(\mathsf{FS} \parallel \cdot)$, $\mathcal{O}(\mathsf{FS} \parallel \mathsf{MT})$ and $\mathcal{O}(\mathsf{PRNG} \parallel \cdot)$ by keeping lists $L_{\mathsf{FS}}, L_{\mathsf{MT}}$, and L_{PRNG} respectively using an on-the-fly method. Sim also keeps a list L to simulate the oracle queries. Take $\mathcal{O}(\mathsf{FS} \parallel \cdot)$ for instance, upon receiving an oracle query as $\mathcal{O}(\mathsf{FS} \parallel x)$, the Sim simulates the oracle as follows.

1. Check whether there exists a pair $(x, y) \in L_{\mathsf{FS}}$ for some y. If so, return y.
2. Otherwise draw $y \leftarrow \{0,1\}^\lambda$ uniformly at random. Add y to the list (x, y) and return y.

Given a statement st in the language of R_{fac}, the simulator Sim simulates the transcripts as follows.

1. Generate $\mathsf{ch} = (c_1, \cdots, c_\lambda) \leftarrow \{0,1\}^\lambda$ uniformly at random.
2. For each $i \in [\lambda]$, run $(\mathsf{com}_i, c_i, \mathsf{resp}_i) \leftarrow \mathsf{Sim}'(\mathsf{st}, c_i)$.
3. Concatenate $\mathsf{com} \leftarrow (\mathsf{com}_1, \cdots, \mathsf{com}_\lambda)$, $\mathsf{resp} \leftarrow (\mathsf{resp}_1, \cdots, \mathsf{resp}_\lambda)$.
4. Add $(\mathsf{com}, \mathsf{ch})$ to the list L_{FS}. If com has been queried before, abort and return $\perp$.
5. Output the transcript $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$.

By Thm. 7.4.5, we know each generation com_i has λ min-entropy. Therefore, the abort in Item 4 occurs with a negligible probability $q_{\mathsf{FS}}/2^{\lambda N}$.

Given such a distinguisher $\mathcal{A}$, one can construct an HVZK adversary $\mathcal{B}$ against the sigma-protocol $\Pi_\Sigma^{\mathsf{base}}$ using $\mathcal{A}$. Recall that when the challenge is 0, the simulation of $\mathsf{Sim}'(\cdot, 0)$ is perfect. The reduction $\mathcal{B}$ using $\mathcal{A}$ proceeds as follows. Upon receiving the statement st and the transcript ensemble $X = \{\mathsf{com}_i, 1, \mathsf{resp}_i\}_i$ for the challenge 1, $\mathcal{B}$ simulates as what Sim does except that the transcripts from $\mathsf{Sim}'(\mathsf{st}, 1)$ in Item 2 is replace by those taken from the ensemble X. $\mathcal{B}$ invokes $\mathcal{A}$ with st and the simulated transcripts. When the ensemble is generated by a real prover, then $\mathcal{B}$ generates the transcripts as a real prover in Π_{NIZK} except for the occurrence of aborts. When the ensemble is generated by a simulator, then $\mathcal{B}$ generates the transcripts as Sim in Π_{NIZK}. Hence, $\mathsf{Adv}^{\mathsf{ZK}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{HVZK}}_{\Pi_\Sigma^{\mathsf{base}}}(\mathcal{B}) + q_{\mathsf{FS}}/2^{\lambda N}$.

Therefore, we have

$$\left|\Pr\left[\mathcal{A}^{\mathcal{O}}(P^{\mathcal{O}}(\mathsf{st}, \mathsf{wt})) = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{O}}(\mathsf{Sim}^{\mathcal{O}}(\mathsf{st})) = 1\right]\right| \leq \frac{q_{\mathsf{PRNG}}}{2^\lambda} + \frac{q_{\mathsf{FS}}}{2^{\lambda N}}.$$

□

Theorem 7.4.8 (Online-extractable). *Assume $\mathcal{O}(\cdot)$ is collision resistant, $|G| \geq 2^\lambda$ (see Rem. 7.2.1), and $N \in \mathbb{N}$. The proof system Π_{NIZK} in Fig. 7.2 is online-extractable. Concretely, for any adversary $\mathcal{A}$ with q_{FS} queries to $\mathcal{O}(\mathsf{FS} \parallel \cdot)$ and q_{PRNG} queries to $\mathcal{O}(\mathsf{PRNG} \parallel \cdot)$,*

$$\mathsf{Adv}^{\mathsf{OE}}_{\Pi_{\mathsf{VRF}}}(\mathcal{A}) \leq \frac{q_{\mathsf{FS}} + 1}{2^\lambda} + \frac{q_{\mathsf{FS}} q_{\mathsf{PRNG}}}{2^{N\lambda}}.$$

Proof. With the extractability access to the oracle, the extractor Ext observes the queries to $\mathcal{O}$ of the form $(\mathsf{PRNG} \parallel \cdot)$, and record (x, y) to the list L_{PRNG} where x is the input and y is the oracle output. Also, Ext does the same for the queries of the form $(\mathsf{FS} \parallel \cdot)$, and keeps a list L_{FS}. We say x is in the list L_{PRNG} if there exists some y such that $(x, y) \in L_{\mathsf{PRNG}}$.

Upon receiving a statement $\mathsf{st} = (E_0, (E_1, \cdots, E_N), E')$, possibly not in the language of $\mathsf{R}_{\mathsf{fac}}$, and a valid proof $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$, the extractor Ext proceeds as follows.

1. Parse $\mathsf{ch} = (c_1, \cdots, c_\lambda)$ where $c_k \in \{0, 1\}$ for $i \in [\lambda]$. Also, parse $\mathsf{com} = (\mathsf{root}_1, \cdots, \mathsf{root}_\lambda)$ and $\mathsf{resp} = (\mathsf{resp}_1, \cdots, \mathsf{resp}_\lambda)$.

2. Collect $K \subseteq [\lambda]$ where $c_k = 1$ for any $k \in K$.

3. Collect the queries $S = \{\mathsf{seed}_j\}_{j \in [q_{\mathsf{PRNG}}]}$ recorded in list L_{PRNG}.

4. Find one $(k, j) \in K \times [q_{\mathsf{PRNG}}]$ such that $\mathsf{root}_k, \mathsf{seed}_j$ satisfy $\mathsf{root}_k = \mathcal{O}(\mathsf{MT} \parallel (r_1, \cdots, r_N) \star (E_1, \cdots, E_N))$ where $(r_1, \cdots, r_N) \leftarrow \mathcal{O}(\mathsf{PRNG} \parallel \mathsf{seed}_j)$. If no such pairs found, return $\perp$.

5. Execute the extractor Ext' described in Thm. 7.4.3 on input two valid transcripts $(\mathsf{com}_k, 0, \mathsf{seed}_j), (\mathsf{com}_k, 1, \mathsf{resp}_k)$ to extract $\mathsf{wt} \in G^N$ and return wt.

We have to argue the pair (k, j) in Item 4 exists with an overwhelming probability.
For simplicity, we say a seed seed can *serve as a 0-response* for root if $(r_1, \cdots, r_N) \leftarrow \mathcal{O}(\mathsf{PRNG} \parallel \mathsf{seed})$ and $\mathsf{root} = \mathcal{O}(\mathsf{MT} \parallel (r_1, \cdots, r_N) \star \mathsf{st}, (\Pi r_i) \star E_0)$. For example, one can interpret Item 4 as finding a 0-response for root_k for some $k \in K$.

Case I: $\mathcal{O}(\mathsf{FS} \parallel \mathsf{st} \parallel \mathsf{com})$ has not been queried before the verification. This implies that $\mathcal{A}$ produces com and resp without knowing the challenge. However, it requires ch equals $\mathcal{O}(\mathsf{FS} \parallel \mathsf{st} \parallel \mathsf{com})$ in the verification process. This occurs with a probability not greater than $1/2^\lambda$.

Analysis. We analyze the advantage of $\mathcal{A}$ against Ext by aiming at each FS challenge query made by the adversary to $\mathcal{O}(\mathsf{FS} \parallel \mathsf{st}' \parallel \cdot)$ for some st'. We analyze when $\mathcal{A}$ submits a new $\mathsf{com}' = (\mathsf{root}'_1, \cdots, \mathsf{root}'_\lambda)$ to the FS oracle, whether there exist 0-responses in the query list L_{PRNG}.

For $K' \subseteq [\lambda]$, we define the $\mathsf{E}_{K'}$ that when $\mathcal{A}$ submitting com to the FS oracle of the form $(\mathsf{FS} \parallel \mathsf{st}' \parallel \mathsf{root}_1, \cdots, \mathsf{root}_\lambda)$ to the random oracle, there exist no 0-responses in the query list L_{PRNG} for root_k for any $k \in [K']$. We also define event $\mathsf{F}_{K'}$ that the FS oracle returns the challenge $(c'_1, \cdots, c'_\lambda)$ where $c'_k = 1$ for all $k \in K'$ and $c_k = 0$ otherwise. Obviously, $\Pr[\mathsf{F}_{K'}] = 1/2^\lambda$ for every new FS query. Denote the event that $\mathcal{A}$ outputs a transcript containing com' by $\mathsf{O}_{\mathsf{com}'}$ (e.g. $(\mathsf{com}', \mathsf{ch}', \mathsf{resp}')$) and the output is extractable for Ext by $\mathsf{L}_{\mathsf{com}'}$. The latter case implies $\mathcal{A}$ fails.

Note that $\mathsf{E}_{K'}$ forms a partition. Therefore, if $\mathcal{A}$ returns $(\mathsf{com}', \mathsf{ch}', \mathsf{resp}')$ we have

$$\begin{aligned}
\Pr[\mathsf{O}_{\mathsf{com}'}] &= \sum_{K' \subseteq [\lambda]} \Pr[\mathsf{O}_{\mathsf{com}'} \cap \mathsf{E}_{K'}] \\
&= \Pr[\mathsf{O}_{\mathsf{com}'} \cap \mathsf{E}_{K'}], \text{ for some } K' \\
&= \Pr[\mathsf{O}_{\mathsf{com}'} \cap \mathsf{E}_{K'} \cap \mathsf{F}_{K'}] + \Pr[\mathsf{O}_{\mathsf{com}'} \cap \mathsf{E}_{K'} \cap \neg\mathsf{F}_{K'}] \\
&\leq 1/2^{\lambda} + \Pr[\mathcal{A} \text{ wins using } \mathsf{com}' \cap \mathsf{E}_{K'} \cap \neg\mathsf{F}_{K'}] + \Pr[\mathsf{L}_{\mathsf{com}'} \cap \mathsf{E}_{K'} \cap \neg\mathsf{F}_{K'}],
\end{aligned}$$

where $\Pr[\mathsf{O}_{\mathsf{com}'} \cap \mathsf{E}_{K'} \cap \mathsf{F}_{K'}] \leq 1/2^{\lambda}$ since $\Pr[\mathsf{F}_{K'}] = 1/2^{\lambda}$. We partition the event that $\mathsf{O}_{\mathsf{com}'} \cap \mathsf{E}_{K'} \cap \neg\mathsf{F}_{K'}$ into two cases: $\mathcal{A}$ wins or not (i.e. whether the tuple $(\mathsf{com}', \mathsf{ch}', \mathsf{resp}')$ is extractable).

Case II: $\mathcal{A}$ wins with a tuple using $\mathsf{com}' \cap \mathsf{E}_{K'} \cap \neg\mathsf{F}_{K'}$. Recall that if there exists $k \in [\lambda] - K'$ such that $c'_k = 1$, then one can invoke Ext to extract the witness using resp'_k and the list of $\mathcal{O}(\mathsf{PRNG} \,\|\, \cdot)$. Therefore, the case that $\mathcal{A}$ wins implies that $c'_k = 0$ for all $k \in [\lambda] - K'$ and $\mathcal{A}$ produces a seed seed_k for some $c'_k = 0, k \in K'$ such that $\mathsf{com}'_k = (r_1, \cdots, r_N) \star E_0$ where $(r_1, \cdots, r_N) \leftarrow \mathcal{O}(\mathsf{PRNG} \,\|\, \mathsf{seed}_k)$. Note that such seed_k is generated after the FS query. Since the protocol has the unique response property [3] and the group elements are generated uniformly from G by $\mathcal{O}(\mathsf{PRNG} \,\|\, \cdot)$, the adversary can generate such a seed with chance not greater than $q_{\mathsf{PRNG}}/|G|^N$.

Therefore,

$$|\Pr[\mathsf{O}_{\mathsf{com}'}] - \Pr[\mathsf{L}_{\mathsf{com}'}]| \leq 1/2^{\lambda} + q_{\mathsf{PRNG}}/|G|^N.$$

Wrapping up, given an adversary with q_{FS} FS queries and q_{PRNG} PRNG queries, by taking a union bound over all FS queries we know the advantage of the adversary:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{OnlineExtract}}_{\Pi_{\mathsf{VRF}}}(\mathcal{A}) &\leq \Pr[\text{Case I}] + \sum_{\mathsf{com} \text{ in } L_{\mathsf{FS}}} \Pr[\text{Case II wrt } \mathsf{com}] \\
&\leq \frac{1}{2^{\lambda}} + \sum_{\mathsf{com} \text{ in } L_{\mathsf{FS}}} |\Pr[\mathsf{O}_{\mathsf{com}'}] - \Pr[\mathsf{L}_{\mathsf{com}'}]| \\
&\leq \frac{q_{\mathsf{FS}} + 1}{2^{\lambda}} + \frac{q_{\mathsf{FS}} q_{\mathsf{PRNG}}}{|G|^N}.
\end{aligned}$$

□

[3]Given $\mathbf{E} \in \mathcal{E}^N$ there exist two unique group elements $\mathbf{g} \in G^N$ and $\mathbf{g} \in G'^N$ such that $\mathbf{E} = \mathbf{g} \star (E_1, \cdots, E_N)$ and $\mathbf{E} = \mathbf{g}' \star E_0$.

7.5 Verifiable Random Functions from Effective Group Actions

In this section, we present our first VRF construction from an effective group action – CAPYBARA (Compact Action factorization Proofs Yielded By A RAndom function):

Construction. $\Pi_{\mathsf{VRF}} = \{\mathsf{ParGen}, \mathsf{KeyGen}, \mathsf{VRFEval}, \mathsf{Ver}\}$ using $\Pi^{\mathsf{fac}}_{\mathsf{NIZK}} = (P, V)$, H where:

- $\mathsf{ParGen}(1^\lambda)$: on input a security parameter 1^λ, it returns $\mathsf{pp} = (G, \mathcal{E}, \star, E_0)$, which is a free, transitive and effective group action.
- $\mathsf{KeyGen}(\mathsf{pp})$: On input public parameter $\mathsf{pp} = (G, \star, E_0, \mathcal{E})$, it returns a secret key $\mathsf{sk} = (c_0, c_1, s_1, \cdots, s_\lambda)$ and a public key $\mathsf{vk} = (c_0 \star E_0, c_1 \star E_0, s_1 \star E_0, \cdots, s_\lambda \star E_0)$.
- $\mathsf{VRFEval}(\mathsf{sk}, x)$ [4]: On input a secret key sk and an input $x = (x_i) \in \{0,1\}^\lambda$, this algorithm outputs (v, π) for the VRF value where $v = (c_0 c_1 \Pi_{i=1}^{\lambda} s_i^{x_i}) \star E_0$ together with the corresponding proof π where $I = \{1,2\} \cup \{i+2 | x_i = 1 \wedge i \in [\lambda]\}$ and $\pi \leftarrow P(\mathsf{st} = (E_0, \mathsf{vk}_I, v), \mathsf{wt} = \mathsf{sk}_I)$ of Π_{NIZK}.
- $\mathsf{Ver}(\mathsf{vk}, v, x, \pi)$: On input (vk, v, x, π), this algorithm computes $b \leftarrow V(\mathsf{st} = (E_0, \mathsf{vk}_I, v), \pi)$ using Π_{NIZK} where $I = \{1,2\} \cup \{i+2 | x_i = 1 \wedge i \in [\lambda]\}$, and returns b.

Theorem 7.5.1. *The VRF construction Π_{VRF} in Fig. 7.3 has provability.*

Proof. Let $(E, \pi) \leftarrow \mathsf{VRFEval}(\mathsf{sk}, x)$ and $v = (c_0 c_1 \Pi_{i=1}^{\lambda} s_i^{x_i}) \star E_0$. The proof π is generated by $P(\mathsf{st} = (E_0, \mathsf{vk}_I, v), \mathsf{wt} = \mathsf{sk}_I)$ and $I = \{1,2\} \cup \{i+2 | x_i = 1 \wedge i \in [\lambda]\}$. Since $(\mathsf{st} = (E_0, \mathsf{vk}_I, v), \mathsf{wt} = \mathsf{sk}_I) \in R$ and Π_{NIZK} has correctness, we have $\mathsf{VRFVer}(\mathsf{vk}, v, x, \pi) = 1$. □

Theorem 7.5.2. *If Π_{NIZK} is extractable, the VRF construction Π_{VRF} in Fig. 7.3 has computational full uniqueness in the random oracle model. Concretely, for any full uniqueness adversary $\mathcal{A}$ against Π_{VRF}, there exists an extractable adversary $\mathcal{B}$ against Π_{NIZK} such that*

$$\mathsf{Adv}^{\mathsf{UP}}_{\Pi_{\mathsf{VRF}}}(\mathcal{A}) \leq 2\mathsf{Adv}^{\mathsf{OE}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{B}).$$

Proof. Given $(\mathsf{vk}, x, v_1, v_2, \pi_1, \pi_2) \leftarrow \mathcal{A}$ where $\mathsf{VRFVer}((E_0, \mathsf{vk}_I, v_1), \pi_1) = \mathsf{VRFVer}((E_0, \mathsf{vk}_I, v_2), \pi_2) = 1$ and $v_1 \neq v_2$. where $I = \{1,2\} \cup \{i+2 | x_i = 1 \wedge i \in [\lambda]\}$.

By invoking the extractor Ext of Π_{NIZK} in Thm. 7.4.8 twice, we have $\mathsf{s}_1 \leftarrow \mathsf{Ext}((E_0, \mathsf{vk}_I, v_1), \pi_1)$, $\mathsf{s}_2 \leftarrow \mathsf{Ext}((E_0, \mathsf{vk}_I, v_2), \pi_2)$ such that $v_1 = (\Pi_i(\mathsf{s}_1)_i) \star E_0$ and $v_2 = (\Pi_i(\mathsf{s}_2)_i) \star E_0$. Also, $\mathsf{vk}_I = \mathsf{s}_1 \star E_0$ and $\mathsf{vk}_I = \mathsf{s}_2 \star E_0$. Since the action is free and transitive, we have $\mathsf{s}_1 = \mathsf{s}_2$, which contradicts $v_1 \neq v_2$.

In other words, if $\mathcal{A}$ wins, then the extractor $\mathcal{E}$ shall fail among two extractions. We can therefore tranform $\mathcal{A}$ into an extractabililty adversary $\mathcal{B}$ against Π_{NIZK}. Concretely, if $\mathcal{A}$ returns

[4]In the formal syntax of VRF, vk is not inclueded in the $\mathsf{VRFEval}$. One can also include vk as part of the public key. In our case, the user can recover vk from sk. Both justify the notation here.

```
ParGen(1^λ)
1: Generate $\mathsf{pp} = (G, \star, E_0, \mathcal{E})$
2: return $\mathsf{pp}$
```

```
KeyGen(pp)
1: $(G, \star, E_0, \mathcal{E}) \leftarrow \mathsf{pp}$
2: $\mathsf{sk} \leftarrow G^{\lambda+2}$
3: $\mathsf{vk} = \mathsf{sk} \star E_0$
4: return $(\mathsf{vk}, \mathsf{sk})$
```

```
VRFEval(sk, x)
1: $(G, \star, E_0, \mathcal{E}) \leftarrow \mathsf{pp}$
2: $v = E_0$
3: $I \leftarrow \{1, 2\}$
4: for $i \in [\lambda]$ do
5:     if $x_i = 1$ then
6:         $I \leftarrow I \cup \{i + 2\}$
7: for $s \in \mathsf{sk}_I$ do
8:     $v \leftarrow s \star v$
9: $\pi \leftarrow P(\mathsf{st} = (E_0, \mathsf{vk}_I, v), \mathsf{wt} = \mathsf{sk}_I)$
10: return $(v, \pi)$
```

```
VRFVer(vk, v, x, π)
1: $(G, \star, E_0, \mathcal{E}) \leftarrow \mathsf{pp}$
2: for $i \in [\lambda]$ do
3:     if $x_i = 1$ then
4:         $I \leftarrow I \cup \{i + 2\}$
5: return $V(\mathsf{st} = (E_0, \mathsf{vk}_I, v), \pi)$
```

Figure 7.3: The verifiable random function scheme Π_{VRF} based an effective group action and on the DDH problem where $\Pi^{\mathsf{fac}}_{\mathsf{NIZK}} = (P, V)$ is an NIZK for the relation R_{fac} described in Sec. 7.4.2.

$(\mathsf{vk}, x, v_1, v_2, \pi_1, \pi_2)$, then $\mathcal{B}$ randomly outputs one of $((E_0, \mathsf{vk}_I, v_1), \pi_1)$ or $((E_0, \mathsf{vk}_I, v_2), \pi_2)$ where $I = \{1, 2\} \cup \{i + 2 | x_i = 1 \wedge i \in [\lambda]\}$. Therefore, we have

$$\mathsf{Adv}^{\mathsf{UP}}_{\Pi_{\mathsf{VRF}}}(\mathcal{A}) \leq 2\mathsf{Adv}^{\mathsf{OE}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{B}).$$

□

Theorem 7.5.3. *If the decisional master DDH problem is hard, then the VRF construction Π_{VRF} in Fig. 7.3, with a subroutine $\Pi_{\mathsf{NIZK}} = (P, V)$ in Fig. 7.2, has (residual) pseudorandomness. Concretely, for any residual pseudorandomness adversary $\mathcal{A}$ against Π_{VRF} with at most q_{PRNG} queries of $\mathcal{O}(\mathsf{PRNG} \,\|\, \cdot)$ and q_{FS} queries of $\mathcal{O}(\mathsf{FS} \,\|\, \cdot)$, there exists a MDDH adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathsf{PR}}_{\Pi_{\mathsf{VRF}}}(\mathcal{A}) \leq \frac{q_{\mathsf{PRNG}}}{2^{\lambda}} + \frac{q_{\mathsf{FS}}}{2^{2\lambda}} + \mathsf{Adv}^{\mathsf{MDDH}}(\mathcal{B}).$$

Proof. We show by using a hybrid argument that such an adversary $\mathcal{A}$ can be transformed into a MDDH adversary $\mathcal{B}_2$. Let Game_0 be the original residual pseudorandomness experiment and Game_1 be the modified experiment. For $i \in \{0, 1\}$, we denote the advantage of $\mathcal{A}$ in Game_i by $\mathsf{Adv}_i(\mathcal{A}) = |\Pr[\mathcal{A}(\mathsf{Game}_i(b = 1)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_i(b = 0)) \to 1]|$, where $b \in \{0, 1\}$ represents the random coin chosen by the challenger (Def. 7.2.3 Item 7). Since Game_0 is the original experiment, we know $\mathsf{Adv}_0(\mathcal{A}) = \mathsf{Adv}^{\mathsf{PR}}_{\Pi_{\mathsf{VRF}}}(\mathcal{A})$ by definition.

We introduce Game_1 which is the same as Game_0 except for the way of evaluating x for a query. Rather than generated via Prove from the subroutine Π_{NIZK}, the proof is generated using the simulator Sim for Π_{NIZK} in Thm. 7.4.7. Here, we have the parameter $N \geq 2$ in Thm. 7.4.7 since the prover uses at least two elements ($\mathsf{sk}_{\{1,2\}}$ is a sub-array of sk_I) to generate a proof. By Thm. 7.4.7, since the simulator Sim is statistically indistinguishable from a real prover, the change in Game_1 results in a negligible loss. Concretely, $|\mathsf{Adv}_0(\mathcal{A}) - \mathsf{Adv}_1(\mathcal{A})| \leq \frac{q_{\mathsf{PRNG}}}{2^\lambda} + \frac{q_{\mathsf{FS}}}{2^{2\lambda}}$.

We now transform an adversary in Game_1 into a MDDH problem adversary $\mathcal{B}$. The reduction $\mathcal{B}$ starts the MDDH problem with parameter $n = \lambda \in \mathbb{N}$, receives $(E_1, \cdots, E_\lambda)$, and proceeds as follows.

1. First, $\mathcal{B}$ simulates the oracle of $\mathcal{O}(\mathsf{FS} \parallel \cdot)$, $\mathcal{O}(\mathsf{FS} \parallel \mathsf{MT})$ and $\mathcal{O}(\mathsf{PRNG} \parallel \cdot)$ by keeping lists $L_{\mathsf{FS}}, L_{\mathsf{MT}}$, and L_{PRNG} respectively using the straight-line and on-the-fly method. $\mathcal{B}$ also keeps a list L to simulate the oracle queries. Take $\mathcal{O}(\mathsf{FS} \parallel \cdot)$ for instance; upon receiving an oracle query as $\mathcal{O}(\mathsf{FS} \parallel x)$, the $\mathcal{B}$ simulates the oracle as follows.
   (a) Check whether there exists a pair $(x, y) \in L_{\mathsf{FS}}$ for some y. If so, return y.
   (b) Otherwise draw $y \leftarrow \{0,1\}^\lambda$ uniformly at random. Add y to the list (x, y) and return y.
2. Generates $c_0, c_1 \leftarrow G$.
3. Invoke $\mathcal{A}$ with $\mathsf{vk} = (c_0 \star E_0, c_1 \star E_0, E_1, \cdots, E_\lambda)$.
4. Upon receiving the evaluation query $x \in \{0,1\}^\lambda$, forward the query x to the MDDH problem oracle and recieve E. Run the simulator in Thm. 7.4.7 to produce a proof $\pi \leftarrow \mathsf{Sim}(E_0, \mathsf{vk}_I, (c_0 c_1) \star E)$ where $I = \{1, 2\} \cup \{i + 2 | x_i = 1 \wedge i \in [\lambda]\}$. Return (x, π) to $\mathcal{A}$.
5. Upon receiving the challenge $\widetilde{x}$, forward the challenge to $\widetilde{x}$ to the MDDH problem challenger and obtains v_b. Forward v_b to $\mathcal{A}$ and output whatever $\mathcal{A}$ returns.

When the MDDH problem challenger using the random coin $b \in \{0,1\}$ in the experiment (Prob. 10 Item 4). $\mathcal{B}$ creates Game_1 using the same random coin b. Therefore,

$$\begin{aligned}\mathsf{Adv}_1(\mathcal{A}) &= |\Pr[\mathcal{A}(\mathsf{Game}_i(b=1)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_i(b=0)) \to 1]| \\ &= \left|\Pr[\mathcal{B}_2(\mathsf{Exp}^{\mathsf{MDDH}}(\lambda, 1)) \to 1] - \Pr[\mathcal{B}_2(\mathsf{Exp}^{\mathsf{MDDH}}(\lambda, 0)) \to 1]\right| \\ &= \mathsf{Adv}^{\mathsf{MDDH}}(\mathcal{B}).\end{aligned}$$

Hence,

$$\mathsf{Adv}^{\mathsf{PR}}_{\Pi_{\mathsf{VRF}}}(\mathcal{A}) \leq \frac{q_{\mathsf{PRNG}}}{2^\lambda} + \frac{q_{\mathsf{FS}}}{2^{2\lambda}} + \mathsf{Adv}^{\mathsf{MDDH}}(\mathcal{B}).$$

□

7.6 TSUBAKI - Twist-Square-Based Tweak for Isogenies

This section presents the variant using the CSIDH-based action with quadratic twists. Let $(G, \mathcal{E}, \star, E_0)$ denote the group action where $E_0 \in \mathcal{E}$ denote the element that has the property that $E_0^t = E_0$. Also, for any $(g, E) \in G \times \mathcal{E}$, we have $(g \star E)^t = g^{-1} \star E^t$.

A variant of CAPYBARA is described as follows.

Construction.

```
$\mathsf{ParGen}(1^\lambda)$
1: Generate $\mathsf{pp} = (G, \star, E_0, \mathcal{E})$
2: return $\mathsf{pp}$
```

```
$\mathsf{KeyGen}(\mathsf{pp})$
1: $(G, \star, E_0, \mathcal{E}) \leftarrow \mathsf{pp}$
2: $\mathsf{sk} \leftarrow G^{\kappa+2}$
3: $\mathsf{vk} = \mathsf{sk} \star E_0$
4: return $(\mathsf{vk}, \mathsf{sk})$
```

```
$\mathsf{Expand}_s(\mathsf{sk})$
1: $(c_0, c_1, s_1, \cdots, s_\kappa) \leftarrow \mathsf{sk}$
2: return $(c_0, c_1, s_1, \cdots, s_\kappa, -s_1, \cdots, -s_\kappa)$
```

```
$\mathsf{Expand}_v(\mathsf{vk})$
1: $(X_1, X_1, E_1, \cdots, E_\kappa) \leftarrow \mathsf{vk}$
2: return $(X_1, X_1, E_1, \cdots, E_\kappa, E_1^t, \cdots, E_\kappa^t)$
```

```
$\mathsf{VRFEval}(\mathsf{sk}, x)$
1: $(G, \star, E_0, \mathcal{E}) \leftarrow \mathsf{pp}$
2: $v = E_0$
3: $I \leftarrow \{1, 2\}$
4: for $i \in [\kappa]$ do
5:     if $x_i = 1$ then
6:         $I \leftarrow I \cup \{i + 2\}$
7:     if $x_i = -1$ then
8:         $I \leftarrow I \cup \{i + \kappa + 2\}$
9: $\mathsf{sk}', \mathsf{vk}' \leftarrow \mathsf{Expand}_s(\mathsf{sk}), \mathsf{Expand}_v(\mathsf{vk})$
10: for $s \in \mathsf{sk}'_I$ do
11:     $v \leftarrow s \star v$
12: $\pi \leftarrow (P(\mathsf{st} = (E_0, \mathsf{vk}'_I, v), \mathsf{wt} = \mathsf{sk}'_I))$
13: return $(v, \pi)$
```

```
$\mathsf{VRFVer}(\mathsf{vk}, v, x, \pi)$
1: $(G, \star, E_0, \mathcal{E}) \leftarrow \mathsf{pp}$
2: for $i \in [\kappa]$ do
3:     if $x_i = 1$ then
4:         $I \leftarrow I \cup \{i + 2\}$
5:     if $x_i = -1$ then
6:         $I \leftarrow I \cup \{i + \kappa + 2\}$
7: $\mathsf{vk}' \leftarrow \mathsf{Expand}_v(\mathsf{vk})$
8: return $V(\mathsf{st} = (E_0, \mathsf{vk}'_I, v), \pi)$
```

Figure 7.4: Our verifiable random function scheme $\Pi_{\mathsf{VRF}^\star}$ based on the sDDH problem where $\Pi^{\mathsf{fac}}_{\mathsf{NIZK}} = (P, V)$ is an NIZK for the relation R_{fac} described in Sec. 7.4.2. The input x is ternary of length $\kappa = \lceil \lambda / \log_2(3) \rceil$.

The complete probability and the unique provability hold naturally by embedding $\Pi_{\mathsf{VRF}^\star}$ in Fig. 7.4 back to Π_{VRF} in Fig. 7.3. We therefore skip the proofs here. We only show the residual pseudorandomness of $\Pi_{\mathsf{VRF}^\star}$.

Theorem 7.6.1. *The VRF construction Π_{VRF} in Fig. 7.4 has complete provability.*

Theorem 7.6.2. *If Π_{NIZK} is extractable, the VRF construction Π_{VRF} in Fig. 7.4 has unique provability in the random oracle model. Concretely, for any unique provability adversary $\mathcal{A}$ against $\Pi_{\mathsf{VRF}^\star}$, there exists an extractable adversary $\mathcal{B}$ against Π_{NIZK} such that*

$$\mathsf{Adv}^{\mathsf{UP}}_{\Pi_{\mathsf{VRF}}}(\mathcal{A}) \leq 2\mathsf{Adv}^{\mathsf{OE}}_{\Pi_{\mathsf{NIZK}}}(\mathcal{B}).$$

Theorem 7.6.3. *If the twist decisional master DDH problem is hard, then the VRF construction $\Pi_{\mathsf{VRF}^\star}$ in Fig. 7.4, with a subroutine $\Pi_{\mathsf{NIZK}} = (P, V)$ in Fig. 7.2, has (residual) pseudorandomness. Concretely, for any residual pseudorandomness adversary $\mathcal{A}$ against Π_{VRF} with at most q_{PRNG} queries of $\mathcal{O}(\mathsf{PRNG} \,\|\, \cdot)$ and q_{FS} queries of $\mathcal{O}(\mathsf{FS} \,\|\, \cdot)$, there exists a tMDDH adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathsf{PR}}_{\Pi_{\mathsf{VRF}}}(\mathcal{A}) \leq \frac{q_{\mathsf{PRNG}}}{2^\lambda} + \frac{q_{\mathsf{FS}}}{2^{2\kappa}} + \mathsf{Adv}^{\mathsf{tMDDH}}(\mathcal{B}).$$

Proof. We show by using a hybrid argument that such an adversary $\mathcal{A}$ can be transformed into a tMDDH adversary $\mathcal{B}_2$. Let Game_0 be the original residual pseudorandomness experiment and Game_1 be the modified experiment. For $i \in \{0, 1\}$, we denote the advantage of $\mathcal{A}$ in Game_i by $\mathsf{Adv}_i(\mathcal{A}) = |\Pr[\mathcal{A}(\mathsf{Game}_i(b = 1)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_i(b = 0)) \to 1]|$, where $b \in \{0, 1\}$ represents the random coin chosen by the challenger (Def. 7.2.3 Item 7). Since Game_0 be the original experiment, we know $\mathsf{Adv}_0(\mathcal{A}) = \mathsf{Adv}^{\mathsf{PR}}_{\Pi_{\mathsf{VRF}}}(\mathcal{A})$ by definition.

We introduce Game_1, which is the same as Game_0 except for the way to respond to an evaluation query. Rather than generated via Prove from the subroutine Π_{NIZK}, the proof is generated using the simulator Sim for Π_{NIZK} in Thm. 7.4.7. Here, we have the parameter $N \geq 2$ in Thm. 7.4.7 since the prover uses at least two elements ($\mathsf{sk}_{\{1,2\}}$ is a sub-array of sk_I) to generate a proof. By Thm. 7.4.7, since the simulator Sim is statistically indistinguishable from a real prover, the change in Game_1 results in a negligible loss. Concretely, $|\mathsf{Adv}_0(\mathcal{A}) - \mathsf{Adv}_1(\mathcal{A})| \leq \frac{q_{\mathsf{PRNG}}}{2^\lambda} + \frac{q_{\mathsf{FS}}}{2^{2\kappa}}$.

We now transform an adversary in Game_1 into a tMDDH problem adversary $\mathcal{B}$. The reduction $\mathcal{B}$ starts the tMDDH problem with parameter $n = \kappa \in \mathbb{N}$, receives $(E, (E_1, \cdots, E_\kappa))$, and proceeds as follows.

1. Firstly, $\mathcal{B}$ simulates the oracle of $\mathcal{O}(\mathsf{FS} \,\|\, \cdot)$, $\mathcal{O}(\mathsf{FS} \,\|\, \mathsf{MT})$ and $\mathcal{O}(\mathsf{PRNG} \,\|\, \cdot)$ by keeping lists $L_{\mathsf{FS}}, L_{\mathsf{MT}}$, and L_{PRNG} respectively using the straight-line and on-the-fly method. $\mathcal{B}$ also keeps a list L to simulate the oracle queries. Take $\mathcal{O}(\mathsf{FS} \,\|\, \cdot)$ for instance; upon receiving an oracle query as $\mathcal{O}(\mathsf{FS} \,\|\, x)$, the $\mathcal{B}$ simulates the oracle as follows.
   (a) Check whether there exists a pair $(x, y) \in L_{\mathsf{FS}}$ for some y. If so, return y.
   (b) Otherwise draw $y \leftarrow \{0, 1\}^\kappa$ uniformly at random. Add y to the list (x, y) and return y.
2. Generates $c_0, c_1 \leftarrow G$.
3. Invoke $\mathcal{A}$ with $\mathsf{vk} = (c_0 \star E_0, E, E_1, \cdots, E_\kappa)$.

4. Upon receiving the evaluation query $x \in \{0, \pm 1\}^\kappa$, forward the query x to the tMDDH problem oracle and recieve E. Write $\mathsf{vk}' = (c_0 \star E_0, c_0 \star E, E_1, \cdots, E_\kappa, E_1^t, \cdots, E_\kappa^t)$ and $I = \{1, 2\} \cup \{i + 2 | x_i = 1 \wedge i \in [\kappa]\} \cup \{i + 2 + N | x_i = -1 \wedge i \in [\kappa]\}$. Run the simulator in Thm. 7.4.7 to produce a proof $\pi \leftarrow \mathsf{Sim}(E_0, \mathsf{vk}'_I, c_0 \star E)$. Return (x, π) to $\mathcal{A}$.

5. Upon receiving the challenge $\widetilde{x}$, forward the challenge to $\widetilde{x}$ to the tMDDH problem challenger and obtains v_b. Forward v_b to $\mathcal{A}$ and output whatever $\mathcal{A}$ returns.

When the tMDDH problem challenger using the random coin $b \in \{0, 1\}$ in the experiment (Prob. 10 Item 4). $\mathcal{B}$ creates Game_1 using the same random coin b. Therefore,

$$\begin{aligned}
\mathsf{Adv}_1(\mathcal{A}) &= |\Pr[\mathcal{A}(\mathsf{Game}_i(b = 1)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_i(b = 0)) \to 1]| \\
&= \left|\Pr[\mathcal{B}_2(\mathsf{Exp}^{\mathsf{tMDDH}}(\kappa, 1)) \to 1] - \Pr[\mathcal{B}_2(\mathsf{Exp}^{\mathsf{tMDDH}}(\kappa, 0)) \to 1]\right| \\
&= \mathsf{Adv}^{\mathsf{tMDDH}}(\mathcal{B}).
\end{aligned}$$

Hence,

$$\mathsf{Adv}^{\mathsf{PR}}_{\Pi_{\mathsf{VRF}^*}}(\mathcal{A}) \leq \frac{q_{\mathsf{PRNG}}}{2^\lambda} + \frac{q_{\mathsf{FS}}}{2^{2\kappa}} + \mathsf{Adv}^{\mathsf{tMDDH}}(\mathcal{B}).$$

□

7.7 Optimization and Performance

We ameliorate the proof size by utilizing the two techniques presented in [BKP20]. We briefly summarise the results.

Unbalanced Challenge Space. One can observe the response of a prover in the proof system Fig. 7.2 for the challenge 0 is much shorter than the one for the challenge 1. The former is a single seed, while the latter is a bunch of group elements. By introducing the unbalanced challenge space $C_{M,K} = \{\mathsf{ch} \in \{0, 1\}^M \mid |\mathsf{ch}| = K\}$, where $|\cdot|$ is the ℓ_1-norm and $2^\lambda \leq \frac{M!}{K!(M-K)!}$. We thereby obtain a much smaller proof size while the online-extractability and zero-knowledge remain the same.

Seed Trees. The seed tree technique allows the prover to produce a large amount of the seeds using PRNG and iteratively generating binary subtrees. The leaves of the tree are the seeds to be used. The prover can later reveal the generating nodes while not disclosing the information of those unrevealed leaves. The method reduces the size of responses for the challenge 0 in our case. Though the proof size regarding this technique is not fixed, we will calculate the worst case for the proof size estimation.

The performances of CAPYBARA and TSUBAKI are given in Tab. 7.1 for the input space $\{0, 1\}^{128}$. CAPYBARA is based on the standard DDH assumption while TSUBAKI is based on the stronger square DDH (Def. 7.2.6), of which the computational version is as hard as the group action inverse problem (Dlog). A very recent work [DHK+23] justifies the hardness of sDDH in a generic model for group actions. We use the group action from CSIDH-512, as specified in [BKV19], with $M = 855$ and $K = 19$ as the unbalanced challenge space in our implementation. Our proof sizes are flexible and depend on the input length, with lengths of approximately $79|x|/128$ for CAPYBARA and $51|x|/81$ for TSUBAKI. The group action from CSIDH-512 has been estimated to have 128 bits of classical security and over 60 bits of quantum security [Pei20]. We also compare our VRFs to other existing post-quantum VRFs, including LB-VRF [EKS+21], X-VRF, SL-VRF [BDE+22], and LaV [ESLR22], all aiming to meet the NIST II security level. LB-VRF and X-VRF have limited residual pseudorandomness, while SL-VRF, LaV, and our VRFs are full VRFs. The security of X-VRF and SL-VRF is based on XMSS, LowMC, respectively, and LB-VRF and LaV rely on a hybrid lattice assumption MSIS/MLWE and MSIS/MLWR respectively.

	$\|\mathsf{sk}\|$	$\|\mathsf{vk}\|$	$\|v\|$	$\|\pi\|$	Assumption	Relaxation
CAPYBARA [Fig. 7.3]	32B	8.3KB	64B	39 KB	DDH (Prob. 4)	
TSUBAKI [Fig. 7.4]	32B	5.3KB	64B	34 KB	sDDH (Def. 7.2.6)	
LB-VRF I [EKS+21]		3.3KB	84B	4.9KB	MSIS/MLWE	1-Time
LB-VRF III [EKS+21]		3.4KB	84B	7.3KB	MSIS/MLWE	5-Time
X-VRF[BDE+22]	132B	64B	32B	2.6 KB	XMSS	2^{15}-Time
SL-VRF[BDE+22]	24B	48B	32B	40 KB	LowMC	
LaV [ESLR22]	6.4KB	3.4KB	124B	12 KB	MSIS/MLWR	

Table 7.1: CAPYBARA and TSUBAKI (Figs. 7.3 and 7.4, resp) using the group action setting CSIDH-512 instantiated in [BKV19]. The unbalanced challenge space $C_{M,K}$ where $M = 855, K = 19$ is used in the proof system Fig. 7.2. Note that our original secret key sizes are 2KB, 1.3KB, respectively, and one can use a 32B seed to generate the entire secret key sk using a PRNG. Our proof sizes are $\approx 79|x|/128$ and $\leq 51|x|/81$ respectively and vary with the density $|x|/\kappa$ of the input x where $|\cdot|$ is the ℓ_1-norm. The notations $|\mathsf{sk}|, |\mathsf{vk}|, |v|, |\pi|$ represent the length of the secret key, verification key, output, and proof, respectively. The security of X-VRF and SL-VRF is based on XMSS and LowMC, respectively, and LB-VRF and LaV rely on a hybrid lattice assumption MSIS/MLWE and MSIS/MLWR respectively.

Chapter 8

Isogeny Path Knowledge

This chapter presents the work carried out in [CLL23], which the author of the thesis co-authored. It is worth noting that Cong gives the main effort of implementation and changing the setting of the `libiop` library. We have an agreement with our coauthors to maintain equal credit distribution for the remainder of the paper. This chapter has been adapted by removing the information of Limbo and different forms of prime numbers without compromising the essence of the original work.

Abstract. This chapter presents the first practical application of a generic proof system to isogeny cryptography for proving knowledge of an isogeny path as an identification scheme. The scheme provides proof of an exact relation and offers statistical zero-knowledge, a feature that has only been achieved by very recent work presented at Eurocrypt'23. The resulting scheme has a proof size comparable to the state-of-the-art work and faster prover and verification times by a factor of 3-10.

8.1 Introduction

Identification schemes and proofs of knowledge are essential tools in cryptography. They allow someone's identity to be confirmed or a prover to convince any person that they have specific knowledge without revealing it. In real-world scenarios, identity can be established by presenting physical credentials like a passport, ID card, or driving license or by providing knowledge-based information such as a password, date of birth or phone number. However, this information can be recorded by a malicious listener and later used for impersonation, making zero-knowledge identification schemes essential. That is, we hope the prover does not "give away the knowledge" when identifying herself.

For instance, in cryptography there are message authentication codes (MACs) to verify message authenticity, and trusted authorities issue certificates to authenticate website identities. One of the most well-known identification schemes is the Schnorr identification scheme which enables non-interactive and zero-knowledge proof of a discrete logarithm solution. Similarly, proving isogeny knowledge is critical in isogeny-based cryptography. In previous sections, we have demonstrated several cryptosystems together with a variety of proof systems. Actually, in the realm of group actions in isogeny cryptography, it has been known to prove an isogeny knowledge is not difficult at all and more sophisticated relations can be made [DG19, BKV19, DM20, BKP20, BDK$^+$22] thanks to the algebraic structure of the underlying group actions.

However, this is not the case in general in isogeny cryptography. For instance, in the case of SIDH, which we assume to be secure in this paragraph, proving the knowledge of the public key is essential for achieving a secure static-static key exchange protocol. A static-static protocol enables participants to execute the desired primitives without changing the public keys from time to time. The main bottleneck for SIDH-family schemes to achieve the static-static property due to the adaptive attacks [GPST16, GL22] that allow an attacker to extract secret keys bit by bit. To address this, a zero-knowledge proof can be embedded, as done in previous works [DFJP14, UJ20], which is thought to be the cheapest method before SIDH got broken by [CD23, Rob23]. Even though there is a flaw in the soundness proof pointed out in [DDGZ22, GPV21], it does not impact the NIZK's applications [UJ20] for SIDH or signature schemes [YAJ$^+$17].

In addition, there is a compact identification/signature scheme [DKL$^+$20] that proves the knowledge of two isogenous supersingular curves[1]. This scheme results in a signature size of only 204 bytes for the NIST-1 level of security. However, none of the aforementioned identification schemes [DFJP14, YAJ$^+$17, UJ20, DKL$^+$20, DDGZ22] achieve statistical zero-knowledge due to the underlying structure known as the SIDH square. The SIDH square makes it difficult for the zero-knowledge simulator to generate a transcript that is guaranteed to be valid. Therefore, all these schemes require ad-hoc computational assumptions to argue that they are zero-knowledge. Recently, a state-of-the-art work [BCC$^+$23] resolved this problem by proving the Ramanujan property of the underlying isogeny graph (with Borel level structure). Put simply, the rapid mixing property means that after walking a long enough path in the graph randomly, we'll have an uniform distribution. The property guarantees that the scheme is zero-knowledge. The scheme requires proof sizes of 191 to 662 KB and prover time of 18 to 162 seconds on a single-threaded normal machine for a long[2] isogeny walk.

Zero-knowledge isogeny identification schemes have practical applications beyond key exchange and signatures. For example, in isogeny-based cryptography, a crucial but notorious problem is sampling a supersingular curve without knowledge of its endomorphism ring [BBD$^+$22, MMP22], which is essential in some constructions [LGD21, BD21, AEK$^+$22] and significantly reduces the information available to attackers [PL17, dQKL$^+$21, FMRV22]. To address this, a recent proposal [BCC$^+$23] suggests a trusted setup ceremony in which each participant computes an isogeny path from the curve generated by the previous participant and the initial curve can be any curve (e.g. j-invariant 1728). Then, the participant generates a

[1]One of the curves is required to have a special extremal order (e.g. j-invariant 0 or 1728).

[2]The length of the walk is roughly 3-4 times longer than necessary for the applications demonstrated.

proof of knowledge of the isogeny path between the newly generated curve and the previous one, passes the new curve to the next participant, makes the proof and the two curves public, and disposes of the path. Meanwhile, every participant verifies the proof. As long as *one* party honestly disposes of the path, it is difficult to recover the final curve's endomorphism ring, even if the other participants collude. The scheme provides a practical way to obtain a public curve with an unknown endomorphism ring and therefore reduces the risk of cryptanalysis. Concretely, a medium-scale trusted setup ceremony with 300 participants using single-threaded normal machines would take more than 1.5 hours for $\lambda = 128$ and 13.5 hours for $\lambda = 256$ to complete, using the proposed identification scheme.

Can we do better?

Generic proof systems allow a prover to prove or argue the knowledge of any NP relation. zkSNARK[3], as one of them, allows a prover to prove the validity of a solution to a given NP language. On top of that, zkSNARK enables a prover to produce a publicly-verifiable proof in a zero-knowledge and non-interactive manner. Moreover, the proof size is succinct (sublinear) and the verification time is much shorter than producing the proof. The area of zero-knowledge proof systems has been very active these years [IKOS09, BCC+16b, AHIV17, KKW18, BCR+19, BFH+20, DOT21] (see [Tha20, Ish20] for surveys). These generic proof systems are very useful in cryptography and blockchain studies. They also work well with symmetric primitives and some post-quantum branches [ZCD+20, GMNO18, DDOS19, BDK+21, FJR22, FMRV22].

In contrast, the applications of these proof systems to isogeny cryptography appear to be elusive. Though there exists a VDF from isogenies using SNARG[4] [CSRT22], the result remains theoretical and it is not easy to evaluate the performance. Indeed, due to the complexity of computing isogenies and the size and structure of the field, building or using generic proof systems in isogeny cryptography is always challenging. There is widespread uncertainty in the isogeny community regarding the plausibility of the answer to the following question:

Can generic proof systems serve as practical tools in isogeny cryptography?

8.2 Preliminaries

8.2.1 zkSNARKs

We give a brief description of zkSNARKs. The definition is an adaptation of the interactive-oracle-proof (IOP) framework using the BCS transformation [BCS16]. The BCS transformation, to turn an IOP into a noninteractive proof, differs from the Fiat-Shamir transformation in that it requires an explicitly programmable random oracle model, which is stronger random oracle model. For a more detailed explanation, please refer to [BCS16]. Throughout this work, we

[3]zero-knowledge succinct, non-interactive, argument of knowledge

[4]succinct, non-interactive argument without zero-knowledgeness guarantee.

only consider *transparent* zkSNARKs, which require no trusted setup and can serve as a more practical tool for the applications. Hence, we will not include the description of a common reference string and the trapdoor in the definition.

Definition 8.2.1 (Zero-Knowledge Succinct Non-interactive Argument of Knowledge (zk-SNARK))**.** *Given $\lambda \in \mathbb{N}$, $s : \{0,1\}^* \to [0,1]$, a non-interactive random-oracle proof system for a relation R with is a tuple $(\mathsf{Prove}^{\mathcal{O}}, \mathsf{Verify}^{\mathcal{O}})$ such that:*

1. *(Completeness.) For every $(\mathsf{st}, \mathsf{wt}) \in R$,*

$$\Pr[\mathsf{Verify}^{\mathcal{O}}(\mathsf{st}, \pi) = 1 \mid \pi \leftarrow \mathsf{Prove}^{\mathcal{O}}(\mathsf{st}, \mathsf{wt})] = 1$$

2. *(Soundness.) For every PPT $\mathcal{A}^{\mathcal{O}}$, we have the probability*

$$\Pr\left[\mathsf{Verify}^{\mathcal{O}}(\mathsf{st}, \pi) = \top \ \wedge \ \mathsf{st} \notin \mathcal{R} \;\middle|\; (\mathsf{st}, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}\right]$$

   to be negligible.

3. *(Argument of Knowledge.) For every PPT adversary $\mathcal{A}^{\mathcal{O}}$, st, there exists a PPT extractor $\mathsf{Extract}$ such that the probability*

$$\left|\Pr\left[(\mathsf{st}, \mathsf{wt}) \in R \;\middle|\; \mathsf{wt} \leftarrow \mathsf{Extract}^{\mathcal{A}}(\mathsf{st})\right] - \Pr\left[\mathsf{Verify}^{\mathcal{O}}(\mathsf{st}, \pi) = \top \;\middle|\; \pi \leftarrow \mathcal{A}^{\mathcal{O}}\right]\right|$$

   is be negligible where $\mathsf{Extract}$ has access to the entire execution and the random coins of $\mathcal{A}$ and (st, π) is returned by $\mathcal{A}$.

4. *(Statistical Zero-Knowledge.) There exists a PPT simulator Sim (in the explicitly programmable random oracle model) such that for any unbounded adversary $\mathcal{A}$, $(\mathsf{st}, \mathsf{wt}) \in R$ and the list L the probability*

$$\left|\Pr\left[\mathcal{A}^{\mathcal{O}|_L}(\pi) \to 1 \;\middle|\; \pi \leftarrow \mathsf{Sim}^{\mathcal{O}}(\mathsf{st})\right] - \Pr\left[\mathcal{A}^{\mathcal{O}}(\pi) \to 1 \;\middle|\; \pi \leftarrow \mathsf{Prove}^{\mathcal{O}}((\mathsf{st}, \mathsf{wt}))\right]\right|$$

   is be negligible where L is a list and the restricted $\mathcal{O}|_L$ on input x returns y if $(y, x) \in L$ or $\mathcal{O}(x)$ otherwise.

5. *(Succinct.) For any $(\mathsf{st}, \mathsf{wt}) \in R$ the size of the proof produced by Prove grows polylogarithmically in the size of wt. Concretely, $|\pi| = \mathsf{poly}(\lambda, |\mathsf{st}|, \log(|\mathsf{wt}|))$.*

8.2.2 Isogeny Problem and Isogeny Path

We recall the isogeny problem, which is commonly used in isogeny-based constructions with the additional restriction of smooth degree isogenies. This restriction is used to make isogeny evaluation more efficient through the use of Vélu formulas. Thus, we can define the restricted version of the original isogeny problem (Prob. 1) as follows.

Problem 11. *Given $k \in \mathbb{N}$, a prime number ℓ and two isogenous supersingular E, E', the task is to find an isogeny $\phi : E_1 \to E_2$ of degree ℓ^k.*

Equivalently, by Thm. 2.1.10, the problem above is equivalent to find a vector (witness) over $\mathbb{F}_{p^2}^{k+1}$ for the statement (E, E') for the relation

$$R_{\ell^k\text{-}\mathsf{Iso}} = \left\{ \left(\mathsf{st} = (E, E'), \mathsf{wt} = (j_0, \cdots, j_k)\right) \middle| \begin{array}{rl} j(E) & = j_0,\ j(E') = j_k, \\ \Phi_\ell(j_{i-1}, j_i) & = 0 \text{ for all } i \in [k] \end{array} \right\}. \tag{8.1}$$

Using zkSNARKs, we can construct an argument system for the relation $R_{\ell^k\text{-}\mathsf{Iso}}$, which in turn provides an identification scheme for isogeny path knowledge. In this work, we demonstrate the efficacy of our method for $\ell = 2$, the power of which is the most common choice of the torsion subgroup size used in isogeny constructions.

Remark 8.2.2. *An argument system for Prob. 11 can be constructed using the Vélu formulas. Specifically, an arithmetic circuit that takes a kernel and E as input, and outputs E' can be used with a generic proof system to create an identification scheme for the isogeny knowledge. However, this approach is inefficient due to the complexity of the Vélu formulas and requires roughly 20 times more gates compared to our method that uses the modular polynomial.*

8.2.3 Rank-1 Constraint System

We recall the definition of rank-1 constraint systems (R1CS), which some zk-SNARKs (e.g. Aurora) take as an input. An R1CS is parameterized by $n, m \in \mathbb{N}$ and a prime power q, and consists of instance-witness pairs $((A, B, C, v), w)$ where $A, B, C \in \mathbb{F}_q^{m\times(n+1)}$, the vectors v,w are over $\mathbb{F}_q$ with $\dim(v) + \dim(w) = n$ such that

$$Az \circ Bz = Cz$$

for $z := (1, v, w) \in \mathbb{F}_q^{n+1}$, where $\circ$ denotes coordinate-wise (Hadamard) product. Conceptually, A, B, C encode constraints on variables v, w which represent public variables and private variables, respectively. The private variable w could contain intermediate variables. Looking ahead, (A, B, C, v) will serve as the statement and the w (or $z = (1, v, w)$) will be the witness.

One way to encode arithmetic circuit satisfiability using R1CS is to encode every addition and multiplication as a new constraint. However, this approach results in impractical results. In Section 8.3.1, we present an alternative method for encoding the isogeny path relation Eq. (8.1) into R1CS with optimizations.

8.3 Construction

In this section, we introduce an efficient method to encode the isogeny relation Eq. (8.1) into an R1CS representation. We start by encoding it over the quadratic field $\mathbb{F}_{p^2}$ in Sec. 8.3.1 and then demonstrate how to lift it to $\mathbb{F}_p$ in an economical manner in Sec. 8.3.2.

8.3.1 Optimization for R1CS over $\mathbb{F}_{p^2}$

Let $R_{\ell^k\text{-Iso}}$ be instantiated by setting $k = 1$, $\ell = 2$, we obtain the base R1CS matrices:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}, B = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_4 & c_4 & 0 & 0 & 0 & 0 & -1 \end{bmatrix},$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ c_0 & c_1 & c_1 & c_2 & c_2 & 1 & 1 & c_3, \end{bmatrix}$$

where

$$c_0 = -157464000000000 \qquad c_1 = 8748000000 \qquad c_2 = -162000$$
$$c_3 = 40773375 \qquad c_4 = 1488,$$

where the c_i's are coefficients from Φ_2 in Sec. 2.1. The first 5 rows represent the constraints of including squaring (j_0^2, j_1^2), cubing (j_0^3, j_1^3), and the multiplication $(j_0 j_1)$ respectively. The last row represents the constraint

$$(-j_0 j_1)(c_4 j_0 + c_4 j_1 - j_0 j_1) = j_0^3 + j_1^3 + c_2(j_0^2 + j_1^2) + c_1(j_0 + j_1) + c_3(j_0 j_1) + c_0. \tag{8.2}$$

We can extend this method to a path of length $k > 1$ by introducing an additional 4 variables (including input) for each j-invariant j_i: j_i, j_i^2, j_i^3, and $j_{i-1}j_i$. It is important to note that the squaring and cubing constraints for each j_{i-1} can be reused. Therefore, entire R1CS representation of $R_{k,\ell}$ requires $n = 4k + 3$ variables and $m = 4k + 2$ constraints over $\mathbb{F}_{p^2}$.

8.3.2 Optimization for Lifting to $\mathbb{F}_p \times \mathbb{F}_p$

This subsection presents several techniques to reduce the overhead to lift arithmetic over a quadratic field to the underlying prime field. We consider a quadratic field $\mathbb{F}_{p^2} \cong \mathbb{F}_p[\alpha]$ where $\alpha^2 = d$ for some quadratic non-residue $d \in \mathbb{F}_p$.

The motivation is that, generally, the j-invariant of an elliptic curve is taken over $\mathbb{F}_{p^2}$ but some proof systems only support arithmetic over a prime field. Indeed, arithmetic computation over $\mathbb{F}_p[\alpha]$ can be viewed as arithmetic computations over an $\mathbb{F}_p$-vector space natively. That is, for $x_1, x_2, y_1, y_2 \in \mathbb{F}_p$ to represent $x_1 + x_2\alpha \in \mathbb{F}_p[\alpha]$, by mapping $x_1 + x_2\alpha$ to (x_1, x_2) the addition is (x_1+y_1, x_2+y_2) and the multiplication is $(x_1y_1+x_2y_2d, x_1y_2+x_2y_1)$. This method results in 4 (variable) $\mathbb{F}_p$-multiplications for one (variable) $\mathbb{F}_{p^2}$-multiplication (i.e. $x_1x_2, y_1y_2, x_1y_2, x_2y_1$). In fact, with a few following tricks (arithmetic folklores), this can be done more efficiently:

Arithmetic. Observe that by letting $u_1 = x_1y_1$, $u_2 = y_2y_2$, and $u_3 = (x_1+x_2)(y_1+y_2)$, we have

- $x_1y_1 + x_2y_2d = u_1 + u_2d$
- $(x_1y_2 + x_2y_1) = u_3 - u_1 - u_2$.

By using the trick, it only requires 3 (variable) $\mathbb{F}_p$-multiplications now. The saving depends on the proof system to be used. In many proof systems, it is much more expensive to verifiy a (variable) multiplication relation than a (variable) linear relation.

Next, we consider variable squaring for $x + y\alpha \in \mathbb{F}_p[\alpha]$ by letting $u_1 = xy$ and $u_2 = (x+y)(x+yd)$. Then, we have

- $(x^2 + y^2\alpha^2) = u_2 - (d+1)u_1$
- $2xy = 2u_1$.

Therefore, it only requires 2 (variable) $\mathbb{F}_p$-multiplications with this trick.

Application to R1CS Matrices. Now we can apply the abovementioned techniques to our R1CS matrices. Recall that in Sec. 8.3.1, we have a witness vector z over $\mathbb{F}_p \times \mathbb{F}_{p^2}^7$. As an abuse of notation, given an element $x := a + b\alpha \in \mathbb{F}_p[\alpha]$, we may denote a as $\mathsf{Re}(x)$ and b as $\mathsf{Im}(x)$. To lift the witness vector z from $\mathbb{F}_p \times \mathbb{F}_{p^2}^7$ to $\mathbb{F}_p$, we embed it naturally into $\mathbb{F}_p \times \mathbb{F}_p^{14}$. We can then build a submatrix for each constraint and introduce intermediate variables as follows:

Square Constraint. Regarding the square constraint, take the subvector $(1, \mathsf{Re}(x), \mathsf{Im}(x), \mathsf{Re}(x^2), \mathsf{Im}(x^2))$ for instance, the corresponding submatrices for this constraint are respectively

$$\begin{bmatrix} 0 & 2 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & d & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 2^{-1}(d+1) \end{bmatrix},$$

where two row represent $2\mathsf{Re}(x)\mathsf{Im}(x) = \mathsf{Im}(x^2)$ and $(\mathsf{Re}(x) + \mathsf{Im}(x))(\mathsf{Re}(x) + d\mathsf{Im}(x)) = \mathsf{Re}(x^2) + 2^{-1}(d+1)\mathsf{Im}(x^2)$, respectively.

Multiplication Constraint. For multiplication relation, we need an additional variable u over $\mathbb{F}_p$. We take the subvector $(1, \mathsf{Re}(x), \mathsf{Im}(x), \mathsf{Re}(y), \mathsf{Im}(y), u, \mathsf{Re}(xy), \mathsf{Im}(xy))$ for instance. The corresponding submatrices for this constraint are respectively

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -d & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1-d & 1 & 1 \end{bmatrix},$$

the three rows represent $\mathsf{Im}(x)\mathsf{Im}(y) = u$, $\mathsf{Re}(x)\mathsf{Re}(y) = \mathsf{Re}(xy) - ud$, and $(\mathsf{Re}(x)+\mathsf{Im}(x))(\mathsf{Re}(y)+\mathsf{Im}(y)) = \mathsf{Im}(xy) + \mathsf{Re}(x)\mathsf{Re}(y) + u$, respectively.

Constraint Eq. (8.2). We can apply our multiplication technique above to the constraint Eq. (8.2). Recall that the final constraint from the modular polynomial is $(-xy)(c_4x+c_4y-xy) = x^3 + y^3 + c_2(x^2 + y^2) + c_1(x + y) + c_3xy + c_0$. The insight is every coefficient c_i is over $\mathbb{F}_p$ so $\mathsf{Re}(\cdot)$ has the linear property $\mathsf{Re}(c_4x + c_4y - xy) = c_4\mathsf{Re}(x) + c_4\mathsf{Re}(y) - \mathsf{Re}(xy)$ and so does the imaginary part $\mathsf{Im}(\cdot)$. Therefore, we can use three constraints for the real part and the imaginary part of $x^3 + y^3 + c_2(x^2 + y^2) + c_1(x + y) + c_3xy + c_0$ in terms of $\mathsf{Re}(X), \mathsf{Im}(X), \mathsf{Re}(Y), \mathsf{Im}(Y)$ where $X = -xy$ and $Y = c_4x + c_4y - xy$ as the method described above.

Concretely, for a subvector

$$z' = \begin{pmatrix} 1 & \mathsf{Re}(x) & \mathsf{Im}(x) & \mathsf{Re}(y) & \mathsf{Im}(y) & \mathsf{Re}(x^2) & \mathsf{Im}(x^2) & \mathsf{Re}(y^2) & \mathsf{Im}(y^2) & \mathsf{Re}(x^3) & \mathsf{Im}(x^3) & \mathsf{Re}(y^3) & \mathsf{Im}(y^3) & \mathsf{Re}(xy) & \mathsf{Im}(xy) & u \end{pmatrix}^T$$

the corresponding submatrices for this constraint are respectively

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \end{bmatrix},$$

$$\begin{bmatrix} 0 & 0 & c_4 & 0 & c_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & c_4 & 0 & c_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & c_4 & c_4 & c_4 & c_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \end{bmatrix},$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ c_0 & c_1 & 0 & c_1 & 0 & c_2 & 0 & c_2 & 0 & 1 & 0 & 1 & 0 & c_3 & 0 & -d \\ c_0 & c_1 & c_1 & c_1 & c_1 & c_2 & c_2 & c_2 & c_2 & 1 & 1 & 1 & 1 & c_3 & c_3 & (1-d) \end{bmatrix},$$

which respectively represent

$$\begin{aligned} \mathsf{Im}(X)\mathsf{Im}(Y) &= u \\ \mathsf{Re}(X)\mathsf{Re}(Y) &= \mathsf{Re}(Z) - ud \\ (\mathsf{Re}(X) + \mathsf{Im}(X))(\mathsf{Re}(Y) + \mathsf{Im}(Y)) &= \mathsf{Im}(Z) + \mathsf{Re}(Z) + (1-d)u, \end{aligned}$$

where

$$\begin{aligned} X &= -xy \\ Y &= c_4x + c_4y - xy \\ Z &= x^3 + y^3 + c_2(x^2 + y^2) + c_1(x + y) + c_3xy + c_0. \end{aligned}$$

In summary, we can transform any isogeny path over a quadratic field $\mathbb{F}_{p^2}$ of length k into an R1CS relation over $\mathbb{F}_p$ with $11k + 3$ variables and $11k + 2$ constraints. This is a substantial improvement over the naive approach, which would result in $16k + 3$ variables and $16k + 2$ constraints.

8.4 Implemention

We implemented our scheme using Aurora [BCR+19] and Ligero [AHIV17] on a single-threaded Intel® Core™ i9-9900 CPU @ 3.10GHz. These proof systems are zkSNARKs that achieve post-quantum and statistical zero-knowledge without requiring a trusted setup. In contrast, Virgo and Orion [ZXZS20, XZS22] also have these properties, but they rely on arithmetic circuits on fields generated by specific Mersenne primes for performance improvements, which is not compatible with our setting.

The implementation is through a fork of `libiop`[5], modified to support larger prime fields or quadratic field extensions. While Ligero's original implementation is closed source, an adaptation is included in `libiop` library interfaced with the R1CS language.

8.4.1 Identification Scheme

Our proof system may also serve as an identification scheme to validate a public key (E, E'), where the prover can use our zkSNARK construction to demonstrate their knowledge of a walk from E to E', of sufficient length to resist the most efficient generic algorithm for recovering the secret isogeny (i.e. the claw finding algorithm) [BJS14]. In this section, we demonstrate the effectiveness of our proof system in this regard.

We show in Tab. 8.1 the R1CS parameter set (m, n) over $\mathbb{F}_p$ (lifted from $\mathbb{F}_{p^2}$ as described in Sec. 8.3.2) and the isogeny walk length k with respect to the security parameter λ. A concrete result is given in Tab. 8.2 regarding prover time, verifier time and the proof size for different forms of the primes.

p	k	n	m	Strength
$p_{434} = 2^{216}3^{137} - 1$	216	2380	2379	NIST 1
$p_{503} = 2^{250}3^{159} - 1$	250	2754	2753	NIST 2
$p_{610} = 2^{305}3^{192} - 1$	305	3359	3358	NIST 3
$p_{751} = 2^{372}3^{239} - 1$	372	4096	4095	NIST 5

Table 8.1: Our R1CS parameter set (m, n) over $\mathbb{F}_p$ and the isogeny walk length k as described in Secs. 8.3.1 and 8.3.2 with respect to the security parameter λ and the prime p.

[5]Original source code available at https://github.com/scipr-lab/libiop. Our fork can be found at https://github.com/levanin/libiop-other-primes.

Prime		Our Work: Aurora	Our Work: Ligero
$p_{434} = 2^{216}3^{137} - 1$	Prove	934ms	587ms
	Verify	99ms	847ms
	$\lvert\pi\rvert$	194kB	1,849kB
$p_{503} = 2^{250}3^{159} - 1$	Prove	1,138ms	686ms
	Verify	114ms	959ms
	$\lvert\pi\rvert$	219kB	2,127kB
$p_{610} = 2^{305}3^{192} - 1$	Prove	3,175ms	2,488ms
	Verify	472ms	2,614ms
	$\lvert\pi\rvert$	517kB	4,084kB
$p_{751} = 2^{372}3^{239} - 1$	Prove	3,882ms	1,951ms
	Verify	824ms	6,407ms
	$\lvert\pi\rvert$	828kB	6,394kB

Table 8.2: Table of results comparing several generic proof systems operating for the R1CS instantiation of $R_{2^k\text{-Iso}}$ without relaxations. The parameter is set according to Tab. 8.1 and Prove, Verify, $|\pi|$ correspond to the prover time, verification time, and proof size respectively. Results displayed are for single-threaded performance.

8.4.2 Application to the Isogeny SECUER Project

To generate a supersingular curve with an unknown endomorphism ring, a recent proposal, called SECUER project, [BCC+23] suggests a trusted setup ceremony in which each participant computes an isogeny path from the curve generated by the previous participant and the initial curve can be any curve (e.g. j-invariant 1728). Then, the participant generates a proof of knowledge of the isogeny path between the newly generated curve and the previous one, passes the new curve to the next participant, makes the proof and the two curves public, and disposes of the path. Meanwhile, every participant verifies the proof. As long as *one* party honestly disposes of the path, it is difficult to recover the final curve's endomorphism ring, even if the other participants collude. The scheme provides a practical way to obtain a public curve with an unknown endomorphism ring, reducing the risk of cryptanalysis.

The proposal in [BCC+23] presents a state-of-the-art isogeny NIZK based on a sigma protocol. Our scheme applies to this application using the R1CS parameter set presented in Tab. 8.3. A comparison of the performance of our scheme and that of [BCC+23] is presented in Tab. 8.4 in terms of prover time, verification time, and proof size. When instantiated with Aurora, our proof system provides 3 to 10 times faster prover time and verification time than those of [BCC+23] with similar proof sizes. Both proof systems provide statistical zero-knowledge security but we note that our proof system requires a stronger random oracle model (EPRO) due to the BCS transform. Notably, our scheme provides a proof of knowledge for the exact relation $R_{2^k\text{-Iso}}$, whereas the scheme in [BCC+23] requires a relaxed relation $R_{2^t\text{-Iso}}$ for some $k \leq t \leq 2k$.

p	k	n	m	Strength
$p_{434} = 2^{216}3^{137} - 1$	705	7759	7758	NIST 1
$p_{503} = 2^{250}3^{159} - 1$	774	8518	8517	NIST 2
$p_{610} = 2^{305}3^{192} - 1$	1010	11114	11113	NIST 3
$p_{751} = 2^{372}3^{239} - 1$	1280	14084	14083	NIST 5

Table 8.3: Our R1CS parameter set (m, n) over $\mathbb{F}_p$ and the isogeny walk length k with respect to the security parameter λ and the prime p.

8.5 Discussion

Our result represents a meaningful progress in the application of generic proof systems to isogeny cryptography, and it is comparable to the state-of-the-art work in the isogeny literature. Several potential improvements or research directions can be explored to further enhance the result in this direction. Firstly, more advanced proof systems such as Ligero++ [BFH+20] or the updated version of Ligero [AHIV22] could be used to improve the prover time, verification time, and the proof size. Additionally, one can construct an identification scheme for an alternative form of the isogeny problem using different proof systems based on, for example, sumcheck or supporting batch argument ([CJJ22, WW22]). Furthermore, some special functions of a proof system can be beneficial for certain applications. For instance, a proof system supporting recursive proof or proof aggregation (e.g., [ACL+22]) is beneficial to the SECUER project by enabling the public to verify the entire trusted setup ceremony much faster than verifying each proof in the ceremony individually. In summary, we hope that our work serves as a bridge between the isogeny community and the zkSNARK community and encourages further research and applications in this direction.

Prime		Our Work: Aurora	Our Work: Ligero	[BCC+23]: SECEUR PoK
$p_{434} = 2^{216}3^{137} - 1$	Prove	4,204ms	1,479ms	12,369ms
	Verify	378ms	1,899ms	1,399ms
	$\|\pi\|$	277kB	3,281kB	191kB
$p_{503} = 2^{250}3^{159} - 1$	Prove	4,944ms	1,722ms	19,296ms
	Verify	440ms	2,171ms	2,173ms
	$\|\pi\|$	313kB	3,778kB	216kB
$p_{610} = 2^{305}3^{192} - 1$	Prove	6,457ms	3,331ms	60,915ms
	Verify	888ms	3,102ms	6,646ms
	$\|\pi\|$	570kB	4,568kB	404kB
$p_{751} = 2^{372}3^{239} - 1$	Prove	12,555ms	5,243ms	141,043ms
	Verify	1651ms	13,509ms	15,931ms
	$\|\pi\|$	688kB	11,302kB	663kB

Table 8.4: Table of results comparing several generic proof systems operating over $\mathbb{F}_{p^2}$ for the R1CS instantiation of the relation $R_{2^k\text{-Iso}}$, and the isogeny SECEUR PoK in [BCC+23]. Security level and walk length is set according to Tab. 8.3 and Prove, Verify, $|\pi|$ correspond to proof time, verification time, and proof size respectively. Results displayed are for single-threaded performance.

Chapter 9

Conclusion

In this thesis, we demonstrate the versatility of isogeny cryptography in various advanced cryptosystems and highlight several potential topics for further research, which can be summarized as follows:

1. We explore the folklore method of finding larger known-order effective group actions using quantum algorithms and lattice reduction techniques. We provide lower bound estimations for precomputation and action evaluation costs and show there is still room for improvement. Further research is needed to enhance the method for instantiating isogeny-based known-order effective group actions.

2. We present the first efficient UC-secure isogeny-based oblivious transfer (OT) in the isogeny literature. We employ several reductions to demonstrate that its security relies on the standard group action inverse problem (GAIP). The development of an efficient round-optimal or adaptively UC-secure OT from isogenies remains an intriguing topic.

3. We introduce the first post-quantum accountable ring signature, which immediately implies a dynamic group signature and the first such construction in the isogeny literature. Additionally, we provide a tightly secure variant for these schemes, a rare feature in the post-quantum ring/group signature literature. We highlight the potential topics of constructing secure and efficient schemes in QROM and improving proof size by expanding the challenge space in each repetition.

4. We present the first provably secure blind signature in the isogeny literature with competitive performance in the post-quantum domain. We also explore the fascinating cyclotomic structure of the ideal class group and propose ring-GAIP to expand the challenge space and improve signature size.

5. We introduce the first provably secure verifiable random functions from isogenies with competitive performance in the post-quantum literature by constructing the proof system for the action factorization relation. Notably, the security is based on the standard DDH problem. To prove it, we propose a generalized DDH problem and show the equivalence, which could serve as a useful tool for other applications.

6. We present the first practical applications of generic proof systems to isogeny cryptography, showcasing competitive performance compared to state-of-the-art results in the isogeny literature for the identification scheme application. There are several potential improvements and interesting directions to enhance the application of generic proof systems to isogeny cryptography, such as utilizing more advanced proof systems, constructing identification schemes for alternative forms of the isogeny problem, and exploring specialized functions of proof systems for specific applications.

In conclusion, this thesis represents substantial progress in isogeny cryptography by exploring advanced constructions that are crucial for real-world applications. We hope to contribute to the growing body of knowledge within isogeny cryptography by offering an in-depth analysis of these cryptosystems, developing novel tools and methods and highlighting potential avenues for future research. This thesis aims to be a helpful guide for beginners, offering a clear understanding of isogeny-based cryptosystems and their uses. We hope to inspire more research and growth in this new area of post-quantum cryptography.

Appendix A

Oblivious Transfer

A.1 Equivalence of The Square/Inverse/Reciprocal CSIDH Problem and The Computational CSIDH Problem

We will show the computational CSIDH problem is equivalent to the square variant with a quantum reduction. The order of the ideal class group can be computed with a quantum algorithm [Sho99, Hal02]. This is the only part of the reduction that is quantum and it can be viewed as a precomputation; the rest of the reduction is classical. Proposition A.1.1 shows the equivalence for the case that the order of the class group of the endomorphism ring is odd which is the case when $p = 3 \mod 4$. The remaining case is that the class number is even which happens when $p = 1 \mod 4$. In this case, the discriminant is $-4p$.

Proposition A.1.1. *([Fel19]) The square CSIDH problem is equivalent to the computational CSIDH problem if the order h of the group Cl is odd and given.*

Lemma A.1.2. *Given $(E, a * E)$. Then for $n \in \mathbb{N}$ one can compute $a^n * E$ with given access to the oracle $\mathcal{O}$ for the square CSIDH problem with $\mathcal{O}(\log(n))$ queries.*

Proof. The reduction is based on the double-and-add method. Firstly, generate $a^{-1} * E = \mathcal{O}(a * E, E)$. Given $a^i * E$, by quering $\mathcal{O}(E, a^i E)$ and $\mathcal{O}(a^{-1} * E, a^i * E)$, one can compute $a^{2i} * E$ and $a^{2i+1} * E$, respectively. Therefore, we can compute $a^n * E$ within $\log(n) + 3$ oracle queries. $\square$

Note that the direct isogeny computation of a^e acting on E where a, E are given and $e = \Theta(\log(\#Cl))$ is avoided in the following proof, since these types of isogeny computations may not be polynomial-time [CLM+18, BFJ16], which would make the reduction non-polynomial-time. Let $(x)_y^{-1}$ denote the inverse of $x \mod y$.

Proposition A.1.3. *The square CSIDH problem is equivalent to the computational CSIDH problem if the order h of the group Cl is given.*

Proof. Let $p = 1 \mod 4$ so that the order h of the class group is even. Since the discriminant of the class group is $-4p$, by Proposition 3.11 of [Cox11], the 2-Sylow subgroup of the class group is of rank 1. Hence, the class group is isomorphic to $\mathbb{Z}_{2^t} \times \mathbb{Z}_{h'}$ for some $t, h' \in \mathbb{N}$ with h' being odd. Given the challenge $(E, a * E, b * E)$ and access to the oracle $\mathcal{O}$, the goal is to find the curve $ab * E$.

Define the mapping $\chi : Cl \to Cl \times Cl$ where $\chi_{2^t}(\mathfrak{a}) = \mathfrak{a}^{h'(h')^{-1}_{2^t}}$, $\chi_{h'}(\mathfrak{a}) = \mathfrak{a}^{2^t(2^t)^{-1}_{h'}}$ and $\chi(\mathfrak{a}) = (\chi_{2^t}(\mathfrak{a}), \chi_{h'}(\mathfrak{a}))$. The image of χ is isomorphic to $\mathbb{Z}_{2^t} \times \mathbb{Z}_{h'}$. The mapping χ satisfies $\chi_{2^t}(x)\chi_{h'}(x) = x$. Given $\mathfrak{a} * E$, the pair $(\chi_{2^t}(\mathfrak{a}) * E, \chi_{h'}(\mathfrak{a}) * E)$ can be efficiently computed by Lemma A.1.2 by using the oracle.

Run the following polynomial-time algorithm.

1. Compute $a'' * E$ and $b'' * E$ where $a'' = \chi_{h'}(a)$ and $b'' = \chi_{h'}(b)$ by Lemma A.1.2.
2. From Proposition A.1.1, given $a'' * E, b'' * E$, one can get $a''b'' * E$ since a'', b'' are of odd order.
3. Compute $(a''b'')^{\frac{h'+1}{2}} * E$ using Lemma A.1.2. Denote $q = (a''b'')^{\frac{h'+1}{2}} \in Cl$. Note that $q^2 = a''b''$.
4. Generate a curve $g * E$ where $g \in Cl$ is of order 2^t. (See details below.)
5. Compute $a' * E$ and $b' * E$ where $a' = \chi_{2^t}(a)$ and $b' = \chi_{2^t}(b)$. Write $a' = g^{\sum a'_i 2^i}$ and $b' = g^{\sum b'_i 2^i}$ where $a'_i, b'_i \in \{0, 1\}$.
6. Obtain a'_0 by computing $a'^{2^{t'-1}} * E$ by Lemma A.1.2. If it is E, then $a'_0 = 0$. Otherwise, $a'_0 = 1$.
7. Iteratively, for $j < t - 1$, assume $a'_0, ..., a'_{j-1}$ are known, then obtain a'_j by computing $a'^{2^{t-j-1}} * E$. If, with Lemma A.1.2, the curve equals
$$g^{\sum_0^{j-1} 2^{i+t-j-1} a'_i} * E,$$
then $a'_j = 0$; otherwise, $a'_j = 1$.
8. Repeat Step 6 and Step 7 for b' to obtain $b'_i \in \{0, 1\}$.
9. Compute $g^{-\sum(a'_i+b'_i)2^i} * E$ using Lemma A.1.2. (Note that $\chi_{2^t}((ab)^{-1}) * E = g^{-\sum(a'_i+b'_i)2^i} * E$.)
10. Compute $\mathcal{O}(g^{-\sum(a'_i+b'_i)2^i} * E, q * E)$ to obtain $ab * E$.

In Step 4, the curve can be generated by sampling a random element $g_{pre} \in Cl$ and raising g_{pre} to the power of $2^{t-1}h'$. If it is the identity element in Cl, then restart. Otherwise, set $g * E$ to be $g_{pre}^{h'} * E$ by Lemma A.1.2. If the sampling is random enough, then the success rate is 1/2 for each trial.

The relation between $a' * E$ and $g * E$ is computed in Step 6. Since g is of order 2^t, then $a'^{2^{t-1}} * E = g^{a'_0 2^{t-1}} * E$. Hence, a'_0 is 0 if and only if the outcome is E.

In Step 7, the idea of Step 6 is taken one step further to recover a'_j for $j = 1, ..., t-1$ iteratively. This idea is known as the Pohlig-Hellman attack [PH78]. If $a'_0, ..., a'_{j-1}$ are known, raising $a' * E$ to the power of $t-j-1$ eliminates $a_{j+1}, ..., a_{t-1}$ in the exponentiation of a' with the base g, since the order of g is 2^t. We can thereby find out a'_j through comparing. To be more specific, due to

$$a'^{2^{t-j-1}} * E = g^{\sum_0^j a'_i 2^{i+t-j-1}} * E,$$

we have $a'_j = 0$ if and only if

$$g^{\sum_0^{j-1} 2^{i+t-j-1} a'_i} * E = g^{\sum_0^j 2^{i+t-j-1} a'_i} * E.$$

The same reasoning holds for b. Hence, we can compute $g^{-\sum (a'_i + b'_i) 2^i} * E$, which is $\chi_{2^t}((ab)^{-1}) * E$.

In step 10, we invoke the oracle of the square CSIDH problem and get

$$\begin{aligned} & \mathcal{O}\left(g^{-\sum(a'_i+b'_i)2^i} * E, q * E\right) \\ = & \left(g^{\sum(a'_i+b'_i)2^i} q^2\right) * E \\ = & \left(\chi_{2^t}(ab)(a''b'')^{h'+1}\right) * E \\ = & \left(\chi_{2^t}(ab)(a''b'')\right) * E \\ = & \left(\chi_{2^t}(ab)\chi_{h'}(ab)\right) * E \\ = & \, ab * E. \end{aligned}$$

□

With the reduction in the context (Proposition 4.2.2), we have shown equivalence between square, inverse, reciprocal variants. Therefore, in a generic CSIDH setting, we have the following relation

Computational CSIDH $=_{quantum}$ **Computational Inverse/Square CSIDH**.

A.2 The Hardness of Tweaked Reciprocal CSIDH Problem

Recall the definition of the tweaked reciprocal CSIDH prolem.

Problem. ***(Tweaked Reciprocal CSIDH Problem)*** *Given E in $\mathcal{E}$. The adversary $\mathcal{A}$ chooses and commits to a curve $X \in \mathcal{E}$.*

(i) $\mathcal{A}$ receives the first challenge $s \star E$ where $s \xleftarrow{\$} Cl$ from the challenger. $\mathcal{A}$ outputs a curve C.

(ii) The challenger sends s and another challenge $s' \star E$ to $\mathcal{A}$ where $s' \xleftarrow{\$} Cl$.

(iii) $\mathcal{A}$ outputs another curve C'.

Write $(C_0, C_1) = (s \star X, s^{-1} \star X)$ *and* $(C'_0, C'_1) = (s' \star X, s'^{-1} \star X)$*. We say* $\mathcal{A}$ *wins if* $(C, C') = (C_i, C'_{1-i})$ *for some* $i \in \{0, 1\}$*.*

The advantage of an adversary $\mathcal{A}$ against the tweaked reciprocal CSIDH problem is defined as $\mathsf{Adv}^{\mathsf{tReGA}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}]$ or, to be more specific, $\mathsf{Adv}^{\mathsf{tReGA}}(\mathcal{A}(E; X)) = \Pr[\mathcal{A}$ commits to X with the public curve E and wins$]$. If $\mathcal{A}$ wins, we say $\mathcal{A}$ solves for i for some $i \in \{0, 1\}$.

Proposition A.2.1. *The tweaked reciprocal CSIDH problem is equivalent to the computational inverse CSIDH problem.*

Proof. Given a computational inverse CSIDH problem oracle $\mathcal{O}(\cdot, \cdot)$, a reduction $\mathcal{B}_1$ proceeds as follows. After obtaining E, $\mathcal{B}_1$ commits to $X = E$ and obtains the first challenge $s \star E$. $\mathcal{B}_1$ returns E, ignore the given group element, and obtains the second challenge $s' \star E$. $\mathcal{B}_1$ obtains $s'^{-1} \star E$ by invoking the oracle $\mathcal{O}(E, s' \star E)$. Hence, $\mathcal{B}_1$ wins the experiment.

Given a tweaked CSIDH problem adversary $\mathcal{A}$, we will construct two inverse CSIDH adversaries (reductions) $\mathcal{B}_2, \mathcal{B}_3$ dealing with distinct circumstances. Upon receiving an inverse CSIDH challenge $(E, s \star E)$ to compute $s^{-1} \star E$, both $\mathcal{B}_2, \mathcal{B}_3$ invoke $\mathcal{A}$ with E. Both $\mathcal{B}_2, \mathcal{B}_3$ will rewind $\mathcal{A}$ later on back to the time when it commits to X to extract $s^{-1} \star E$. Say $\mathcal{A}$ solves for i in the first experiment and then, after rewinding, solves for i'. We define an event $\mathbf{E}$ that $i = i'$ and thereby the event $\neg\mathbf{E}$ implies $i = 1 - i'$. We have

$$\begin{aligned}(\mathsf{Adv}^{\mathsf{tReGA}}(\mathcal{A}(E; X)))^2 = \quad & \Pr[\mathcal{A} \text{ commits to } X \text{ and wins} \mid \mathbf{E}] \Pr[\mathbf{E}] \\ & + \Pr[\mathcal{A} \text{ commits to } X \text{ and wins} \mid \neg\mathbf{E}] \Pr[\neg\mathbf{E}].\end{aligned}$$

$\mathcal{B}_2$ continues $\mathcal{A}$ conditioned on $\mathbf{E}$. Firstly, $\mathcal{B}_2$ sends the first challenge $t_1 s \star E$ to the adversary where $t_1 \xleftarrow{\$} Cl$ for randomizing the challenge. After receiving C_i from the $\mathcal{A}$, we rewind $\mathcal{A}$ to the time when it outputs X. Recall that we say $i = 0$ if $C_i = t_1 s \star X$ and $i = 1$ if $C_i = (t_1 s)^{-1} \star X$. $\mathcal{B}_2$ resends $t \star E$ where $t \xleftarrow{\$} Cl$ as a new first challenge to $\mathcal{A}$. Say $\mathcal{A}$ returns $C'_{i'}$. Recall that we say $i' = 0$ if $C'_{i'} = t \star X$ and $i' = 1$ if $C'_{i'} = t^{-1} \star X$. If $\mathcal{A}$ returns $t \star X$ or $t^{-1} \star X$, then $\mathcal{B}_2$ sends t to $\mathcal{A}$ and asserts the target of $\mathcal{A}$ is $i = i' = 0$ or $i = i' = 1$, respectively.

If $i = 0$, then, by using t_1, the reduction $\mathcal{B}_2$ sends $t_2 s \star X$ as the challenge with respect to committed X where $t_2 \xleftarrow{\$} Cl$ for randomizing the challenge. After receiving C'_1 from the adversary, $\mathcal{B}_2$ outputs $t_2 \star C'_1$.

Claim $t_2 \star X_1 = s^{-1} \star E$. Write $X = b \star E$ for some $b \in Cl$ due to the transitive action, so $t_2 s \star X = (t_2 s b) \star E$. Then, since the second challenge is $t_2 s \star X = (t_2 s b) \star E$, we have $t_2 \star X_1 = (sb)^{-1} \star X = s^{-1} \star E$.

If $i = 1$, then, by using t_1, the reduction $\mathcal{B}_2$ sends $((t_2 s)^{-1} \star X)^t$ as the challenge with respect to committed X where $t_2 \xleftarrow{\$} Cl$ for randomizing the challenge. After receiving C'_0 from the adversary, $\mathcal{B}_2$ outputs $t_2 \star (C'_0)^t$.

Claim $t_2 \star (C_0')^t = s^{-1} \star E$. Write $X = b \star E$ and $E = t \star E_0$ for some $b, t \in Cl$ thanks to the transitive action, so $((t_2 s)^{-1} \star X)^t = t_2 s b^{-1} t^{-2} \star E$. Then, we have $C_0' = t_2 s b^{-1} t^{-2} \star X = t_2 s t^{-1} \star E_0 = t_2 s \star E^t$. Therefore, $t_2 \star (C_0')^{=} s^{-1} \star E$.

$\mathcal{B}_3$ continues $\mathcal{A}$ conditioned on $\neg\mathbf{E}$. Firstly, $\mathcal{B}_3$ guesses $i \in \{0, 1\}$. $\mathcal{B}_3$ and sets $i = 1 - i'$. If $i = 0$, then $\mathcal{B}_3$ executes as the reduction in the first approach of the proof in Prop 4.2.2. If $i = 1$, then $\mathcal{B}_3$ executes as the reduction in the second approach of the proof in Prop 4.2.2.

Therefore, we have

$$(\mathsf{Adv}^{\mathsf{tReGA}}(\mathcal{A}(E; X)))^2 \leq \mathsf{Adv}_{\mathsf{iCDH}}(\mathcal{B}_2) + 2\mathsf{Adv}_{\mathsf{iCDH}}(\mathcal{B}_3).$$

□

Appendix B

Verifiable Random Functions

B.1 Hardness of Twisted Master Decisional Problem

We start from a quick recap of the assumptions in Sec. 7.2.3.

Definition B.1.1 (Decisional Square CSIDH (sDDH) Problem). *Let $(G, \mathcal{E}, \star, E_0)$ be a group action. The decisional square CSIDH problem is that the adversary $\mathcal{A}$ is given $T_b = (g_1 \star E_0, h_b \star E_0)$ where $h_0 = g_1^2, h_1 = g_2$ and $(g_1, g_2, b) \leftarrow G^2 \times \{0,1\}$ and return $b' \in \{0,1\}$ to guess b.*

Definition B.1.2 (Decisional Reciprocal CSIDH (rDDH) Problem). *Let $(G, \mathcal{E}, \star, E_0)$ be a group action. The decisional reciprocal CSIDH problem is that the adversary $\mathcal{A}$ is given $T_b = (g_1 \star E_0, g_2 \star E_0, h_b \star E_0, h'_b \star E_0)$ where $h_0 = g_1 g_2, h_1 = g_3, h'_0 = g_1 g_2^{-1}, h'_1 = g_4$ and $(g_1, g_2, g_3, g_4, b) \leftarrow G^4 \times \{0,1\}$, and return $b' \in \{0,1\}$ to guess b.*

Definition B.1.3 (Multi-challenge Decisional Reciprocal CSIDH (mcrDDH) Problem). *Let $(G, \mathcal{E}, \star, E_0)$ be a group action and $b \in \{0,1\}$. The multi-challenge decisional reciprocal Diffie-Hellman experiment $\mathsf{Exp}^{\mathsf{mcrDDH}}(b)$ on input b proceeds as follows. The adversary $\mathcal{A}$ is given $(g_1 \star E_0)$ where $g_1 \leftarrow G$ together with access to $\mathcal{O}_b^{\mathsf{mcrDDH}}$ defined as follows:*

1. *$\mathcal{O}_0^{\mathsf{mcrDDH}}$: $(g_2 \star E_0, (g_1 g_2) \star E_0, (g_1 g_2^{-1}) \star E_0)$ where $g_2 \leftarrow G$,*
2. *$\mathcal{O}_1^{\mathsf{mcrDDH}}$: $(g_2 \star E_0, g_3 \star E_0, g_4 \star E_0)$ where $g_2, g_3, g_4 \leftarrow G$,*

and outputs $b' \in \{0,1\}$ to guess b.

We denote the advantage of a mcrDDH problem adversary $\mathcal{A}$ problem by

$$\mathsf{Adv}^{\mathsf{mcrDDH}}(\mathcal{A}) = \left|\Pr[\mathcal{A}(\mathsf{Exp}^{\mathsf{mcrDDH}}(b=0)) \to 1] - \Pr[\mathcal{A}(\mathsf{Exp}^{\mathsf{mcrDDH}}(b=1)) \to 1]\right|,$$

where b is the randomness in the experiment, and the probability is taken over the randomness used by $\mathcal{A}$ and the randomness used in the experiment.
The group action $(G, \mathcal{E}, \star, E_0)$ is implicitly parameterized in the experiment. We say the mcrDDH problem is hard, if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl such that

$\mathsf{Adv}^{\mathsf{mcrDDH}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$. One can use a standard hybrid argument to give a reduction from the rDDH problem to the mcrDDH problem. We skip the proof here.

Definition B.1.4 (Twisted Master Decisional CSIDH (tMDDH) Problem)**.** *Let $(G, \mathcal{E}, \star, E_0)$ be a group action, $n \in \mathbb{N}$, and $b \in \{0, 1\}$. The twisted master DDH problem experiment $\mathsf{Exp}^{\mathsf{tMDDH}}(n, b)$ on input (n, b) proceeds as follows.*

1. *The challenger $\mathcal{C}$ computes $E = g \star E_0$ where $g \leftarrow G$.*
2. *$\mathcal{C}$ generates a tuple $(g_1 \star E, \cdots, g_n \star E)$ where $g_1, \cdots, g_n \leftarrow G$, and sends the tuple to the adversary $\mathcal{A}$.*
3. *$\mathcal{A}$ is given access to a Diffie-Hellman (DH) oracle on input $(x_1, \cdots, x_n) \in \{0, \pm 1\}^n$ returning $\prod_i^n g_i^{x_i} \star E$.*
4. *$\mathcal{A}$ sends a string $v = (v_1, \cdots, v_n) \in \{0, \pm 1\}^n$ to $\prod_i^n g_i^{v_i} \star E$ to the challenge oracle $\mathcal{C}$.*
5. *$\mathcal{C}$ ignores if v has been queried before or is of Hamming weight less than 2. Otherwise, $\mathcal{C}$, depending on b, computes $X_0 = \prod_i^n g_i^{v_i} \star E$ or $X_1 = r \star E$ for some $r \leftarrow G$, and send X_b to $\mathcal{A}$. This process will only output for one time.*
6. *$\mathcal{A}$ outputs $b' \in \{0, 1\}$ to guess b.*

We denote the advantage of the decisional problem adversary $\mathcal{A}$ by

$$\mathsf{Adv}^{\mathsf{tMDDH}}(\mathcal{A}) = \left| \Pr[\mathcal{A}(\mathsf{Exp}^{\mathsf{tMDDH}}(n, b = 0)) \to 1] - \Pr[\mathcal{A}(\mathsf{Exp}^{\mathsf{tMDDH}}(n, b = 1)) \to 1] \right|,$$

where b is the randomness in the experiment, and the probability is taken over the randomness used by $\mathcal{A}$ and the randomness used in the experiment. The group action $(G, \mathcal{E}, \star, E_0)$ is implicitly parameterized in the experiment. We say the tMDDH problem is hard, if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl such that $\mathsf{Adv}^{\mathsf{tMDDH}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$.

Theorem B.1.5. *The* tMDDH *problem is not easier than the* mcrDDH *problem. Concretely, let $\mathcal{A}$ be an adversary against the* mcrDDH *problem with the parameter $(G, \mathcal{E}, \star, E_0)$ and $n \in \mathbb{N}$. If at most $q_{\mathsf{DH}} = poly(\lambda)$ queries are made in the* mcrDDH *experiment, then there exists* tMDDH *problem adversaries $\mathcal{B}_2, \cdots \mathcal{B}_n$ such that*

$$\mathsf{Adv}^{\mathsf{tMDDH}}(\mathcal{A}) \leq \sum_{i=2}^{n} \mathsf{Adv}^{\mathsf{mcrDDH}}(\mathcal{B}_i).$$

Proof. We prove the theorem via a hybrid argument by introducing two series of games $\mathsf{Game}_1, \cdots, \mathsf{Game}_n$ by modifying the responses of the DH oracle and the challenge oracle in the tMDDH experiment gradually. Among the games, Game_1 be the original tMDDH experiment, We will modify the response of the challenge oracle and the DH oracle together, which will be explained later. For $i \in [n]$ where $b \in \{0, 1\}$, let $\mathcal{A}(\mathsf{Game}_i(b))$ represent $\mathcal{A}$ running the Game_i, the modified tMDDH experiment with the random coin b used in the experiment, and $\mathcal{A}$ will return 0 or 1. Therefore, by definition,

$$\mathsf{Adv}^{\mathsf{tMDDH}}(\mathcal{A}) = |\Pr[\mathcal{A}(\mathsf{Game}_1(b=1)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_1(b=0)) \to 1]|. \tag{B.1}$$

Looking ahead, Game_n is the modified tMDDH experiment where both the DH oracle and the challenger reply with random elements in $\mathcal{E}$. Therefore, since b is information theoretically hidden from $\mathcal{A}$,

$$|\Pr[\mathcal{A}(\mathsf{Game}_n(b=1)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_n(b=0)) \to 1]| = 0. \tag{B.2}$$

Game_1 : the original tMDDH experiment starting with a tuple $(g_1 \star E, \cdots, g_n \star E)$ where $g_1, \cdots, g_n \leftarrow G$ and the oracle response as specified.

Game_2 to Game_n: for $j \in \{2, \cdots, n\}$, Game_j is the same as Game_{j-1} except that the response of the DH oracle and the challenge oracle is modified as follows. The modification starts with a list L which is initially

$$\{(\mathbf{0}, E), (\mathbf{e}_1, g_1 \star E), \cdots, (\mathbf{e}_j, g_j \star E)\} \subseteq \{0, \pm 1\}^j \times \mathcal{E}.$$

On the query $x = (x_1, \cdots, x_n) \in \{0, \pm 1\}^n$, if $((x_1, \cdots, x_j), X) \in L$ for some $X \in \mathcal{E}$, the oracle returns $(\prod_{i=j+1}^n g_i^{x_i}) \star X$; otherwise, it draws $g' \leftarrow G$, computes $X = g' \star E$, adds $((x_1, \cdots, x_j), X)$ to the list L, and returns $(\prod_{i=j+1}^n g_i^{x_i}) \star X$ to $\mathcal{A}$. The reply for the challenge query is modified in the same way if the random coin $b = 0$.

Claim that $\mathsf{Game}_{j-1} \approx_c \mathsf{Game}_j$ for $\mathcal{A}$ for any $2 \le j \le n$ by assuming the mcrDDH problem. Concretely, a reduction $\mathcal{B}_j$ to the mcrDDH problem proceeds as follows

1. Obtain $T = (g' \star E_0, \{(X_i, X_i', X_i'')\}_{i \in [q_{\mathsf{DH}}+j-1]})$ from the mcrDDH oracle.
2. Overwrite the notations of X_i, X_i', X_i'' by $\left(g' \star E, \{X_i, X_i', X_i''\}_{i \in [q_{\mathsf{DH}}+j-1]}\right) \leftarrow g \star T$ where $g \leftarrow G$.
3. Then, $\mathcal{B}_j$ initializes with a list

$$L = \left\{ \begin{matrix} (\mathbf{e}_1, X_1), \cdots, (\mathbf{e}_{j-1}, X_{j-1}), (\mathbf{e}_1 + \mathbf{e}_j, X_1'), \cdots, (\mathbf{e}_{j-1} + \mathbf{e}_j, X_{j-1}'), (\mathbf{0}, E), \\ (\mathbf{e}_1 - \mathbf{e}_j, X_1''), \cdots, (\mathbf{e}_{j-1} - \mathbf{e}_j, X_{j-1}''), (\mathbf{e}_j, g' \star E) \end{matrix} \right\} \subset \{0, \pm 1\}^j \times \mathcal{E},$$

4. Invoke $\mathcal{A}$ on input $(E, X_1, \cdots, X_{j-1}, g' \star E_0, g_{j+1} \star E_0, \cdots, g_n \star E_0)$ where $g_{j+1}, \cdots, g_n \leftarrow G$.
5. Upon receiving the oracle query $(x_1, \cdots, x_n) \in \{0, \pm 1\}^n$, check whether $((x_1, \cdots, x_j), X) \in L$ for some $X \in \mathcal{E}$. If so, return $\prod_{i=j+1}^n g_i^{x_i} \star X$. Otherwise, update

$$L \leftarrow \{((x_1, \cdots, x_{j-1}, 0), X_{\mathsf{ct}}),\ ((x_1, \cdots, x_{j-1}, 1), X_{\mathsf{ct}}'), ((x_1, \cdots, x_{j-1}, -1), X_{\mathsf{ct}}'')\} \cup L,$$

and set $\mathsf{ct} \leftarrow \mathsf{ct} + 1$, and rerun this step again.

6. Output whatever $\mathcal{A}$ returns.

Note that in Step 1. if $\mathcal{B}_j$ is in $\mathsf{Exp}^{\mathsf{mcrDDH}}(0)$ (Def. B.1.3 Item 1) then $\mathcal{B}_j$ generates Game_{j-1} because $g' \star X_i = X_i'$ and $g'^{-1} \star X_i = X_i''$. In contrast, if it is in $\mathsf{Exp}^{\mathsf{mcrDDH}}(1)$ (Def. B.1.3 Item 2), then $\mathcal{B}_j$ generates Game_j. It follows that for $b \in \{0,1\}$,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{mcrDDH}}(\mathcal{B}_j) &= |\Pr[\mathcal{B}_j(\mathsf{Exp}^{\mathsf{mcrDDH}}(0)) \to 1] - \Pr[\mathcal{B}_j(\mathsf{Exp}^{\mathsf{mcrDDH}}(1)) \to 1]| \\ &= |\Pr[\mathcal{A}(\mathsf{Game}_{j-1}(b)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_j(b)) \to 1]|. \end{aligned} \tag{B.3}$$

Therefore, we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{tMDDH}}(\mathcal{A}) &= |\Pr[\mathcal{A}(\mathsf{Game}_1(b=1)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_1(b=0)) \to 1]| && \text{(By Eq. (B.1))} \\ &\leq \sum_{j=2}^{n}(|\Pr[\mathcal{A}(\mathsf{Game}_{j-1}(b=1)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_j(b=0)) \to 1]| \\ &\quad + |\Pr[\mathcal{A}(\mathsf{Game}_{j-1}(b=1)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_1(b=0)) \to 1]|) \\ &\quad + |\Pr[\mathcal{A}(\mathsf{Game}_n(b=1)) \to 1] - \Pr[\mathcal{A}(\mathsf{Game}_{j-1}(b=1)) \to 1]| \\ &&& \text{(Union bounds.)} \\ &= \sum_{j=2}^{n-1} \mathsf{Adv}^{\mathsf{mcrDDH}}(\mathcal{B}_j). && \text{(By Eqs. (B.2) and (B.3))}\end{aligned}$$

The result follows. □

Bibliography

[Abe01] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 136–151. Springer, Heidelberg, May 2001.

[ACF14] Michel Abdalla, Dario Catalano, and Dario Fiore. Verifiable random functions: Relations to identity-based key encapsulation and new constructions. *Journal of Cryptology*, 27(3):544–593, July 2014.

[ACL$^+$22] Martin R. Albrecht, Valerio Cini, Russell W. F. Lai, Giulio Malavolta, and Sri AravindaKrishnan Thyagarajan. Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable. Cryptology ePrint Archive, Report 2022/941, 2022. https://eprint.iacr.org/2022/941.

[ADMP20] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 411–439. Springer, Heidelberg, December 2020.

[AEK$^+$22] Michel Abdalla, Thorsten Eisenhofer, Eike Kiltz, Sabrina Kunzweiler, and Doreen Riepel. Password-authenticated key exchange from group actions. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 699–728. Springer, Heidelberg, August 2022.

[AF96] Masayuki Abe and Eiichiro Fujisaki. How to date blind signatures. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 244–251. Springer, Heidelberg, November 1996.

[AFMP20] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In *ASIACRYPT 2020*, volume 12492 of *Springer LNCS*, pages 411–439, 2020.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2087–2104. ACM Press, October / November 2017.

[AHIV22] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. Cryptology ePrint Archive, Paper 2022/1608, 2022. https://eprint.iacr.org/2022/1608.

[AKSY22] Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav. Practical, round-optimal lattice-based blind signatures. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 39–53. ACM Press, November 2022.

[ALS20] Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. Practical product proofs for lattice commitments. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 470–499. Springer, Heidelberg, August 2020.

[AO00] Masayuki Abe and Tatsuaki Okamoto. Provably secure partially blind signatures. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 271–286. Springer, Heidelberg, August 2000.

[BBD$^+$22] Jeremy Booher, Ross Bowden, Javad Doliskani, Tako Boris Fouotsa, Steven D. Galbraith, Sabrina Kunzweiler, Simon-Philipp Merz, Christophe Petit, Benjamin Smith, Katherine E. Stange, Yan Bo Ti, Christelle Vincent, José Felipe Voloch, Charlotte Weitkämper, and Lukas Zobernig. Failing to hash into supersingular isogeny graphs. Cryptology ePrint Archive, Report 2022/518, 2022. https://eprint.iacr.org/2022/518.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004.

[BCC04] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 2004*, pages 132–145. ACM Press, October 2004.

[BCC$^+$15] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *ESORICS 2015, Part I*, volume 9326 of *LNCS*, pages 243–265. Springer, Heidelberg, September 2015.

[BCC$^+$16a] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 16*, volume 9696 of *LNCS*, pages 117–136. Springer, Heidelberg, June 2016.

[BCC$^+$16b] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Heidelberg, May 2016.

[BCC$^+$23] Andrea Basso, Giulio Codogni, Deirdre Connolly, Luca De Feo, Tako Boris Fouotsa, Guido Maria Lido, Travis Morrison, Lorenz Panny, Sikhar Patranabis, and Benjamin Wesolowski. Supersingular curves you can trust. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 405–437. Springer, Heidelberg, April 2023.

[BCN$^+$10] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10*, volume 6280 of *LNCS*, pages 381–398. Springer, Heidelberg, September 2010.

[BCN18] Cecilia Boschini, Jan Camenisch, and Gregory Neven. Floppy-sized group signatures from lattices. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18*, volume 10892 of *LNCS*, pages 163–182. Springer, Heidelberg, July 2018.

[BCR$^+$19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Heidelberg, October / November 2016.

[BD21] Jeffrey Burdges and Luca De Feo. Delay encryption. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 302–326. Springer, Heidelberg, October 2021.

[BDD$^+$17] Paulo SLM Barreto, Bernardo David, Rafael Dowsley, Kirill Morozov, and Anderson CA Nascimento. A framework for efficient adaptively secure composable oblivious transfer in the ROM. *arXiv preprint arXiv:1710.08256*, 2017.

[BDE$^+$22] Maxime Buser, Rafael Dowsley, Muhammed F. Esgin, Shabnam Kasra Kermanshahi, Veronika Kuchta, Joseph K. Liu, Raphaël C.-W. Phan, and Zhenfei Zhang. Post-quantum verifiable random function from symmetric primitives in PoS blockchain. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *ESORICS 2022, Part I*, volume 13554 of *LNCS*, pages 25–45. Springer, Heidelberg, September 2022.

[BDE$^+$23] Maxime Buser, Rafael Dowsley, Muhammed Esgin, Clémentine Gritti, Shabnam Kasra Kermanshahi, Veronika Kuchta, Jason Legrow, Joseph Liu, Raphaël Phan, Amin Sakzad, et al. A survey on exotic signatures for post-quantum blockchain: Challenges and research directions. *ACM Computing Surveys*, 55(12):1–32, 2023.

[BDK$^+$21] Carsten Baum, Cyprien Delpech de Saint Guilhem, Daniel Kales, Emmanuela Orsini, Peter Scholl, and Greg Zaverucha. Banquet: Short and fast signatures from AES. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 266–297. Springer, Heidelberg, May 2021.

[BDK$^+$22] Ward Beullens, Samuel Dobson, Shuichi Katsumata, Yi-Fu Lai, and Federico Pintore. Group signatures and more from isogenies and lattices: Generic, simple, and efficient. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 95–126. Springer, Heidelberg, May / June 2022.

[BFH$^+$20] Rishabh Bhadauria, Zhiyong Fang, Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, Tiancheng Xie, and Yupeng Zhang. Ligero++: A new optimized sublinear IOP. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 2025–2038. ACM Press, November 2020.

[BFJ16] Jean-François Biasse, Claus Fieker, and Michael J Jacobson. Fast heuristic algorithms for computing relations in the class group of a quadratic order, with applications to isogeny evaluation. *LMS Journal of Computation and Mathematics*, 19(A):371–390, 2016.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–432. Springer, Heidelberg, May 2003.

[BGZ23] Dan Boneh, Jiaxin Guan, and Mark Zhandry. A lower bound on the length of signatures based on group actions and generic isogenies. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 507–531. Springer, Heidelberg, April 2023.

[BHSB19] Michael Backes, Lucjan Hanzlik, and Jonas Schneider-Bensch. Membership privacy for fully dynamic group signatures. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2181–2198. ACM Press, November 2019.

[BJS14] Jean-François Biasse, David Jao, and Anirudh Sankar. A quantum algorithm for computing isogenies between supersingular elliptic curves. In Willi Meier and Debdeep Mukhopadhyay, editors, *INDOCRYPT 2014*, volume 8885 of *LNCS*, pages 428–442. Springer, Heidelberg, December 2014.

[BKL+22] Gustavo Banegas, Juliane Krämer, Tanja Lange, Michael Meyer, Lorenz Panny, Krijn Reijnders, Jana Sotáková, and Monika Trimoska. Disorientation faults in CSIDH. Cryptology ePrint Archive, Report 2022/1202, 2022. https://eprint.iacr.org/2022/1202.

[BKM+21] Andrea Basso, Péter Kutas, Simon-Philipp Merz, Christophe Petit, and Antonio Sanso. Cryptanalysis of an oblivious PRF from supersingular isogenies. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part I*, volume 13090 of *LNCS*, pages 160–184. Springer, Heidelberg, December 2021.

[BKP20] Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 464–492. Springer, Heidelberg, December 2020.

[BKV19] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 227–247. Springer, Heidelberg, December 2019.

[BKW20] Dan Boneh, Dmitry Kogan, and Katharine Woo. Oblivious pseudorandom functions from isogenies. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 520–550. Springer, Heidelberg, December 2020.

[BL07] Ernie Brickell and Jiangtao Li. Enhanced privacy id: A direct anonymous attestation scheme with enhanced revocation capabilities. In *Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 21–30, 2007.

[BLL+21] Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 33–53. Springer, Heidelberg, October 2021.

[BLMW07] Emmanuel Bresson, Yassine Lakhnech, Laurent Mazaré, and Bogdan Warinschi. A generalization of DDH with applications to protocol analysis and computational soundness. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 482–499. Springer, Heidelberg, August 2007.

[BLNS23] Ward Beullens, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Lattice-based blind signatures: Short, efficient, and round-optimal. Cryptology ePrint Archive, Report 2023/077, 2023. https://eprint.iacr.org/2023/077.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

[BLS19] Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 176–202. Springer, Heidelberg, August 2019.

[BM89] Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and spplications. In *CRYPTO '89*, pages 547–557, 1989.

[BMM$^+$22] Saikrishna Badrinarayanan, Daniel Masny, Pratyay Mukherjee, Sikhar Patranabis, Srinivasan Raghuraman, and Pratik Sarkar. Round-optimal oblivious transfer and MPC from computational CSIDH. Cryptology ePrint Archive, Report 2022/1511, 2022. https://eprint.iacr.org/2022/1511.

[BMR10] Dan Boneh, Hart William Montgomery, and Ananth Raghunathan. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 131–140. ACM Press, October 2010.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, Heidelberg, May 2003.

[BN18] Xavier Bonnetain and María Naya-Plasencia. Hidden shift quantum cryptanalysis and implications. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 560–592. Springer, Heidelberg, December 2018.

[Boy08] Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 39–56. Springer, Heidelberg, September 2008.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Heidelberg, April 2012.

[Bra94] Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 302–318. Springer, Heidelberg, August 1994.

[BS20] Xavier Bonnetain and André Schrottenloher. Quantum security analysis of CSIDH. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 493–522. Springer, Heidelberg, May 2020.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, Heidelberg, February 2005.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of CRYPTOLOGY*, 13(1):143–202, 2000.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001.

[CD23] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 423–447. Springer, Heidelberg, April 2023.

[CDEL21] Wouter Castryck, Ann Dooms, Carlo Emerencia, and Alexander Lemmens. A fusion algorithm for solving the hidden shift problem in finite abelian groups. In Jung Hee Cheon and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021*, pages 133–153. Springer, Heidelberg, 2021.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Heidelberg, August 1994.

[CFN90] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 319–327. Springer, Heidelberg, August 1990.

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

[Cha88] David Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, pages 177–182. Springer, Heidelberg, May 1988.

[CHVW22] Wouter Castryck, Marc Houben, Frederik Vercauteren, and Benjamin Wesolowski. On the decisional diffie–hellman problem for class group actions on oriented elliptic curves. *Research in Number Theory*, 8(4):99, 2022.

[CJJ22] Arka Rai Choudhuri, Abhihsek Jain, and Zhengzhong Jin. SNARGs for $\mathcal{P}$ from LWE. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 68–79. IEEE, 2022.

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In *EUROCRYPT 2003*, Springer LNCS, pages 68–86, 2003.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Heidelberg, May 2001.

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Heidelberg, August 2006.

[CLG09] Denis Xavier Charles, Kristin E. Lauter, and Eyal Z. Goren. Cryptographic hash functions from expander graphs. *Journal of Cryptology*, 22(1):93–113, January 2009.

[CLL23] Kelong Cong, Yi-Fu Lai, and Shai Levin. Efficient isogeny proofs using generic techniques. In Mehdi Tibouchi and XiaoFeng Wang, editors, *Applied Cryptography and Network Security*, pages 248–275, Cham, 2023. Springer Nature Switzerland.

[CLM$^+$18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part III*, volume 11274 of *LNCS*, pages 395–427. Springer, Heidelberg, December 2018.

[CO15] Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In *International Conference on Cryptology and Information Security in Latin America*, volume 9230 of *Springer LNCS*, pages 40–58. Springer, 2015.

[Cou06] Jean Marc Couveignes. Hard homogeneous spaces., 1997. *IACR Cryptology ePrint Archive*, 2006:291, 2006.

[Cox11] David A Cox. *Primes of the form $x^2 + ny^2$: Fermat, class field theory, and complex multiplication.* John Wiley & Sons, 2011.

[Cox22] David A Cox. *Primes of the Form x2+ ny2: Fermat, Class Field Theory, and Complex Multiplication. with Solutions*, volume 387. American Mathematical Soc., 2022.

[CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Heidelberg, August 1993.

[CS20] Remi Clarisse and Olivier Sanders. Group signature without random oracles from randomizable signatures. In Khoa Nguyen, Wenling Wu, Kwok-Yan Lam, and Huaxiong Wang, editors, *ProvSec 2020*, volume 12505 of *LNCS*, pages 3–23. Springer, Heidelberg, November / December 2020.

[CSCJR22] Jorge Chávez-Saab, Jesús-Javier Chi-Domínguez, Samuel Jaques, and Francisco Rodríguez-Henríquez. The SQALE of CSIDH: sublinear Vélu quantum-resistant isogeny action with low exponents. *Journal of Cryptographic Engineering*, 12(3):349–368, September 2022.

[CSRT22] Jorge Chávez-Saab, Francisco Rodríguez-Henríquez, and Mehdi Tibouchi. Verifiable isogeny walks: Towards an isogeny-based postquantum VDF. In Riham AlTawy and Andreas Hülsing, editors, *SAC 2021*, volume 13203 of *LNCS*, pages 441–460. Springer, Heidelberg, September / October 2022.

[CSV20] Wouter Castryck, Jana Sotáková, and Frederik Vercauteren. Breaking the decisional Diffie-Hellman problem for class group actions using genus theory. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 92–120. Springer, Heidelberg, August 2020.

[CSW20] Ran Canetti, Pratik Sarkar, and Xiao Wang. Blazing fast OT for three-round UC OT extension. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 299–327. Springer, Heidelberg, May 2020.

[Cv91] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Heidelberg, April 1991.

[CvdGT95] Claude Crépeau, Jeroen van de Graaf, and Alain Tapp. Committed oblivious transfer and private multi-party computation. In *CRYPTO '95*, volume 963 of *Springer LNCS*, pages 110–123, 1995.

[DDGZ22] Luca De Feo, Samuel Dobson, Steven D. Galbraith, and Lukas Zobernig. SIDH proof of knowledge. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 310–339. Springer, Heidelberg, December 2022.

[DDOS19] Cyprien Delpech de Saint Guilhem, Lauren De Meyer, Emmanuela Orsini, and Nigel P. Smart. BBQ: Using AES in picnic signatures. In Kenneth G. Paterson and Douglas Stebila, editors, *SAC 2019*, volume 11959 of *LNCS*, pages 669–692. Springer, Heidelberg, August 2019.

[DF19] Luca De Feo. Seasign: Compact isogeny signatures from class group actions, 2019. Talk at Eurocrypt 2019.

[DFJP14] Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *Journal of Mathematical Cryptology*, 8(3):209–247, 2014.

[DFK$^+$23] Luca De Feo, Tako Boris Fouotsa, Péter Kutas, Antonin Leroux, Simon-Philipp Merz, Lorenz Panny, and Benjamin Wesolowski. SCALLOP: Scaling the CSI-FiSh. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 345–375. Springer, Heidelberg, May 2023.

[DFMS22] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Online-extractability in the quantum random-oracle model. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 677–706. Springer, Heidelberg, May / June 2022.

[DG16] Christina Delfs and Steven D Galbraith. Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$. *Designs, Codes and Cryptography*, 78(2):425–440, 2016.

[DG19] Luca De Feo and Steven D. Galbraith. SeaSign: Compact isogeny signatures from class group actions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 759–789. Springer, Heidelberg, May 2019.

[DGH$^+$20] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, Daniel Masny, and Daniel Wichs. Two-round oblivious transfer from CDH or LPN. In *EUROCRYPT 2020*, volume 12106 of *Springer LNCS*, pages 768–797, 2020.

[DGKR18] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 66–98. Springer, Heidelberg, April / May 2018.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DHK$^+$23] Julien Duman, Dominik Hartmann, Eike Kiltz, Sabrina Kunzweiler, Jonas Lehmann, and Doreen Riepel. Generic models for group actions. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 406–435. Springer, Heidelberg, May 2023.

[dK22] Rafaël del Pino and Shuichi Katsumata. A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 306–336. Springer, Heidelberg, August 2022.

[DKL$^+$20] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact post-quantum signatures from quaternions and isogenies. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 64–93. Springer, Heidelberg, December 2020.

[DLLW23] Luca De Feo, Antonin Leroux, Patrick Longa, and Benjamin Wesolowski. New algorithms for the deuring correspondence - towards practical and secure SQISign signatures. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 659–690. Springer, Heidelberg, April 2023.

[dLS18] Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. Lattice-based group signatures and zero-knowledge proofs of automorphism stability. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 574–591. ACM Press, October 2018.

[DM20] Luca De Feo and Michael Meyer. Threshold schemes from isogeny assumptions. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 187–212. Springer, Heidelberg, May 2020.

[DMPS19] Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. Verifiable delay functions from supersingular isogenies and pairings. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 248–277. Springer, Heidelberg, December 2019.

[DN19] Itai Dinur and Niv Nadler. Multi-target attacks on the Picnic signature scheme and related protocols. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 699–727. Springer, Heidelberg, May 2019.

[DNMQ12] Bernardo Machado David, Anderson CA Nascimento, and Jörn Müller-Quade. Universally composable oblivious transfer from lossy encryption and the McEliece assumptions. In *International Conference on Information Theoretic Security*, pages 80–99. Springer, 2012.

[DOPS20] Cyprien Delpech de Saint Guilhem, Emmanuela Orsini, Christophe Petit, and Nigel P. Smart. Semi-commutative masking: A framework for isogeny-based protocols, with an application to fully secure two-round isogeny-based OT. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *CANS 20*, volume 12579 of *LNCS*, pages 235–258. Springer, Heidelberg, December 2020.

[DOT21] Cyprien Delpech de Saint Guilhem, Emmanuela Orsini, and Titouan Tanguy. Limbo: Efficient zero-knowledge MPCitH-based arguments. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 3022–3036. ACM Press, November 2021.

[DP06] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 06*, volume 4341 of *LNCS*, pages 193–210. Springer, Heidelberg, September 2006.

[DPV19] Thomas Decru, Lorenz Panny, and Frederik Vercauteren. Faster SeaSign signatures through improved rejection sampling. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 271–285. Springer, Heidelberg, 2019.

[dQKL$^+$21] Victoria de Quehen, Péter Kutas, Chris Leonardi, Chloe Martindale, Lorenz Panny, Christophe Petit, and Katherine E. Stange. Improved torsion-point attacks on SIDH variants. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 432–470, Virtual Event, August 2021. Springer, Heidelberg.

[DS18] David Derler and Daniel Slamanig. Highly-efficient fully-anonymous dynamic group signatures. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *ASIACCS 18*, pages 551–565. ACM Press, April 2018.

[DvdGMN08] Rafael Dowsley, Jeroen van de Graaf, Jörn Müller-Quade, and Anderson C. A. Nascimento. Oblivious transfer based on the McEliece assumptions. In *ICITS 2008*, volume 5155 of *Springer LNCS*, pages 107–117, 2008.

[EKP20] Ali El Kaafarani, Shuichi Katsumata, and Federico Pintore. Lossy CSI-FiSh: Efficient signature scheme with tight reduction to decisional CSIDH-512. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 157–186. Springer, Heidelberg, May 2020.

[EKS$^+$21] Muhammed F. Esgin, Veronika Kuchta, Amin Sakzad, Ron Steinfeld, Zhenfei Zhang, Shifeng Sun, and Shumo Chu. Practical post-quantum few-time verifiable random function with applications to algorand. In Nikita Borisov and Claudia Diaz, editors, *FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part II*, volume 12675 of *Lecture Notes in Computer Science*, pages 560–578. Springer, 2021.

[ELL$^+$15] Martianus Frederic Ezerman, Hyung Tae Lee, San Ling, Khoa Nguyen, and Huaxiong Wang. A provably secure group signature scheme from code-based assumptions. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 260–285. Springer, Heidelberg, November / December 2015.

[ENS20] Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 259–288. Springer, Heidelberg, December 2020.

[ESLL19] Muhammed F. Esgin, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 115–146. Springer, Heidelberg, August 2019.

[ESLR22] Muhammed F. Esgin, Ron Steinfeld, Dongxi Liu, and Sushmita Ruj. Efficient hybrid exact/relaxed lattice proofs and applications to rounding and VRFs. Cryptology ePrint Archive, Report 2022/141, 2022. https://eprint.iacr.org/2022/141.

[ESZ22] Muhammed F. Esgin, Ron Steinfeld, and Raymond K. Zhao. MatRiCT$^+$: More efficient post-quantum private blockchain payments. In *2022 IEEE Symposium on Security and Privacy*, pages 1281–1298. IEEE Computer Society Press, May 2022.

[EZS$^+$19] Muhammed F. Esgin, Raymond K. Zhao, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 567–584. ACM Press, November 2019.

[Fel19] Joël Felderhoff. Hard homogeneous spaces and commutative supersingular isogeny based diffie-hellman. Internship report, LIX, Ecole polytechnique ; ENS de Lyon, August 2019.

[FG19] Luca De Feo and Steven D. Galbraith. Seasign: Compact isogeny signatures from class group actions. In *EUROCRYPT 2019*, volume 11478 of *Springer LNCS*, pages 759–789, 2019.

[FI06] Jun Furukawa and Hideki Imai. An efficient group signature scheme from bilinear maps. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 89(5):1328–1338, 2006.

[FIM$^+$14] Katalin Friedl, Gábor Ivanyos, Frédéric Magniez, Miklos Santha, and Pranab Sen. Hidden translation and translating coset in quantum computing. *SIAM Journal on Computing*, 43(1):1–24, 2014.

[Fis06] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 60–77. Springer, Heidelberg, August 2006.

[FJR22] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. Cryptology ePrint Archive, Report 2022/188, 2022. https://eprint.iacr.org/2022/188.

[FMRV22] Thibauld Feneuil, Jules Maire, Matthieu Rivain, and Damien Vergnaud. Zero-knowledge protocols for the subset sum problem from MPC-in-the-head with rejection. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 371–402. Springer, Heidelberg, December 2022.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Heidelberg, August 1999.

[FOO93] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In Jennifer Seberry and Yuliang Zheng, editors, *AUSCRYPT'92*, volume 718 of *LNCS*, pages 244–251. Springer, Heidelberg, December 1993.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[Gal99] Steven D Galbraith. Constructing isogenies between elliptic curves over finite fields. *LMS Journal of Computation and Mathematics*, 2:118–138, 1999.

[GHM$^+$17] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 51–68. ACM, 2017.

[GHS02] Steven D. Galbraith, Florian Hess, and Nigel P. Smart. Extending the GHS Weil descent attack. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 29–44. Springer, Heidelberg, April / May 2002.

[GKV10] S. Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. A group signature scheme from lattice assumptions. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 395–412. Springer, Heidelberg, December 2010.

[GL22] Steven D. Galbraith and Yi-Fu Lai. Attack on sheals and heals: The second wave of GPST. In Jung Hee Cheon and Thomas Johansson, editors, *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022, Virtual Event, September 28-30, 2022, Proceedings*, volume 13512 of *Lecture Notes in Computer Science*, pages 399–421. Springer, 2022.

[GMNO18] Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. Lattice-based zk-SNARKs from square span programs. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 556–573. ACM Press, October 2018.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC*, pages 218–229. ACM, 1987.

[GN08] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 31–51. Springer, Heidelberg, April 2008.

[GNP$^+$15] Sharon Goldberg, Moni Naor, Dimitrios Papadopoulos, Leonid Reyzin, Sachin Vasant, and Asaf Ziv. NSEC5: Provably preventing DNSSEC zone enumeration. In *NDSS 2015*. The Internet Society, February 2015.

[Goo23] Suppressing quantum errors by scaling a surface code logical qubit. *Nature*, 614(7949):676–681, 2023.

[GPST16] Steven D. Galbraith, Christophe Petit, Barak Shani, and Yan Bo Ti. On the security of supersingular isogeny cryptosystems. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 63–91. Springer, Heidelberg, December 2016.

[GPSV21] Steven Galbraith, Lorenz Panny, Benjamin Smith, and Frederik Vercauteren. Quantum equivalence of the DLP and CDHP for group actions. *Mathematical Cryptology*, 1(1):40–44, Jun. 2021.

[GPV21] Wissam Ghantous, Federico Pintore, and Mattia Veroni. Collisions in supersingular isogeny graphs and the SIDH-based identification protocol. Cryptology ePrint Archive, Report 2021/1051, 2021. https://eprint.iacr.org/2021/1051.

[Gro07] Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 164–180. Springer, Heidelberg, December 2007.

[GS13] Steven Galbraith and Anton Stolbunov. Improved algorithm for the isogeny problem for ordinary elliptic curves. *Applicable Algebra in Engineering, Communication and Computing*, 24(2):107–131, 2013.

[Hal02] Sean Hallgren. Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem. In *34th ACM STOC*, pages 653–658. ACM Press, May 2002.

[Hal05] Sean Hallgren. Fast quantum algorithms for computing the unit group and class group of a number field. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 468–474. ACM Press, May 2005.

[HBD$^+$22] Andreas Hülsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. SPHINCS$^+$. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[HIP$^+$22] Scott Hendrickson, Jana Iyengar, Tommy Pauly, Steven Valdez, and Christopher A. Wood. Private access tokens. internet-draft draft-private-access-tokens-01, April 2022. Work in Progress.

[HKL19] Eduard Hauck, Eike Kiltz, and Julian Loss. A modular treatment of blind signatures from identification schemes. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 345–375. Springer, Heidelberg, May 2019.

[HKLN20] Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen. Lattice-based blind signatures, revisited. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 500–529. Springer, Heidelberg, August 2020.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient secure two-party protocols: Techniques and constructions*. Springer Science & Business Media, 2010.

[HMW18] Timo Hanke, Mahnush Movahedi, and Dominic Williams. DFINITY technology overview series, consensus system. *CoRR*, abs/1805.04548, 2018.

[IBM] IBM. Ibm unveils 400 qubit-plus quantum processor and next-generation ibm quantum system two. *IBM Newsroom*. available at https://newsroom.ibm.com/2022-11-09-IBM-Unveils-400-Qubit-Plus-Quantum-Processor-and-Next-Generation-IBM-Quantum-System-Two.

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM Journal on Computing*, 39(3):1121–1152, 2009.

[Ish20] Yuval Ishai. Zero-knowledge proofs from information-theoretic proof systems. *Zkproofs Blog*, 2020.

[JAC$^+$22] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, David Urbanik, Geovandro Pereira, Koray Karabina, and Aaron Hutchinson. SIKE. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

[Jag15] Tibor Jager. Verifiable random functions from weaker assumptions. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 121–143. Springer, Heidelberg, March 2015.

[JD11] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*, pages 19–34. Springer, Heidelberg, November / December 2011.

[JS19] Samuel Jaques and John M. Schanck. Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 32–61. Springer, Heidelberg, August 2019.

[Kit95] A Yu Kitaev. Quantum measurements and the abelian stabilizer problem. *arXiv preprint quant-ph/9511026*, 1995.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537. ACM Press, October 2018.

[KLLQ23] Shuichi Katsumata, Yi-Fu Lai, Jason T. LeGrow, and Ling Qin. CSI-Otter: Isogeny-based (partially) blind signatures from the class group action with a twist. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 729–761, Cham, 2023. Springer Nature Switzerland.

[KLX22a] Julia Kastner, Julian Loss, and Jiayu Xu. The abe-okamoto partially blind signature scheme revisited. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 279–309. Springer, Heidelberg, December 2022.

[KLX22b] Julia Kastner, Julian Loss, and Jiayu Xu. On pairing-free blind signature schemes in the algebraic group model. In Goichiro Hanaoka, Junji Shikata, and Yohei

Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 468–497. Springer, Heidelberg, March 2022.

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[KOS15] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 724–741. Springer, Heidelberg, August 2015.

[KP17] Sudhakar Kumawat and Souradyuti Paul. A new constant-size accountable ring signature scheme without random oracles. In *International Conference on Information Security and Cryptology*, pages 157–179. Springer, 2017.

[Kup05] Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM Journal on Computing*, 35(1):170–188, 2005.

[Kup11] Greg Kuperberg. Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *arXiv preprint arXiv:1112.3333*, 2011.

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 2003*, pages 155–164. ACM Press, October 2003.

[KY19] Shuichi Katsumata and Shota Yamada. Group signatures without NIZK: From lattices in the standard model. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 312–344. Springer, Heidelberg, May 2019.

[Laa15] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 3–22. Springer, Heidelberg, August 2015.

[Lai23] Yi-Fu Lai. Capybara and tsubaki: Verifiable random functions from group actions and isogenies. Cryptology ePrint Archive, Paper 2023/182, 2023. https://eprint.iacr.org/2023/182.

[LDK$^+$22] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[Ler22] Antonin Leroux. A new isogeny representation and applications to cryptography. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 3–35. Springer, Heidelberg, December 2022.

[LGD21] Yi-Fu Lai, Steven D. Galbraith, and Cyprien Delpech de Saint Guilhem. Compact, efficient and UC-secure isogeny-based oblivious transfer. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 213–241. Springer, Heidelberg, October 2021.

[Lin17] Yehuda Lindell. How to simulate it–a tutorial on the simulation proof technique. In *Tutorials on the Foundations of Cryptography*, pages 277–346. Springer, 2017.

[LLLS13] Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. Lattice-based group signatures with logarithmic signature size. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 41–61. Springer, Heidelberg, December 2013.

[LLNW16] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 1–31. Springer, Heidelberg, May 2016.

[LMPY16] Benoît Libert, Fabrice Mouhartem, Thomas Peters, and Moti Yung. Practical "signatures with efficient protocols" from simple assumptions. In Xiaofeng Chen, XiaoFeng Wang, and Xinyi Huang, editors, *ASIACCS 16*, pages 511–522. ACM Press, May / June 2016.

[LNP22] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Efficient lattice-based blind signatures via gaussian one-time signatures. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 498–527. Springer, Heidelberg, March 2022.

[LNS20] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Practical lattice-based zero-knowledge proofs for integer relations. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1051–1070. ACM Press, November 2020.

[LNS21] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. SMILE: Set membership from ideal lattices with applications to ring signatures and confidential transactions. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 611–640, Virtual Event, August 2021. Springer, Heidelberg.

[LNWX18] San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Constant-size group signatures from lattices. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 58–88. Springer, Heidelberg, March 2018.

[LPY15] Benoît Libert, Thomas Peters, and Moti Yung. Short group signatures via structure-preserving signatures: Standard model security from simple assumptions. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 296–316. Springer, Heidelberg, August 2015.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Heidelberg, December 2009.

[LZCS16] Russell W. F. Lai, Tao Zhang, Sherman S. M. Chow, and Dominique Schröder. Efficient sanitizable signatures without random oracles. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *ESORICS 2016, Part I*, volume 9878 of *LNCS*, pages 363–380. Springer, Heidelberg, September 2016.

[MCR19] Michael Meyer, Fabio Campos, and Steffen Reith. On lions and elligators: An efficient constant-time implementation of CSIDH. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 307–325. Springer, Heidelberg, 2019.

[Mil86] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *CRYPTO'85*, volume 218 of *LNCS*, pages 417–426. Springer, Heidelberg, August 1986.

[MMP22] Marzio Mula, Nadir Murru, and Federico Pintore. Random sampling of supersingular elliptic curves. Cryptology ePrint Archive, Report 2022/528, 2022. https://eprint.iacr.org/2022/528.

[MMP$^+$23] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 448–471. Springer, Heidelberg, April 2023.

[Mon18] Hart Montgomery. More efficient lattice PRFs from keyed pseudorandom synthesizers. In Debrup Chakraborty and Tetsu Iwata, editors, *INDOCRYPT 2018*, volume 11356 of *LNCS*, pages 190–211. Springer, Heidelberg, December 2018.

[MOT20] Tomoki Moriya, Hiroshi Onuki, and Tsuyoshi Takagi. SiGamal: A supersingular isogeny-based PKE and its application to a PRF. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 551–580. Springer, Heidelberg, December 2020.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, October 1999.

[MZ22] Hart Montgomery and Mark Zhandry. Full quantum equivalence of group action DLog and CDH, and more. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 3–32. Springer, Heidelberg, December 2022.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *12th SODA*, pages 448–457. ACM-SIAM, January 2001.

[NR99] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *Journal of Computer and System Sciences*, 58(2):336–375, 1999.

[Ode09] Goldreich Oded. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2009.

[Onu21] Hiroshi Onuki. On oriented supersingular elliptic curves. *Finite Fields and Their Applications*, 69:101777, 2021.

[OO92] Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 324–337. Springer, Heidelberg, August 1992.

[OZ23] Emmanuela Orsini and Riccardo Zanotto. Simple two-round ot in the explicit isogeny model. Cryptology ePrint Archive, Paper 2023/269, 2023. https://eprint.iacr.org/2023/269.

[Pan23] Lorenz Panny. Csi-fish really isn't polynomial-time, 2023. https://yx7.cc/blah/2023-04-14.html.

[Pas03] Rafael Pass. On deniability in the common reference string and random oracle model. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 316–337. Springer, Heidelberg, August 2003.

[Pei20] Chris Peikert. He gives C-sieves on the CSIDH. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 463–492. Springer, Heidelberg, May 2020.

[PH78] Stephen Pohlig and Martin Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on information Theory*, 24(1):106–110, 1978.

[PL17] Christophe Petit and Kristin Lauter. Hard and easy problems for supersingular isogeny graphs. Cryptology ePrint Archive, Report 2017/962, 2017. https://eprint.iacr.org/2017/962.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO 2008*, volume 5157 of *Springer LNCS*, pages 554–571, 2008.

[PWH$^+$17] Dimitrios Papadopoulos, Duane Wessels, Shumon Huque, Moni Naor, Jan Včelák, Leonid Reyzin, and Sharon Goldberg. Making NSEC5 practical for DNSSEC. Cryptology ePrint Archive, Report 2017/099, 2017. https://eprint.iacr.org/2017/099.

[Rab81] Michael O Rabin. How to exchange secrets with oblivious transfer. *Technical Report TR-81, Aiken Computation Lab, Harvard University*, page 187, 1981.

[Reg04] Oded Regev. A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space. *arXiv preprint quant-ph/0406151*, 2004.

[Rob23] Damien Robert. Breaking SIDH in polynomial time. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 472–503. Springer, Heidelberg, April 2023.

[RS06] Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies. *IACR Cryptology ePrint Archive*, 2006:145, 2006.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, February 1978.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, Heidelberg, December 2001.

[SAB$^+$22] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.

[Sch01] Claus-Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *ICICS 01*, volume 2229 of *LNCS*, pages 1–12. Springer, Heidelberg, November 2001.

[Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[Sil09] Joseph H Silverman. *The arithmetic of elliptic curves*, volume 106. Springer, 2009.

[Sta96] Markus Stadler. Publicly verifiable secret sharing. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 190–199. Springer, Heidelberg, May 1996.

[Tan07] Seiichiro Tani. An improved claw finding algorithm using quantum walk. In Luděk Kučera and Antonín Kučera, editors, *Mathematical Foundations of Computer Science 2007*, pages 536–547, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[Tha20] Justin Thaler. Proofs, arguments, and zero-knowledge, 2020.

[TZ22] Stefano Tessaro and Chenzhi Zhu. Short pairing-free blind signatures with exponential security. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 782–811. Springer, Heidelberg, May / June 2022.

[UJ20] David Urbanik and David Jao. New techniques for SIDH-based NIKE. *Journal of Mathematical Cryptology*, 14(1):120–128, 2020.

[Unr15] Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 755–784. Springer, Heidelberg, April 2015.

[Vit19] Vanessa Vitse. Simple oblivious transfer protocols compatible with supersingular isogenies. In *AFRICACRYPT 2019*, volume 11627 of *Springer LNCS*, pages 56–78, 2019.

[VPN22] VPN by Google one, explained. https://one.google.com/about/vpn/howitworks, 2022. Accessed: 2022-02-02.

[Was08] Lawrence C Washington. *Elliptic curves: number theory and cryptography*. Chapman and Hall/CRC, 2008.

[WW22] Brent Waters and David J Wu. Batch arguments for np and more from standard bilinear group assumptions. In *Advances in Cryptology–CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part II*, pages 433–463. Springer, 2022.

[XY04] Shouhuai Xu and Moti Yung. Accountable ring signatures: A smart card approach. In *Smart Card Research and Advanced Applications VI*, pages 271–286. Springer, 2004.

[XZS22] Tiancheng Xie, Yupeng Zhang, and Dawn Song. Orion: Zero knowledge proof with linear prover time. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 299–328. Springer, Heidelberg, August 2022.

[YAJ$^+$17] Youngho Yoo, Reza Azarderakhsh, Amir Jalali, David Jao, and Vladimir Soukharev. A post-quantum digital signature scheme based on supersingular isogenies. In Aggelos Kiayias, editor, *FC 2017*, volume 10322 of *LNCS*, pages 163–181. Springer, Heidelberg, April 2017.

[YAZ$^+$19] Rupeng Yang, Man Ho Au, Zhenfei Zhang, Qiuliang Xu, Zuoxia Yu, and William Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 147–175. Springer, Heidelberg, August 2019.

[YL19] Xun Yi and Kwok-Yan Lam. A new blind ECDSA scheme for bitcoin transaction anonymity. In Steven D. Galbraith, Giovanni Russello, Willy Susilo, Dieter Gollmann, Engin Kirda, and Zhenkai Liang, editors, *ASIACCS 19*, pages 613–620. ACM Press, July 2019.

[ZCD$^+$20] Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Jonathan Katz, Xiao Wang, Vladmir Kolesnikov, and Daniel Kales. Picnic. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[ZXZS20] Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. Transparent polynomial delegation and its applications to zero knowledge proof. In *2020 IEEE Symposium on Security and Privacy*, pages 859–876. IEEE Computer Society Press, May 2020.