Abstract of thesis entitled

"Trustless Digital Signatures in Blockchain"

Submitted by

Handong Cui

for the degree of Doctor of Philosophy
at The University of Hong Kong
in December, 2023

Trustless in cryptography is an important property which means that there exists no trusted authority in a cryptosystem. Schnorr and ECDSA are two popular trustless signatures since they work in a trapdoorless cyclic group. On the contrary, RSA style signatures, like RSA signature and GQ signature, rely on system parameter containing trapdoor, which cannot be initialized by non-trusted system developer, and hence hinders their application in trustless environment like public blockchain. This thesis considers the weakness of Schnorr and ECDSA in public blockchain and eXpressive Internet Architecture (XIA) and thus proposes an alternative trustless signature; optimizes threshold ECDSA which is also a trustless signature with multiple participants; explores the possibility to reuse GQ signature in a trustless environment.

Blockchain and XIA-like systems have proposed using the hash of a public key as an address, with signatures validated against these addresses. In this context, we aim to define the concept of address-based signatures to scrutinize the security aspects of these address-based systems. We put forth a robust model that takes into account the security of multiple addresses, even in scenarios where attackers even know the randomness employed by system developers. We introduce an effective and secure method for creating address-based signatures that surmount the existing issues of address-based ECDSA (low efficiency, malleability) and Schnorr (lack of BIP-32 compatibility). Additionally, we offer two generic constructions for address-based signatures and deduce that our proposed method always outperforms the implementations of these constructions in Schnorr, ECDSA, BLS/BB signatures, either in efficiency or security.

Next, we move forward to trustless multiparty threshold ECDSA signatures, typically employed for digital wallets or cryptocurrency asset custody. For

most threshold ECDSA signatures that utilize additively homomorphic encryption, the zero-knowledge proofs often become the limiting factor in terms of bandwidth and computational power. As a solution, we introduce a compact zero-knowledge (ZK) proof related to the Castagnos-Laguillaumie (CL) encryption. This new method is 32% more concise in size and 29% quicker in computation than prior work in PKC 2021. Moreover, we present new ZK proofs that relate to homomorphic operations over the CL ciphertext. These new ZK proofs are instrumental in constructing a bandwidth-efficient UC-secure threshold ECDSA that doesn't sacrifice proactive security or non-interactivity.

Lastly, we delve into the Guillou-Quisquater (GQ) signature, a renowned and computationally efficient successor of the Fiat-Shamir follow-ons along with Schnorr. However, the GQ's storage-heavy group element representation and an RSA trapdoor limit its broader application in both industry and academia. We start by formalizing the definition and security proof of the class group-based GQ signature (CL-GQ). This new approach redefines GQ as a trustless signature by eliminating the RSA trapdoor and by enhancing the bandwidth efficiency compared to the original GQ signature. Subsequently, we extend it to a trustless GQ multi-signature scheme, by leveraging non-malleable equivocable commitments and our uniquely designed compact non-interactive zero-knowledge proofs (NIZK). Our scheme demonstrates competitive performance when compared with existing multiparty GQ, Schnorr, and ECDSA.

An abstract of exactly 485 words

Trustless Digital Signatures in Blockchain

by

Handong Cui

Department of Computer Science

The University of Hong Kong

Supervised by

Dr. John T.H. Yuen and Prof. S.M. Yiu

A thesis submitted in partial fulfillment of the requirements for the degree of *Doctor of Philosophy in Computer Science* at *The University of Hong Kong*

December, 2023

Declaration

I declare that this thesis represents my own work, except where due acknowledgement is made, and that it has not been previously included in any thesis, dissertation or report submitted to this university or any institution for any diploma, degree or other qualifications.

(Note: If part of the research work in your thesis has been carried out in collaboration with other parties, please indicate here the extent of collaboration, including jointly published work.)

Signed: 

Handong Cui
December, 2023

Acknowledgements

I would like to thank first my parents for their unconditional love, constant encouragement, and unwavering support throughout my life. Their sacrifices, guidance and support have been instrumental in allowing me to pursue my dreams and achieve my goals. I am immensely grateful for their love and support.

I would like to express my sincere gratitude to my primary supervisor Dr. John Tsz Hon Yuen and my co-supervisor Prof. Siu Ming Yiu for their support, guidance and inspiration throughout my PhD journey. Their supervision, generous help for my research journey have been invaluable in shaping me into a PhD researcher. I am grateful for their mentorship and for the opportunities they have provided me to grow both personally and academically.

I would like to express my gratitude to all the participants in my research work during my PhD studies. Without their willingness to participate, my research work would not have been made possible.

Finally, I am grateful to my colleagues in the Department of Computer Science of HKU, Dr. Xianrui Qin, Dr. Mingli Wu, Dr. Cailing Cai, Kwan Yin Chan, Shimin Pan, Dr. Haiyang Xue, Dr. Yuechen Chen, Dr. Jingxuan Wang, Dr. Jun Zhang, Dr. Meiqi He, Dr. Gongxian Zeng, Dr. Linru Zhang, Dr. Ruoqing Zhang, Dr. Xingye Lu, Dr. Yichen Wei, Dr. Mingshu Cong, Dr. Guowen Yuan, Dr. Yan Zhu, and also grateful to my friends outside HKU CS, for their invaluable support, motivation, and encouragement throughout my PhD studies. Their support and encouragement have made my time during PhD all the more enjoyable and fulfilling.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problems

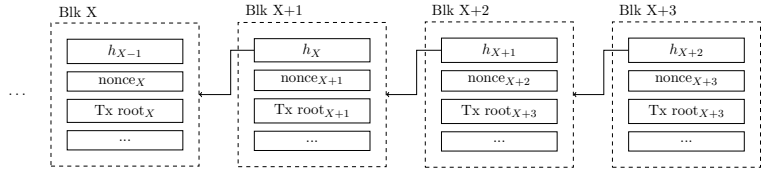
1.1.1 Address in blockchain and XIA

Public Key Infrastructure (PKI) aims to provide secure authentication services for SSL/TLS, Email, VPN, digital signatures, where there exists a vital entity called certificate authority (CA), serving as a trusted third party which issues digital certificates containing public key and other identification information for a certificate holder. In this case, the authenticity of daily used public keys is ensured. However, the removal of trusted third party makes PKI useless in "trustless" environment like public blockchains, Ethereum, Bitcoin, for example. Blockchain is a distributed ledger system created by [61] in 2009, where no trusted authority exists like CA. The name *blockchain* is derived from its data structure as shown in Figure (1.1a). Each block means a consensus made during a specific period, contains a Merkle root, which is computed from the transactions included in each block. As shown in Figure (1.1b). When new consensus is reached through a process called mining, a new block will be appended to the current chain, accordingly leads to a new state of blockchain, which is almost impossible to reverse to the past. Each transaction in a block contains:

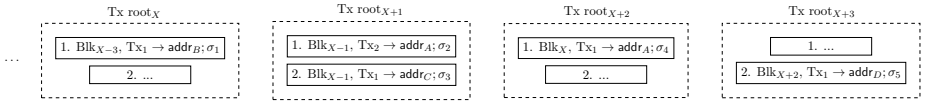
(1) the sequence number of this transaction in this block.

(2) the sender's address referenced by the block number and the sequence number (and also the output number, which is omitted in the figure for simplicity). For example, in Block $X + 2$ transaction 1, the sender is the output address of Block X transaction 1, which is equal to the address addr_B.

(3) the output address, which is the hashing of the recipient's public key.



(A) Chain of data. In each block, it includes the Merkle root of the transaction data below.



(B) Transaction data used to build the Merkle root for each block. For simplicity, we just show one input and one output for each transaction and we omit the transaction amount.

(4) the signature from the sender σ.

In many blockchain systems, public keys are not directly stored on the blockchain. Instead, the hash value of the public key, typically a 160-bit sequence, is used as an *address* on the blockchain. A transaction is accepted if the signature can be verified with the *address*. The concept of *address* is employed because storage on the blockchain is deemed expensive. Transaction fees are proportional to the transaction length, hence the address, rather than the longer public key, represents the transaction recipient on the blockchain. As illustrated in Figure (1.1b), the total cost for Block X transaction 1 is influenced by the size of addr_B and the size of σ_1. Thus, an ideal digital signature for blockchain systems should minimize the sum of address size and signature size, a departure from traditional public key signatures where only the signature size matters. In Bitcoin and Ripple, the ECDSA public key is explicitly revealed along with the signature. Ethereum, conversely, can derive the public key from the ECDSA signature, resulting in a more compact overall transaction data size by using the *address* — a hash of the public key. This ECDSA variant utilized in Ethereum is referred to as an *address-based signature*.

Meanwhile, the *address-based signature* is not exclusive to blockchain but is also implemented in XIA (eXpressive Internet Architecture). In 2006, the U.S. National Science Foundation initiated a future Internet architecture project. Proposed in 2011, the eXpressive Internet Architecture (XIA) [1] is one of three projects entering the Future Internet Architecture-Next Phase (FIA-NP) in 2014. XIA identified the lack of built-in security as a significant issue in today's Internet and extended the self-certifying identifiers [2] to offer intrinsic security for the future Internet.

XIA suggested the use of the public key hash for host and service as an *address* to ensure accountability. As far as the authors are aware, they didn't specify

any signature scheme. In the XIA prototype released on GitHub[1], 1024-bit RSA keys are used, and the *address* is a 160-bit hash output from the RSA key.

XIA network elements [1] include hosts, services, content, users, etc. Network and host principals (NIDs and HIDs) represent autonomous routing domains and hosts that attach to the network. Services represent an application service running on one or more hosts within the network. NIDs, HIDs and SIDs are generated by hashing the public key of an autonomous domain, a host or a service respectively. XIA uses a restricted directed acyclic graph (DAG) representation of XID (including NIDs, HIDs, SIDs, etc.), to specify addresses. Therefore, an *address* in XIA may contain a number of hashed public keys. The DAG of XID is logically equivalent to IP address in today's Internet. Different XIDs are used for different routing modules. In this work, the *address-based signature* of XIA actually refers to the signature with respect to NIDs, HIDs and SIDs in XIA, instead of the DAG of XID (called XIP address in [1]).

Lack of formal security model. To this end, we have seen the power of *address*, which optimizes the transaction cost of blockchain, also brings great compatibility between different hosts using different signature schemes in XIA system. Despite its widespread use, no formal security model currently exists for address-based signatures. We argue that formalizing this concept is crucial, considering that as of April 2020, over USD 170 billion in cryptocurrency is safeguarded by the address-based signature notion. Moreover, the formalization would better accommodate the future Internet XIA.

Drawbacks of ECDSA and Schnorr. ECDSA is the most widely used digital signature scheme in blockchains. However, it suffers the drawbacks of higher signature size and higher computational cost, more difficulties when extending to multisignature, weaker provable security models [15, 38], compared with the efficient Schnorr signature. The reason why the first blockchain system Bitcoin invented in 2009 adopted ECDSA instead of Schnorr is considered usually due to the Schnorr's unexpired patent protection at that time. There has been some explores that replace ECDSA with more efficient Schnorr, like Bitcoin Improvemet Protocol BIP-340. But still the proposal to use Schnorr in Bitcoin is hindered by many reasons. It is only considered as a soft fork which is backward-compatible with older versions of the Bitcoin protocols since nodes running older versions of the software should be still able to participate in Bitcoin network. Nodes include developers, miners, users need a high degree collaboration to take changes of the new fork, which are quite difficult. Therefore, both ECDSA and Schnorr are not always a good or perfect choice for address-based systems like blockchain and XIA due to various kinds of drawbacks.

[1]https://github.com/XIA-Project/xia-core

We claim that both Schnorr and ECDSA, and their *address-based* versions are *trustless signatures* since they works in trapdoorless EC-based cyclic group like secp256k1 which can be deployed or selected by non-trusted system developers. On the contrary, RSA cryptosystem works in an unknown order group containing trapdoor $N = pq$ and it cannot be used as system parameter in a trustless way, namely, where there is no trusted dealer who initializes the system.

It is worth thinking that if there is alternative signature scheme whose performance falls between two existing classical signatures, ECDSA and Schnorr, satisfying the *trustless* property in the meantime, then achieving a "global optimum" so called.

1.1.2 Multiparty signatures: progress and challenges

The concept of multi-signature was proposed in [51]. This joint signing protocol allows a group of signers to collectively generate a compact signature on a shared message, with constant verification time and signature size. In contrast to the multi-signature scheme, threhold signature allows signers to dynamically join a signing group to produce a compact signature and also requires that the participants in a threshold signature are predefined. In threshold signature, a valid signature can only be generated by at least t signers among a total of n participants, usually achieved through a Verifiable Secret Sharing (VSS) approach. Both multi-signature and threshold signature are considered as *multiparty signatures*.

Digital wallets and asset custody are two key applications of multiparty signatures. A digital wallet typically requires its user to divide their secret key across multiple devices and use all (or some) of these devices to transfer the currencies they hold. Asset custody, on the other hand, is a banking service that safeguards a customer's currencies or physical assets. For security reasons, no single entity (be it the bank, customer, or a third-party institution) should have direct access to the secret key, especially when large amounts of currencies are being protected. Therefore, the secret key should also be split into multiple shares.

In 2017, the Multi-sig smart contract (a digital wallet) offered by Parity Technologies was hacked twice and it was estimated that USD 30 million worth of Eth got stolen and USD 280 million worth of Eth got locked permanently. The first instance was due to a bug in Parity's library that allows unauthorized third party to gain ownership of the Multi-sig contract. The second instance was due to the deployed Multi-sig wallet's reliance on the functionalities of a deployed library code. The attacker gains ownership of the deployed library contract and "accidentally" killed the library by calling the

self-destruct method. Hereby the security research of multiparty signature raised many attentions of cryptography researchers. But for the most popular ECDSA and most efficient Schnorr, the research concentration varies. Schnorr is easy to build multiparty signature, the signature can be easily aggregated into one compat signatures. Researchers focus more on the security issues, key aggregation properties. On the contrary, ECDSA cannot be easily build multiparty signature due to its more involved non-linear structure, which requires more tricks like Multiplication to Addition (MtA) protocols including zero-knowledge proofs, which makes the multiparty ECDSA much more costly than multiparty Schnorr, then many research on threshold ECDSA aimed to optimize the performance as the principal goal. In the meantime, more properties like identifiable abort, UC-security, non-interativity are considered later on. But more properties introduced will also lead to more computational cost. Here we briefly review the development of multiparty versions of Schnorr and ECDSA.

Multiparty Schnorr. The efficient Schnorr multi-signature scheme proposed by Bellare and Neven (ACM-CCS 2006 [5]) operates under a *plain public-key model* and allows for the existence of dishonest signers, but it does not support key aggregation. The *plain public-key model* achieves security against rogue-key attack[2] without relying on the KOSK (Knowledge of Secret Key) assumption like [9, 58], thereby reducing some burdensome computation[3]. Maxwell *et. al.* adopted the same *plain public-key model* and proposed a variant of Bellare and Neven's Schnorr multi-signature, known as MuSig, which adds the property of key aggregation [59] (DCC 19). Subsequently, MuSig2 [63] and MuSig-DN [62] were proposed, both of which optimize the round complexity of MuSig from 3 rounds to 2 rounds. However, MuSig and MuSig2 have considerable reduction loss caused by a double-forking technique [59]. MuSig-DN achieves deterministic signing at the cost of expensive zero-knowledge proofs. All of the above schemes cannot achieve identifiable abort, as there are no checks on the correctness of either R_i or s_i. They can achieve identifiable abort by adding an additional check $g^{s_i} = R_i X_i^{c_i}$ without any zero-knowledge proofs.

Multiparty ECDSA. Lindell et. al. proposed the first practical full threshold ECDSA (ACM-CCS 2018 [55]) and a parallel work by *Gennaro et. al.* introduced the first efficient threshold ECDSA construction relying on a game-based security proof (ACM-CCS 2018 [42]). There has been an abundance of follow-up work [20, 23, 36, 41, 43, 78] to improve these two schemes and

[2]A rogue-key attack refers to an adversary being able to forge multi-signature by arbitrarily choosing his public key, or using a function of the public keys of honest signers.

[3]While the KOSK effectively resists rogue-key attack, it requires the proof of knowledge of secret key when mounting attacks by submitting corresponding public keys, which incurs expensive computation.

notable advancements have been made in various aspects, such as waiving expensive range proofs, lowering the signing rounds, and adding the identifiable abort functionality. All the mentioned threshold ECDSA schemes operate in the *dishonest majority model*, which is much stronger than the *plain public-key model*, especially for decentralized and trustless settings. Gennaro and Goldfeder's scheme [43] achieves identifiable abort, which is attributed to a specific phase. A bandwidth-efficient threshold ECDSA scheme based on class groups was proposed by [23], which first adopted class groups as their homomorphic encryption and waived the expensive range proof of Paillier public keys, thus achieving great bandwidth. Soon after, a threshold ECDSA using the CL encryption was proposed by Castagnos *et. al.* [23], with new properties of non-interactive signing and identifiable abort. However, it cannot achieve UC security. In terms of theoretical complexity, the scheme from [23] reduces the bandwidth consumption of the Paillier-based threshold ECDSA [20] by up to a factor of 10. However, [23] requires an expensive interactive setup as [23]. Another threshold ECDSA using CL encryption was proposed by Deng *et. al.* [33], which introduces a weaker soundness called *promise extractability* and waives the interactive setup. However, they cannot achieve non-interactive signing and proactive security.

1.1.3 Zero-knowledge proofs for class group encryption in threshold ECDSA

From above literature review for threshold ECDSA, we find interestingly the research work building up ECDSA mainly fall into two categories, one is Paillier-based threshold ECDSA [20, 41–43, 55], one is class group based threshold ECDSA [23, 23, 33, 78]. Paillier based scheme is computationally efficient but bandwidth unfriendly, but class group based scheme is conversely computationally costly but bandwidth efficient. In the decentralized blockchain world, the transaction shall be online and storage issue is more important than computational cost, in which sense, class group based schemes seem more appealing to a blockchain era. There is also another OT-based threshold ECDSA like [36], but it is less comparable with Paillier-based and class group-based schemes since it does not adopt a homomorphic encryption fashion to build up MtA. We confess that it is faster than the other two, but at a cost of extremely demanding bandwidth condition.[4]

One-bit challenge problem in ZK Proofs. Dating back to the first time that class group, more specifically, CL-encryption was applied in two-party

[4]The non-linear structure of threshold ECDSA requires the use of Multiplication to Addition protocol, where there exists the additively homomorphic encryption. Paillier encryption and class group encryption are two outstanding schemes which have different advantages respectively. OT is also one way to build MtA.

ECDSA in 2019 [22], it suffers a low-speed zero-knowledge proof for CL-encryption correctness which works in the unknown order cyclic group – class group, in that it has to adopt a one-bit challenge fashion to make the soundness extractor possible, but at a cost that it has to repeat for λ_s times for a soundness error of $2^{-\lambda_s}$. This limitation also affects the later threshold version of class group based ECDSA [23]. Although a *lowest common multiple* trick in [23] was proposed to reduce the proof size of [22] by 10 times, it actually proves a more loosen ZK relationship.

Very first trial to bypass one-bit challenge. Now we discuss the possibility to bypass the inefficient one-bit challenge fashion in zero-knowledge proof which is non-trivial. Following the settings in the CL encryption [26], we use an unknown order group G, which contains a subgroup F in which the DL problem is tractable. Consider the argument of knowledge for a simple DL relation $\mathcal{R}$ in G for some group elements $g, w \in G \setminus F$: $\mathcal{R} = \{x \in \mathbb{Z} : w = g^x\}$. For the DL relation in an unknown order group, Boneh *et al.* [11] proposed to use a random prime ℓ as a challenge, and the prover computes d and $e \in [0, \ell - 1]$ such that $x = d\ell + e$. The prover sends $D = g^d$ and e to the verifier to check if $D^\ell g^e = w$. Yuen *et al.* [78] showed that this method does not work if G has a known order subgroup F with a generator f. One possible attack is that $w = g^x f^y$ for some $x, y \in \mathbb{Z}$. The adversary can set $D' = g^d f^{y/\ell}$. The value (D', e) can also pass the verification. Therefore, they proposed another round of challenges using the prime q, the order of F. The prover additionally needs to compute r and $s \in [0, q - 1]$ such that $x = rq + s$. The prover also sends $R = g^r$ and s to the verifier to check if $R^q g^s = w$. This checking eliminates the possibility of having some order q element in w, at a cost of almost doubling the proof size and the running time in the original scheme [11]. Under this technique, Yuen *et al.* [78] proposed a compact one-pass ZK proof without repetition for the (generalized) DL relation which further reduced the proof size of [23] by around 4 times. They also create the ZK proof for the well-formedness of a CL ciphertext is also used in the signing phase of two-party ECDSA [22] and threshold ECDSA [23].

Outlook for Yuen's one-pass ZK fashion. However, the above-mentioned Yuen's ZK method incurs a higher round complexity. It is worth thinking that how to lower further its round complexity to make the proof more efficient when we consider round trip costs. Moreover, it is also interesting how Yuen's ZK proofs can be applied in more scenarios not covered in their research paper [78], like converting the UC-secure, proactive and non-interactive threshold ECDSA [20] from Paillier version to a bandwidth efficient class group version, to see what kind trade-off can CL-encryption bring to us.

1.1.4 A broader view: make GQ possible again

The Guillou-Quisquater signature, or GQ signature for short, was put forward by Guillou and Quisquater in 1988 [46]. Alongside the Schnorr signature [68], the GQ signature scheme is recognized as one of the most efficient and renowned successors to the Fiat-Shamir heuristic [39]. Schnorr relies on discrete logarithm system parameter, uses generator g in group G as the base, and puts a secret integer in Z_q (secret key x or randomness r) onto the exponent where q is the order of G, then forms the public key $X = g^x$. Then follows a commit g^r, challenge c computed by hashing the previous message and response $z = r + cx$ fashion to produce a valid signature (c, z) for verification. Very similarly, GQ relies on an RSA group as the system parameter where we have public N and secret primes p, q s.t. $N = pq$, uses integer v satisfying $0 < v < \phi(N)$ and v is co-prime to $\phi(N)$ as the public parameter like the g in Schnorr system. Signer chooses a group element B from Z_N as her secret key, and publishes the related public key $J = B^{-v}$. In the first round GQ signer also makes a commit $T = r^v$ where r is also drawn from Z_N and kept secret. For the second round, GQ signer computes the challenge h by hashing the previous messages. Finally, she outputs a valid signature (t, h), where $t = rB^h$ for verification. Both signatures follow a classical three-move fashion to finish the signing.

Drawbacks of GQ signature. At first glance, GQ is highly efficient. Also, it is as easy as Schnorr to build multi-signature since the signature t is easy to aggregate, in which sense it obviously outperforms ECDSA. But what are the reasons that GQ's application scenarios and research discussions are rather limited nowadays? One outstanding issue is that GQ is RSA based, hence when it is enxtended to a cryptosystem including multiple parties, it usually requires a trusted authority to set up the system parameter $N = pq$ and keep p, q from any disclosure[5], the requirement of which is also similar to RSA accumulator which also requires a trusted setup. This is prohibitive for the adoption of GQ in a trustless environment, like a public blockchain. Another important reason is that the signature size is much larger than ECDSA and Schnorr since the group element of RSA group itself is quite cumbersome. For example, the RSA group element has to be represented by a 3072-bit string (induced from N's length requirement) for 128-bit security. On the contrary, Schnorr and ECDSA built upon EC only require 256 bits to represent an EC element. To wrap up, GQ is mainly hindered by its trusted setup requirement and cumbersome signature size.

Make GQ trustless. For the first issue, actually in 2000, Hamdy and Möller mentioned in their paper [47] that class groups [18] can be used to replace RSA

[5]Note that the q here is simply a prime consistent throught Chapter 5, but it is not the same q which represents order or EC cyclic group covered in Chapter 3 and 4.

group in GQ signature and hence makes GQ can be set up without a trapdoor like p, q. However, to formally construct such a trustless GQ signature which can be applied in blockchain or digital wallet, it still lacks a formal definition and a rigorous security proof, its hardness assumptions are also worth further discussion. For the second issue, by intuition, class group will become a better alternative for RSA group since class group uses shorter representation, which makes it more practical in blockchain where we focus more on storage issues than computation issues. In other words, lower signature size costs lower transaction fees.

Extend trustless GQ to multiparty scenario. Moreover, good aggregation property makes GQ a good choice to construct multi-signature, which can be a competitive candidate when building up digital wallets and asset custody. The most recent multi-signature (identity-based) of GQ we can trace back is Bellare and Neven's work in CT-RSA 2006 [6]. While the GQ signature excels in computational efficiency and has been proven secure via the forking lemma, it relies on a delicate security model that assumes all signers to be *honest*. This assumption falls flat in real-world scenarios where dishonest adversaries are present, rendering the model impractical. Hence, it is interesting to see how we make trustless multiparty GQ signature work in a highly malicious environment where every participant can behave maliciously, in which scenario it is essential to allow honest signers to detect and reject incorrect/malicious messages from other participants.

1.2 Main Contributions

The main contributions are as follows.

- For the first time, we formalize the concept of address-based signature, which captures the signature systems in XIA and blockchain, and we develop a strong security model concerning to the security of multiple addresses and system initializer's attack.
- Concerning the disadvantages of ECDSA like malleability, weak provable security model, low efficiency, difficulties in building up multisignature, and concerning the attack in BIP-32 for Schnorr, we create a novel "globally optimum" address-based signature scheme. It solves these problems of address-based ECDSA and Schnorr, and its security is reduced to the discrete logarithm hardness assumption in random oracle model.
- We build up two generic constructions under our definition of address-based signature. We evaluate many instantiations, like ECDSA, Schnorr,

BLS/BB. We draw the conclusion that they are always worse than our proposed address-based signature scheme.

- We optimize the existing zero-knowledge proofs for discrete logarithm and well-formedness of a CL ciphertext in [78]. We lower the round complexity by removing one round challenge. The proof size is reduced by at least 332% and the computational cost is reduced by at least 29%.

- A frequently used zero-knowledge proof in threshold ECDSA with identifiable abort [43] and UC-secure threshold ECDSA [21] is designed to demonstrate the knowledge of (A, B), such that the plaintext m within a CL ciphertext transforms to a ciphertext of $Am + B$. This type of association is referred to as the *affine transformation*, where the prover lacks knowledge of m. We have devised a new zero-knowledge proof for the affine transformation over the CL ciphertext, an area not addressed by the existing study [78].

- We obatin the sampling bounds of the above two types of ZK proofs in the process of proving their honest verifier zero knowledge property and use these ZK proofs to construct the most bandwidth-efficient UC-secure threshold ECDSA. As a competitive class group based counterpart of [21], we lowered the communication and computation cost of the key refresh algorithm in from $O(n^3)$ to $O(n^2)$.

- For the first time, we present formal definition for class group based GQ signature and we name it CL-GQ. It is proved secure against existential unforgeability under chosen message attack in the random oracle model, with an additional prime root assumption. It is beneficial for understanding the details of upgrading GQ to a trustless CL-GQ which is early mentioned in [47] in 2000.

- We design a compact zero-knowledge proof for proving the knowledge of a base.[6] We avoid the explicit way of using one-bit challenge fashion like the ZK proof for Paillier ciphertext in [77], which is easy to build the soundness extractor but at a cost of many times of repetition for an acceptable soundess. Instead, we find Bezout trick useful to eliminate the need of one-bit challenge, and thus achieve very efficient one-pass ZK proof. We also extend such a ZK proof to proving the well-formedness of a CL-GQ signature.

- We extend our CL-GQ to its multi-signature version. Using the well-designed ZK proofs for RSA-style relation (knowledge/witness is in the base), our CL-GQ multi-signature can achieve identifiable abort property in a highly trustless dishonest majority model.

[6] For an exponentiation, in Schnorr or ECDSA, the secret value is put onto the exponent and the base is fixed and public; on the contrary, the secret value is put into the base but exponent is a public parameter.

1.3 Thesis Organization

The rest of the thesis is organized as follows.

Chapter 2 overviews the background knowledge required for this thesis. Chapter 3 introduces the address-based signature, including its formal definition, security model, generic construction of address-based signatures, and our proposed novel scheme which is a well-rounded address-based signature. Chapter 4 bandwidth-efficient zero-knowledge proofs for threshold ECDSA, including a novel ZK fashion to prove the knowledge of a discrete logarithm in an unknown order group, and the application of our ZK proofs: a bandwidth-optimal UC-secure threshold ECDSA. Chapter 5 introduces a trapdoorless GQ and its multi-signature version which can work in a highly malicious and trustless environment with an identifiable abort property by using compact ZK proofs. Chapter 6 summarizes the research results of this thesis.

Chapter 2

Background

Notations. We denote d being drawn from a distribution $\mathcal{D}$ and b being randomly selected from the set B as $d \leftarrow \mathcal{D}$ and $b \xleftarrow{\$} B$, respectively. A negligible (resp. exponential) function is denoted as $\mathsf{negl}(\lambda)$ (resp. $\mathsf{exp}(\lambda)$). We use $\mathsf{ord}_{\mathbb{G}}(g)$, ϵ_s and ϵ_d to denote the order of $g \in \mathbb{G}$, the parameter for soundness error and statistical distance respectively. Furthermore, we utilize a group where the *hard subgroup membership* assumption [22] is valid. Let $\mathcal{D}$ (resp. $\mathcal{D}_q$) represent a distribution over the integers. From the uniform distribution in G (resp. G^q), the distribution over $\{g^x, x \xleftarrow{\$} \mathcal{D}\}$ (resp. $\{g_q^x, x \xleftarrow{\$} \mathcal{D}_q\}$) is at a distance less than 2^{λ}.

2.1 Discrete Logarithm Assumption

Adopting the notation from [66], for a security parameter 1^{λ}, a cyclic group $\mathbb{G}$, its order p, and the generator g are categorized as the algebraic structure SI of the discrete logarithm (DL) problem instance. The DL assumption is valid if no probabilistic polynomial-time (PPT) adversary can produce x when given $X = g^x$ and SI, where x is randomly selected from $\mathbb{Z}_p$.

2.2 Hash Functions

We revisit some properties of hash functions based on [65]. We consider a hash function H as a family of functions, $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$.

The *always second-preimage resistant* property is reviewed as follows. Let m be a number such that $\{0,1\}^m \subseteq M$. Let $\mathcal{A}$ be an adversary. Then a family of hash functions is deemed *always second-preimage resistant* if the probability:

$$\max_{K \in \mathcal{K}} \left\{ \Pr \left[\begin{matrix} M \leftarrow_R \{0,1\}^m; M' \leftarrow \mathcal{A}(M) : \\ (M \neq M') \wedge (H_K(M) = H_K(M')) \end{matrix} \right] \right\}$$

is negligible.

We review the *collision resistance* property as follows. A hash function family is *collision resistance* if the probability:

$$\Pr \left[\begin{matrix} K \leftarrow_R \mathcal{K}; (M, M') \leftarrow \mathcal{A}(K) : \\ (M \neq M') \wedge (H_K(M) = H_K(M')) \end{matrix} \right]$$

is negligible.

2.3 RKA and Strong Known RKA Models

We recall Φ-EUF-CM-RKA and Φ-EUF-CM-sKRKA, which are respectively the existential unforgeability security models under chosen messages against RKA [4] and sKRKA [79], where $\Phi^+ = \{\phi_i(x) = x + b_i : b_i \in \mathcal{S}\}$, $\Phi^* = \{\phi_i(x) = x * a_i : a_i \in \mathcal{S}\}$ and $\Phi^{\mathsf{aff}} = \{\phi_i(x) = a_i x + b_i : a_i, b_i \in \mathcal{S}\}$.

Algorithm 1: Game Φ-EUF-CM-RKA.

Procedure $\textsc{Init}(1^\lambda)$**:**
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow_s \mathsf{KeyGen}(1^\lambda)$;
  $\mathbb{L} \leftarrow \emptyset$;
  return pk;
Procedure $\textsc{Sign}(m_i, \phi_i)$**:**
  if $\phi_i \notin \{\Phi \cup$ *identity map*$\}$ **then**
    return $\perp$;
  $\sigma_i \leftarrow_s \mathsf{Sign}(\phi_i(\mathsf{sk}), m_i)$;
  if ϕ_i *is identity map or* $\phi_i(\mathsf{sk}) = \mathsf{sk}$ **then**
    $\mathbb{L} \leftarrow \mathbb{L} \cup \{m_i\}$;
  return σ_i;
Procedure $\textsc{Fin}(m^*, \sigma^*)$**:**
  if $m^* \in \mathbb{L}$ **then**
    stop with 0;
  if $\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = 0$ **then**
    stop with 0;
  stop with 1;

Algorithm 2: Game Φ-EUF-CM-sKRKA.

Procedure $\textsc{Init}(1^\lambda)$**:**
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow_s \mathsf{KeyGen}(1^\lambda)$;
  $\mathbb{L} \leftarrow \emptyset$;
  $\phi_0 \leftarrow$ identity map;
  $\mathbb{S} \leftarrow \{\phi_0\}$;
  for $j \leftarrow 1$ **to** q_s **do**
    $\phi_j \leftarrow_s \Phi$;
    $\mathbb{S} \leftarrow \mathbb{S} \cup \{\phi_j\}$;
  return pk, $\mathbb{S}$;
Procedure $\textsc{Sign}(m_i, j)$**:**
  if $j \notin [0, q_s]$ **then**
    return $\perp$;
  $\sigma_i \leftarrow_s \mathsf{Sign}(\phi_j(\mathsf{sk}), m_i)$;
  $\mathsf{pk}_i \leftarrow \mathcal{T}(1^\lambda, \phi_j(\mathsf{sk}))$;
  $\mathbb{L} \leftarrow \mathbb{L} \cup \{(\mathsf{pk}_i, m_i)\}$;
  return $(\mathsf{pk}_i, \sigma_i)$;
Procedure $\textsc{Fin}(i^*, m^*, \sigma^*)$**:**
  if $i^* \notin [0, q_s]$ **then**
    stop with 0;
  if $(\mathsf{pk}_{i^*}, m^*) \in \mathbb{L}$ **then**
    stop with 0;
  if $\mathsf{Verify}(\mathsf{pk}_{i^*}, m^*, \sigma^*) = 0$ **then**
    stop with 0;
  stop with 1;

Definition 2.1. A signature scheme is (t, q_s, ϵ)-secure under the Φ-EUF-CM-RKA (resp. Φ-EUF-CM-sKRKA) if there is no adversary running in time t, with q_s queries to the signing oracle, has advantage larger than ϵ in Game Φ-EUF-CM-RKA (resp. Φ-EUF-CM-sKRKA).

Relations between RKA and sKRKA. The relationship between the RKA model and the Strong KRKA model which is an open problem and there is no straightforward implication from one to another. It is proved that Schnorr is neither secure in RKA nor sKRKA ; ECDSA is not RKA secure but sKRKA secure [60, 79].

2.4 ECC and HSM Groups

We remark some group generation algorithms as in [22]:

- The $\mathsf{GGen}_{\mathsf{ECC}}$ algorithm generates a cyclic group $\hat{G}$ with prime order q, and returns $\mathcal{G}_{\mathsf{ECC}} = (\hat{G}, q, \hat{P})$, where $\hat{P}$ is a generator of $\hat{G}$. Note that the security parameter input is 1^λ.

- The $\mathsf{GGen}_{\mathsf{HSM}}$ algorithm returns $\mathcal{G}_{\mathsf{HSM}} = (\tilde{s}, g, f, g_q, \tilde{G}, G, F, G^q)$ with the inputs of a prime number q and a security parameter 1^λ.

  The finite abelian group $(\tilde{G}, \cdot)$ is of order $q \cdot \hat{s}$, where the length of $\hat{s}$ is a function of λ and $\mathsf{gcd}(q, \hat{s}) = 1$. The value of the upper bound of $\hat{s}$ is $\tilde{s}$, any element can be decided in polynomial time if it is in $\tilde{G}$. The set $(F, \cdot)$ is the unique cyclic subgroup of $\tilde{G}$ of order q, generated by f. The group generated by g_q is a subgroup of G of order s is denoted as $G^q := \{x^q, x \in G\}$. The cyclic subgroup of $\tilde{G}$ of order $q \cdot s$ is denoted as $(G, \cdot)$, where s divides $\hat{s}$. With the construction of $F \subset G$, it holds $G = G^q \times F$ and $g := f \cdot g_q$ is the generator of G. A polynomial time algorithm Solve solves the discrete logarithm problem in F:

  $$x \leftarrow \mathsf{Solve}_{\mathcal{G}_{\mathsf{HSM},\mathsf{q}}}(f^x), \forall x \xleftarrow{\$} \mathbb{Z}_q.$$

  We call this *HSM group*, and drop the subscript for Solve in the following paragraphs for simplicity.

2.5 Class Group of Imaginary Quadratic Field

Let $-\Delta$ be a random (large) λ-bit prime such that $\Delta \equiv 1 \bmod 4$. The ring $\mathcal{O}_\Delta = \mathbb{Z} + \frac{\Delta+\sqrt{\Delta}}{2}\mathbb{Z}$ is an imaginary quadratic order of discriminant Δ. Its field of fractions is $\mathcal{Q}(\sqrt{\Delta})$. The fractional ideals of $\mathcal{O}_\Delta$ are of the form $q(a\mathbb{Z} + \frac{b+\sqrt{\Delta}}{2}\mathbb{Z})$ with $q \in \mathcal{Q}, \alpha \in \mathbb{Z}^+, b \in \mathbb{Z}$ and $4a|(b^2 - \Delta)$. An ideal is integral if $q = 1$, and it can be represented by a pair (a, b). Two factional ideals $\mathfrak{a}, \mathfrak{b} \in \mathcal{O}_\Delta$ are equivalent if for some non-zero $\alpha \in \mathcal{Q}(\sqrt{\Delta}), \mathfrak{a} = \alpha\mathfrak{b}$. The set of equivalence classes form an Abelian group under ideal multiplication, which is known as the class group of imaginary quadratic order $\mathrm{CL}(\Delta)$. Sometimes we denote the group as D_i, where $i = -\Delta$. One set of equivalence classes can be represented by a unique (a, b) form through a reduction algorithm satisfying that $gcd(a, b, c) = 1, -a < b \leq a \leq c$,and $b \geq 0$ if $a = c$. The class group of imaginary quadratic order D_i is an Abelian group with ideal multiplication.

Meanwhile, class group is always finite and the group order is unknown. More description can be found in [47, 48].

The HSM group can be instantiated by class groups of imaginary quadratic order.

The $\mathsf{GGen}_{\mathsf{HSM}}$ algorithm computes $\Delta_K = -q\tilde{q}$ and $\Delta_q = q^2\Delta_K$, where $\tilde{q}$ is a random prime such that $q\tilde{q} \equiv 1 \pmod 4$ and $(q/\tilde{q}) = -1$. Denote $\tilde{G}$ as the class group $Cl(\Delta_q)$ with order of $h(\Delta_q) = q \cdot h(\Delta_K)$. It computes $\tilde{s} := \left\lceil \frac{1}{\pi} \log |\Delta_K| \sqrt{|\Delta_K|} \right\rceil$ and thus $h(\Delta_K) < \tilde{s}$.

$\mathsf{GGen}_{\mathsf{HSM}}$ requires $F = \langle f \rangle$, where $f = [(q, q^2)] \in Cl(\Delta_q)$. It takes a small prime r, where $r \neq q$ and $(\frac{\Delta_K}{r}) = 1$, and sets an ideal lying above r as I. The algorithm computes $g_q = [\varphi_q^{-1}(I^2)]^q \in Cl(\Delta_q)$ and sets $G^q = \langle g_q \rangle$, where φ^{-1} is the surjection defined in the Algorithm 1 of [25]. It finally computes $g = f \cdot g_q$, sets $G = \langle g \rangle$, and outputs $\mathcal{G}_{\mathsf{HSM}} = (\tilde{s}, g, f, g_q, \tilde{G}, G, F, G^q)$.

2.6 Castagnos and Laguillaumie (CL) Encryption from HSM Group

A framework of a group with HSM was introduced in [26] for an easy DL subgroup. The CL Encryption algorithm [22] from the class group of quadratic fields is composed of six parts: Setup, Key Generation (KeyGen), Encryption (Encrypt), Decryption (Decrypt), Scalar Evaluation (EvalScal), and Sum Evaluation (EvalSum). Let's review them:

1. **Setup**: Given a prime p and a security parameter 1^λ, it executes $\mathcal{G}_{\mathsf{HSM}} \leftarrow \mathsf{GGen}_{\mathsf{HSM},\mathsf{q}}(1^\lambda)$ and parses $\mathcal{G}_{\mathsf{HSM}} = (\tilde{s}, g, f, g_q, \tilde{G}, G, F, G^q)$. It then outputs $\mathsf{param} = \mathcal{G}_{\mathsf{HSM}}$. Set $S = \tilde{s} \cdot 2^{\epsilon_d}$ where ϵ_d are some statistical distance. The input param is omitted for simplicity.

2. **Key Generation (KeyGen)**: It returns a pair of keys $(\mathsf{sk}, \mathsf{pk})$, where sk is randomly chosen from the range $[0, S]$ and $\mathsf{pk} = g_q^{\mathsf{sk}}$.

3. **Encryption (Encrypt)**: Given a message m and a public key pk, it returns the ciphertext $C = (C_1, C_2)$ with a random value ρ chosen from the range $[0, S]$. The ciphertext components are computed as $C_1 = f^m \mathsf{pk}^\rho$ and $C_2 = g_q^\rho$.

4. **Decryption (Decrypt)**: Given a ciphertext $C = (C_1, C_2)$ and a secret key sk, it computes $M = C_1 / C_2^{\mathsf{sk}}$ and solves for m using the $\mathsf{Solve}(M)$ operation.

5. **Scalar Evaluation (EvalScal)**: Given a scalar s, a ciphertext $C = (C_1, C_2)$, and a public key pk, it returns a new ciphertext $C' = (C'_1 = C_1^s, C'_2 = C_2^s)$.

6. **Sum Evaluation (EvalSum)**: Given two ciphertexts $C = (C_1, C_2)$, $C' = (C'_1, C'_2)$ and a public key pk, it returns a new ciphertext $\hat{C} = (\hat{C}_1 = C_1 C'_1, \hat{C}_2 = C_2 C'_2)$.

These components allow the CL Encryption scheme to provide homomorphic properties over the class group of quadratic fields, enabling computations on encrypted data.

2.7 Generic Group Model for HSM Group

The HSM group is modelled by the generic group model for groups of unknown order [28] together with groups of known order. Yuen *et al.* [78] generalized the generic group model for HSM Group, and they are reviewed as follows:

A group $\mathbb{G} = \mathbb{G}_1 \times \mathbb{G}_2$ is parameterized by three integer public parameters q, A, B. $\mathbb{G}$ is defined by a random injective function σ: $\mathbb{Z}_{|\mathbb{G}_1| \times q} \to \{0,1\}^\ell$ for some ℓ where $2^\ell \gg |\mathbb{G}_1| \times q$. The order of $\mathbb{G}_1$ is sampled uniformly from $[A, B]$ and the order of $\mathbb{G}_2$ is q. The group elements are $\sigma(0), \sigma(1), ..., \sigma(|\mathbb{G}_1| \times q - 1)$.

A generic algorithm $\mathcal{A}$ is a probabilistic algorithm that takes $(q, \mathcal{L})$ as input, where $\mathcal{L} = \mathcal{L}_0 \cup \mathcal{L}_1$ is a list that is initialized with the encodings. We further defined a function $\pi(a, b) = qa + b$ for $a \in \mathbb{Z}_{|\mathbb{G}_1|}$ and $b \in \mathbb{Z}_q$. $\mathcal{A}$ queries two generic group oracles:

- $\mathcal{O}_1$ takes $b' \leftarrow \{0,1\}$. If $b' = 0$, it samples a random $a \in \mathbb{Z}_{|\mathbb{G}_1|}$, $b \in \mathbb{Z}_q$ and returns $\sigma(\pi(a, b))$, which is appended to $\mathcal{L}_0$. If $b' = 1$, it samples a random $b \in \mathbb{Z}_q$ and returns $\sigma(\pi(0, b))$, which is appended to $\mathcal{L}_1$.
- When $\mathcal{L}$ has size $\tilde{q}$, $\mathcal{O}_2(i, j, \pm)$ takes $i, j \in [1, \tilde{q}]$ as indices and a sign bit, and returns $\sigma(\pi(a_i \pm a_j \bmod |\mathbb{G}_1|, b_i \pm b_j \bmod q))$, which is appended to $\mathcal{L}_1$ if $a_i \pm a_j \neq 0 \bmod |\mathbb{G}_1|$. Otherwise, it is appended to $\mathcal{L}_0$.

This model treats the output of $\mathcal{O}_1(1)$ as the elements in F and the output of $\mathcal{O}_1(0)$ as the element in G for the group $\mathcal{G}_{\mathsf{HSM}}$. For some random a, the generator g_q in G^q is initialized as $\sigma(\pi(a, 0))$. It is difficult to distinguish whether it is in G^q, given the output of $\mathcal{O}_1(0)$. Suppose that for some $b^* \in \mathbb{Z}_q$, f is initialized as $\sigma(\pi(0, b^*))$. For input $\tilde{f} \in F$, the Solve algorithm can be modelled by finding the encoding of $\tilde{f}$ in $\mathcal{L}_1$ as $\sigma(\pi(0, \tilde{b}))$ for some $\tilde{b} \in \mathbb{Z}_q$ and returning $\tilde{b}/b^* \bmod q$. We recap two related lemmas from [78].

Lemma 2.2 (Subgroup Element Representation)**.** *Let $\mathbb{G}$ be a generic group and $\mathcal{A}$ be a generic algorithm making q_1 queries to $\mathcal{O}_1$ and q_2 queries to $\mathcal{O}_2$. Let $\{g_1, ..., g_{m_0}\}$ be the outputs of $\mathcal{O}_1(0)$. There is an efficient algorithm* Ext *that given as input the transcript of $\mathcal{A}$'s interaction wit the generic group oracles, produces for every element $u \in \mathbb{G}$ that $\mathcal{A}$ outputs, a tuple $(\alpha_1, ..., \alpha_{m_0}) \in \mathbb{Z}^m$ and $\gamma \in \mathbb{Z}_q$ such that $u = f^\gamma \cdot \prod_{i=1}^{m_0} g_i^{\alpha_i}$ and $\alpha_i \leq 2^{q_2}$.*

Lemma 2.3 (Subgroup Discrete Logarithm)**.** *Let $\mathbb{G} = \mathbb{G}_1 \times \mathbb{G}_2$ be a generic group where $|\mathbb{G}_1|$ is a uniformly chosen integer in $[A, B]$ and $1/A$ and $1/|B-A|$ are negligible in λ. Let $\mathcal{A}$ be a polynomial time generic algorithm and let $g_1, ..., g_{m_0}$ be the outputs of $\mathcal{O}_1(0)$. The probability that $\mathcal{A}$ succeeds in outputting $\alpha_1, ..., \alpha_{m_0}, \beta_1, ..., \beta_{m_0} \in \mathbb{Z}$ and $\gamma, \delta \in \mathbb{Z}_q$, such that $f^\gamma \cdot \prod_{i=1}^{m_0} g_i^{\alpha_i} = f^\delta \cdot \prod_{i=1}^{m_0} g_i^{\beta_i} \in \mathbb{G}$, $\alpha_i \neq \beta_i$ and $\gamma \neq \delta$ mod q, is negligible.*

2.8 Hard Subgroup Membership Assumption

The hard subgroup membership assumption for the group $\mathcal{G}_{\mathsf{HSM}}$ means that is difficult to identify the elements of G^q in G. For every polynomial time algorithm $\mathcal{A}$:

$$\left| Pr\left[b = b^* \middle| \begin{array}{l} \mathcal{G}_{\mathsf{HSM}} \leftarrow \mathsf{GGen}_{\mathsf{HSM},\mathsf{q}}(1^\lambda), \\ x \hookleftarrow \mathcal{D}, x' \hookleftarrow \mathcal{D}_q, b \xleftarrow{\$} \{0,1\}, \\ Z_0 = g^x, Z_1 = g_q^{x'}, \\ b^* \leftarrow \mathcal{A}(\mathcal{G}_{\mathsf{HSM}}, Z_b, \mathsf{Solve}(\cdot)) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

2.9 Adaptive Root Subgroup Assumption.

The adaptive root subgroup assumption is the modification of the adaptive root assumption [11] in the group $\mathcal{G}_{\mathsf{HSM}}$. Denote $\mathsf{Primes}(\lambda)$ as the set of odd primes less than 2^λ.

The adaptive root subgroup assumption holds for the group $\mathcal{G}_{\mathsf{HSM}}$ if for all polynomial time algorithms $(\mathcal{A}_0, \mathcal{A}_1)$:

$$Pr\left[\begin{array}{l} u^\ell = w, \\ w^q \neq 1 \end{array} \middle| \begin{array}{l} q > 2^\lambda, \mathcal{G}_{\mathsf{HSM}} \leftarrow \mathsf{GGen}_{\mathsf{HSM},\mathsf{q}}(1^\lambda), \\ (w, \mathsf{state}) \leftarrow \mathcal{A}_0(\mathcal{G}_{\mathsf{HSM}}), \\ \ell \xleftarrow{\$} \mathsf{Primes}(\lambda), u \leftarrow \mathcal{A}_1(\ell, \mathsf{state}) \end{array} \right] \leq \mathsf{negl}(\lambda).$$

Yuen *et al.* [78] shows the intractability of the adaptive root subgroup problem and the non-trivial order element problem in a generic group model.

Corollary 2.4. *(Adaptive Root Subgroup Hardness). Let $G \in \mathcal{G}_{\mathsf{HSM}}$ be a generic group where $|\mathcal{G}^q|$ is a uniformly chosen integer in $[A, B]$ such that $1/A$ and $1/|B - A|$ are negligible in λ. Any generic adversary $\mathcal{A}$ that performs a polynomial number of queries to oracle $\mathcal{O}_2$ succeeds in breaking the adaptive root subgroup assumption on $\mathcal{G}_{\mathsf{HSM}}$ with at most negligible probability in λ.*

Corollary 2.5. *(Non-trivial order hardness). Let $G \in \mathcal{G}_{\mathsf{HSM}}$ be a generic group where $|G^q|$ is a uniformly chosen integer in $[A, B]$ such that $1/A$ and $1/|B-A|$ are negligible in λ. Any generic adversary $\mathcal{A}$ that performs a polynomial number of queries to oracle $\mathcal{O}_2$ succeeds in finding an element $h \neq 1 \in G$ and a positive integer d such that $h^d = 1$ and $d < q$ with at most negligible probability in λ.*

2.10 Digital Signature Scheme

A digital signature scheme $\mathcal{S}$ is comprised of three algorithms: $\{\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}\}$.

1. The randomized key generation algorithm, KeyGen, processes global information $\mathcal{I}$ to yield a pair $(\mathsf{sk}, \mathsf{pk})$, which comprises a secret key and a public key. The global information might include details such as a security parameter, a description of the group and its generator, and the hash function's description. We disregard the origin of these parameters, assuming they are readily accessible to the public.

2. The signature generation algorithm, Sign, which can be randomized, accepts a message M for signing, the global information $\mathcal{I}$, and a secret key sk. It produces M along with a signature σ.

3. The deterministic verification algorithm, Verify, processes a public key pk, a message M, and a signature σ. It returns 1 (accept) if the signature is valid, and 0 (reject) otherwise.

In the random oracle model, both the signing and verification algorithms can access the random hash oracle. Typically, M belongs to the set $\{0,1\}^*$. It is generally required that $\mathsf{Verify}(\mathsf{pk}, \mathsf{Sign}(\mathcal{I}, \mathsf{sk}, M)) = 1$ for every M in $\{0,1\}^*$.

Definition 2.6 (Existential Unforgeability under Chosen Message Attack (EUF-CMA))**.** Given a digital signature scheme $\mathcal{S} = \{\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}\}$, consider a PPT adversary $\mathcal{A}$ who is given a public key generated by KeyGen and the oracle access to the Sign which it can adaptively send query messages.

Let $\mathcal{M}$ be the set of messages queried by $\mathcal{A}$. The digital signature scheme $\mathcal{S}$ is said to be existentially unforgeable under chosen message attack if there is no such a PPT adversary $\mathcal{A}$ that can produce, except with negligible probability, a valid signature on a message $m \notin \mathcal{M}$.

2.11 Elliptic Curve Digital Signature Algorithm (ECDSA)

The ECDSA algorithm can be split into four parts, which we describe as follows:

1. Setup. Given the security parameter 1^λ, the setup runs $\mathcal{G}_{\mathsf{ECC}} \leftarrow \mathsf{GGen}_{\mathsf{ECC}}(1^\lambda)$, outputting $\mathsf{param} = \mathcal{G}_{\mathsf{ECC}}$. We omit the input param for brevity.

2. KeyGen. This part outputs $(\hat{Q}, x)$, where $x \xleftarrow{\$} \mathbb{Z}_q$ is a randomly chosen secret key, and $\hat{Q} = \hat{P}^x$ serves as the public key.

3. Sign. This component calculates $\hat{R} = (r_x, r_y) = \hat{P}^k$, $r = r_x \bmod q$, and $s = k^{-1}(xr + H(m)) \bmod q$, where $k \xleftarrow{\$} \mathbb{Z}_q$ and m is the input message. It provides (r, s) as the signature.

4. Verify. Upon receiving the public key $\hat{Q}$, a message m, and a signature (r, s), this part computes $\hat{R} = (r_x, r_y) = (\hat{Q}^r \hat{P}^{H(m)})^{1/s}$. It outputs 1 if r equals r_x, and 0 otherwise.

2.12 Guillou-Quisquater Signature (GQ)

We review the original GQ signature scheme in [46].It was later used to construct an identity-based signature version in ISO/IEC 14888-2 standard through introducing a Public Key Generator (PKG) similar to the RSA scheme and hash function to generate the public/secret key pairs for users.

1. KeyGen. Choose randomly two large primes p and q and compute $n = pq$. Select an integer v s.t. $0 < v < \phi(n)$ and $gcd(v, \phi(n)) = 1$, where ϕ(n) is the Euler function. Select a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_{v-1}$. Randomly select the secret key B from $\mathbb{Z}_n$ and compute $J = B^{-v} \bmod n$. Set $PK = (n, v, J, H)$ and $SK = (p, q, B)$.

2. Sign. Randomly select r from $\mathbb{Z}_n$, then compute $T = r^v \bmod n$, $h = H(M, T)$ and $t = rB^h \bmod n$, where M is the message to be signed. Output signature $\sigma = (t, h)$.

3. Verify. Upon receiving a signature $\sigma = (t, h)$ of message M, compute $T' = t^v J^h \bmod n$. If $h = H(M, T')$, output 1; otherwise, output 0.

The correctness of the process is confirmed by $T' = t^v J^h = (rB^h)^v J^h = r^v (JB^v)^h = r^v = T \mod n$.

According to [7], GQ identification is secure under the RSA-OMI (RSA one-more inversion) assumption, and after applying the Fiat-Shamir transformation, GQ signature is secure under the RSA-OMI assumption in the ROM (random oracle model). As for [46], h should be uniformly chosen from $[0, v-1]$. Therefore, v must be sufficiently large to ensure collision resistance for H.

RSA Trapdoor. If p and q are known to the system developer, malicious developer could easily acquire the secret key B from public J by simply calculating $d = v^{-1} \bmod (p-1)(q-1)$ and then $B = J^{-d}$.

2.13 Definition of Multi-Signature Scheme

Let $\mathcal{P} = \{P_1, P_2, ..., P_n\}$ be a group of n players. Let $\mathcal{I}$ be the global information string. The algorithms of a multi-signature scheme $\mathcal{MS} = (\mathsf{MKeyGen}, \mathsf{MSign}, \mathsf{Verify})$ are defined as follows:

1. **Key Generation** (MKeyGen): A randomized key generation algorithm, MKeyGen takes a global information $\mathcal{I}$ and outputs a pair $(\mathsf{sk}, \mathsf{pk})$ of a secret key and a public key. Each player $P_i \in \mathcal{P}$ runs MKeyGen to obtain a pair of secret and public keys $(\mathsf{sk}_i, \mathsf{pk}_i)$.

2. **Multi-Signature Generation** (MSign): This is a potentially randomized interactive protocol run by an arbitrary subset of players $\mathcal{L} \subseteq \mathcal{P}$. Each player $P_i \in \mathcal{L}$ inputs a message $M \in \{0,1\}^*$, the global information $\mathcal{I}$, and their secret key sk_i. The output of the algorithm is a triple $\mathcal{T} = (M, \mathcal{L}, \sigma)$, which includes the message, a description of the subgroup $\mathcal{L}$, and the multi-signature.

3. **Verification** (Verify): This deterministic algorithm takes $(M, \mathcal{L}, \sigma)$, the public keys of all players in $\mathcal{L}$, and $\mathcal{T}$, and outputs either 1 (accepts) or 0 (rejects).

2.14 Multi-signature Unforgeability in Dishonest Majority Model with Static Corruption

In the dishonest majority model, a majority of adversaries can exist who may arbitrarily deviate from the protocol. Aborting the protocol is not considered a security violation in this model. It assumes the existence of both a broadcast channel and a point-to-point channel among every participant, and allows for static corruption, which requires adversaries to select participants to corrupt before the start of the protocol.

Following [44], we present a game-based definition of security analogous to EUF-CMA (Existential Unforgeability under Chosen Message Attacks): multi-signature unforgeability under chosen message attacks (MU-CMA)

Definition 2.7 (Multi-signature Unforgeability)**.** Consider a multi-signature scheme $\mathcal{MS} = (\mathsf{MKeyGen}, \mathsf{MSign}, \mathsf{Verify})$ with N parties and a PPT malicious adversary $\mathcal{A}$ who corrupts at most $N-1$ players, given the view of $\mathsf{MKeyGen}$ and MSign on inputs of adaptively chosen messages, denoted by $\mathcal{M}$, and the corresponding signatures on those messages. The multi-signature scheme $\mathcal{MS}$ is said to be existentially unforgeable (EUF-CMA) if there is no such a PPT adversary $\mathcal{A}$ that can produce, except with negligible probability, a valid signature on a message $m \notin \mathcal{M}$.

2.15 Non-Malleable Equivocable Commitment

A non-interactive trapdoor commitment scheme consists of four algorithms: KG, Com, Ver, Equiv.

1. **Key Generation** (KG): This algorithm takes a security parameter as input, and outputs a pair (pk, tk). Here, pk is the public key of the commitment scheme, and tk is the trapdoor key.

2. **Commitment** (Com): This algorithm is used to create the commitment. Given the public key pk and a message M, it outputs a pair $[C_M, D_M] = \mathsf{Com}(pk, M, R)$, where R is obtained by coin tossing. C_M is the commitment string, and D_M is the decommitment string.

3. **Verification** (Ver): This is the verification algorithm. Given the public key pk, commitment C, and decommitment string D, it outputs the original message M if the verification passes, or $\perp$ otherwise.

4. **Equivalence (Equiv)**: This algorithm reveals a commitment in any possible way given the trapdoor key. It takes as input the public key pk, strings M and R such that $[C_M, D_M] = \mathsf{Com}(pk, M, R)$, a different message $M' \neq M$, and a string T. If $T = tk$, then Equiv outputs a decommitment string D' such that $\mathsf{Ver}(pk, C_M, D') = M'$.

This set of algorithms define the non-interactive trapdoor commitment scheme. The Equiv algorithm, in particular, makes it possible to open the commitment in a different way, given the trapdoor key, which can be useful in certain cryptographic protocols.

The correctness holds if $[C_M, D_M] = \mathsf{Com}(pk, M, R)$, then we have $\mathsf{Ver}(pk, C_M, D_M) = M$. Now we review the properties of information theoretic security, secure binding and non-malleability.

Information theoretic security. For every message pair (M, M') the distributions C_M and $C_{M'}$ are statistically close.

Secure binding. We say that an adversary $\mathcal{A}$ wins if it outputs (C, D, D') s.t. $\mathsf{Ver}(pk, C, D) = M$, $\mathsf{Ver}(pk, C, D') = M'$, $M' \neq M$, $M \neq \perp$, $M' \neq \perp$. We require that for all efficient algorithms $\mathcal{A}$, the probability that $\mathcal{A}$ wins is negligible.

Non-Malleability[34]. A commitment is non-malleable if no adversary $\mathcal{A}$, given a commitment C to message M, is able to produce another commitment C' s.t. after the revealing of C to M, $\mathcal{A}$ can successfully decommit to a related message M'.

Remarks. As in [42], we can use any secure hash function $H : \{0,1\}^* \to \{0,1\}^n$ and instantiate the commitment to x as $h = H(x, r)$, where r is the blind factor uniformly chosen from $\{0,1\}^n$, assuming that H *behaves as a random oracle*. The decommitment contains a blind factor and the committed value x. For a user receiving a commitment h, it can use the later received decommitment (x, r) to check the validity of the commitment x. The validation passes if and only if $h = H(x, r)$. We adopt this *efficient random oracle version* in our implementation.

2.16 Zero-knowledge Proof

A proof system for a relation $\mathcal{R} \subset \mathcal{X} \times \mathcal{W}$ is a triplet of randomized polynomial time algorithms $(\mathsf{Setup}, \mathsf{P}, \mathsf{V})$, which function as follows:

1. **Setup (Setup)**: This algorithm takes a security parameter 1^λ and generates a common reference string (CRS), denoted as param.

2. **Prover** (P): This algorithm takes the CRS param, a statement $x \in \mathcal{X}$, and a witness $w \in \mathcal{W}$ as inputs.

3. **Verifier** (V): This algorithm takes the CRS param and statement x as inputs. After interacting with P, it outputs either 0 or 1.

Interaction between the prover and the verifier is represented by the transcript $\langle \mathsf{V}(\mathsf{param}, x), \mathsf{P}(\mathsf{param}, x, w) \rangle$, which equals 1 if V accepts the transcript.

The zero-knowledge proof system implemented in this context is a canonical Σ Protocol, which involves three moves between the prover P and the verifier V: The first message is a commitment from V to P, denoted by t. The second message is a random coin from V to P, denoted by c. The third message is a response from P to V, denoted by z.

We require the following properties for our Σ protocol.

Completeness. If $(x, w) \in \mathcal{R}$, prover P with auxiliary input w convinces V with overwhelming probability.

Special soundness. Given two transcripts (t, c, z) and (t, c', z') where $c' \neq c$ and $z \neq z'$ for a statement x, there exists an extractor which produce w s.t. $(x, w) \in \mathcal{R}$ in polynomial time with non-negligible probability.

Honest verifier zero-knowledge (HVZK). Given x and c, there exists a simulator without knowing w which outputs (t, z) such that (t, c, z) is indistinguishable to the real transcript between P with auxiliary input w and V in polynomial time with non-negligible probability.

Chapter 3

Address-based Signature

3.1 Motivation

Public key infrastructure (PKI) is a crucial component of modern secure communication, where a public key is used to verify a signature. The authenticity of the public key is validated by a digital certificate issued by a trusted Certificate Authority (CA). However, managing PKI can be difficult and cumbersome due to the large number of pre-installed intermediary certificates issued and signed by CAs. This increases the risk of key compromise and creates complexity in deciding which CAs or intermediate CAs to trust. The Google Chrome incident in 2018, where millions of certificates issued by Symantec were distrusted, is an example of the fragility of this chain of trust.

Identity-based cryptography [69] was proposed to address these issues by using a human-recognizable string, such as an email address, as the user identity. The user identity string is used as the public key for encryption or signature verification. However, this approach still requires a trusted authority to generate the user's identity-based secret key.

How to adapt public key cryptography to a distributed system without relying on a trusted authority remains a challenging question. The address systems used in blockchain and XIA deviate from classical digital signatures, inspiring a new concept: *Address-Based Signature*.

Hereby, we formally propose the concept of *address-based signature*. Here, an "address" is defined as an arbitrary, collision-resistant string that describes the signer. In our proposal, we consider the address as the hash of the public key. (In some cases, the address may be appended with a checksum value.) The definitions of the key generation and signing algorithms for address-based signatures closely resemble those of classical digital signatures, with one key

difference. During the verification of a signature over a message, only the address—the hash of the public key—is given to the verifier. Despite this, the verifier can still check the validity of the signature. This innovative concept of an address-based signature is particularly suitable for applications in blockchain systems or XIA, offering a novel approach that streamlines the process for users while maintaining security.

Advantages of Address-based Signatures. The concept of address-based signature brings several distinct advantages to the table:

- **Shorter Address**: The primary benefit of using an address-based signature is the shorter address length compared to the size of a public key. As we progress towards higher security levels—such as using 512-bit ECC—or post-quantum secure signature schemes like lattice-based signatures, which typically have public key sizes greater than 1000 bytes, this advantage becomes more significant. A shorter address means lower transaction fees for blockchain applications, making this approach cost-effective.
- **Compatibility**: Address-based signatures enable the use of the same address data structure across different signature schemes, enhancing the system's versatility. This feature is especially beneficial for XIA, where different hosts may use different signature schemes. For instance, users employing ECDSA and Schnorr signatures can use the same 160-bit address data structure. Similarly, signature schemes using 256-bit ECC and 512-bit ECC can use the same 160-bit address data structure. Only a few bits are needed to indicate the signature scheme and the bit-length used, making this system highly scalable and adaptable.

The ECDSA used in Ethereum can be viewed as an address-based signature. We review it in section 3.4.

3.2 Our Contributions

There are three main contributions in this work.

Contribution 1: Formalization of address-based signature and its security model.

We present a formal definition of the address-based signature and delineate two security paradigms for it: unforgeability and collision resistance. In these models, we account for an attacker who could acquire the randomness utilized

in the initialization of system parameters. This approach offers robust security assurances by taking into consideration the potential risks posed by blockchain system developers who might embed a trapdoor in the system parameters. In the unforgeability model, the attacker is allowed to obtain a number of addresses and to corrupt some of them (by obtaining their secret keys). We require that no PPT adversary can forge a signature with respect to any uncorrupted address. This gives a strong model of unforgeability which is comparable to the multi-user unforgeability of standard signature.

Contribution 2: Constructing a new, compact address-based signature.

In this study, we introduce a compact address-based signature scheme tailored for cryptocurrency, detailed in Algorithm 3. This scheme matches the efficiency of both the ECDSA and Schnorr signatures but eliminates the key drawbacks these systems exhibit when implemented in a blockchain context. A comparative analysis of their efficiencies is provided in Figure 3.1.

Algorithm 3: Our Address-based Signature. The secret key is x, the public key is $X = g^x$ and the address is $A = H(X)$.

Procedure SIGN(x, m)**:**
- $r \leftarrow_s \mathbb{Z}_p$;
- $R = g^r$;
- $c = H_{zp}(R, m)$;
- $z = r^{-1}(c + x) \mod p$;
- return $\sigma = (R, z)$;

Procedure VERIFY$(A, m, \sigma = (R, z))$**:**
- $c = H_{zp}(R, m)$;
- $X = R^z g^{-c}$;
- **if** $A = H(X)$ **then**
  - stop with 1;
- stop with 0;

When it comes to provable security, the ECDSA has only been proven secure in the generic group model [15] or the less conventional bijective random oracle model [38]. In contrast, our proposed scheme's security is based on the discrete logarithm assumption in the random oracle model. Focusing on blockchain applications, the Schnorr signature shows incompatibility with the Bitcoin Improvement Protocol (BIP)-32, as evidenced in [79]. Conversely, our scheme aligns seamlessly with addresses that utilize the BIP 32 non-hardened key derivation. Hence, given these factors, our proposed scheme proves to be more apt for address-based systems like blockchain and XIA.

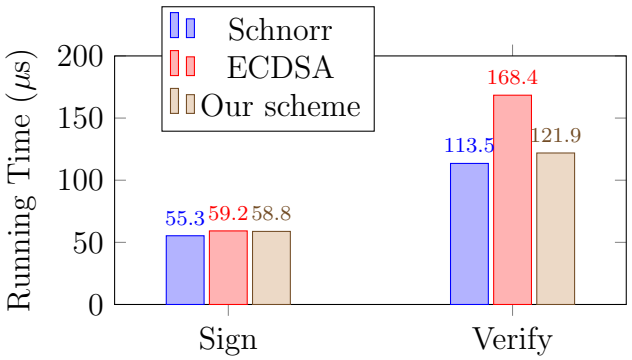


FIGURE 3.1: Comparison of running time.

The security proof of our construction holds not just practical significance but also offers intriguing theoretical insights. In the proof, we illustrate that the signature's unforgeability necessitates the address hash function H to exhibit *always second preimage resistance*. This conclusion is far from being trivial, given that *always second preimage resistance* is not a property inherently implied by *collision resistance*, as established in [65]. This outcome highlights the criticality of formalizing the concept of address-based signatures. From a practical perspective, a hash function like SHA256 is anticipated to be both collision-resistant and always second-preimage resistant. Yet, our security proof establishes that these two unique properties are required from the hash function. Theoretically, it's fascinating to observe how these two distinct properties of a hash function can influence two different properties (collision resistance and unforgeability) of an applicable signature scheme.

Contribution 3: Constructing address-based signature from existing signature schemes. We demonstrate that address-based signatures can be constructed from other traditional digital signatures using the generic constructions GC1 in Section §3.8.1 and GC2 in Section §3.8.2. The efficiency of these schemes is compared in Table 3.1. In the context of secp256k1, the secret key can be encapsulated in just 32 bytes. The compressed public key can be represented using 33 bytes (32 bytes for the x-coordinate and a single byte, 0x02 or 0x03, for the sign of the y-coordinate). Certain cryptocurrencies, such as Bitcoin, append the address with a checksum value, while others like Ethereum do not. In Bitcoin, the ECDSA signature undergoes further encoding in DER format. For an equitable comparison, we have excluded the size of the address checksum and signature encoding from Table 3.1.

As summarized in Table 3.1, the most efficient schemes are is the address-based ECDSA, Schnorr signature and our construction. However, they both have their own drawbacks. For ECDSA, it is known to be malleable and several attacks on cryptocurrency exchange are found using this property [30]. In addition, the security proof of ECDSA is only given in the non-standard

Address-based Signature	Signature type	Address type	Cost in bits	major drawbacks
ECDSA (SEC Standard)	(r, s)	$H(X)$	672	Malleable, non-standard security proof, repeated computation in verification.
ECDSA (Ethereum)	(r, s, v)	$H(X)$	680	Non-standard security proof.
ECDSA + GC 2 (Bitcoin/Ripple)	(r, s, X)	$H(X)$	936	Malleable, non-standard security proof, longer signature
Schnorr (Appendix 3.5)	(R, z)	$H(X)$	680	Not secure in strong known related-key attack [79] (BIP32)
Key-prefix Schnorr + GC 1	(c, z)	X	776	Longer address
Key-prefix Schnorr + GC 2 (BIP)	$(c, z), X$	$H(X)$	936	Longer signature
BLS/BB + GC 1	σ	X	776	Slow verification using pairing
BLS/BB + GC 2	σ, X	$H(X)$	936	Slow verification using pairing,longer signature
Our construction (§3.6.1)	(R, z)	$H(X)$	680	-

TABLE 3.1: Comparison of address-based signatures. The total communication cost is the sum of address and the signature size. GC stands for the generic construction in section 3.8. X is the public key. Details of the symbols are explained in section 3.8.

bijective random oracle model [38]. For Schnorr signature, it is known to the Bitcoin community that Schnorr signature is not compatible with some Bitcoin Improvement Protocol (BIP), such as BIP32's non-hardened derivation. The attack is summarized in the strong known related-key attack [79]. As a result, key-prefixed Schnorr is preferred in the recent BIP proposal [75]. However, using key-prefixed Schnorr will result in a 50% increase in signature size if we keep the old address structure, or a 60% increase in address size if we keep the old signature structure. The pairing-based signatures like BLS [13] and BB [10], as shown in 4, 5, require a larger public key. Therefore, our construction is the optimal solution for address-based signature.

Algorithm 4: BLS Signature

Procedure SIGN(x, m)**:**
  return $\sigma = H_{G2}(m)^x$;
Procedure VERIFY(X, m, σ)**:**
  if $\hat{e}(g_1, \sigma) = \hat{e}(X, H_{G2}(m))$ **then**
    stop with 1;
  stop with 0;

Algorithm 5: BB Signature

Procedure SIGN(x, m)**:**
  $c = H_{zp}(m)$;
  return $\sigma = g_2^{1/(x+c)}$;
Procedure VERIFY(X, m, σ)**:**
  $c = H_{zp}(m)$;
  if $\hat{e}(g_1, g_2) = \hat{e}(Xg_1^c, \sigma)$ **then**
    stop with 1;
  stop with 0;

3.3 Security Model

3.3.1 Address-based Signature

Address-based signatures bear a resemblance to identity-based signatures in that a string is utilized as the "public key" for signature verification. An advantage of identity-based signatures is that the string can be something significant, like an email address. However, this system involves a trusted third party to issue identity-based secret keys. Contrastingly, no trusted third party is needed to issue address-based keys to distinct users. This makes address-based signatures particularly apt for XIA and blockchain applications, where decentralization and lack of a single trusted authority are key principles.

Address-based signature is also quite similar to the multi-user security of digital signature (MU-EUF-CMA) [54]. However, one key difference is that the challenger for address-based signature is not allowed to have any trapdoor when generating mpk. It is guaranteed by allowing the adversary to obtain the randomness used to run Setup. This security requirement is important for public blockchain, since they do not allow trusted setup.

An address-based signature scheme consists of the following four algorithms:

1. $\mathsf{Setup}(1^\lambda)$: On input a security parameter λ, it outputs a master public key mpk.
2. $\mathsf{KeyGen}(\mathsf{mpk})$: It outputs an address addr, a public key pk and a secret key sk.
3. $\mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, m)$: On input a master public key mpk, a secret key sk and a message m, it outputs a signature σ.
4. $\mathsf{Verify}(\mathsf{mpk}, \mathsf{addr}, m, \sigma)$: On input a master public key mpk, an address addr, a message m and a signature σ, it outputs $(1, \mathsf{pk})$ for valid signature (where pk is the corresponding public key) and outputs $(0, \perp)$ otherwise.

Threat Model. In the system of address-based signature, there are three parties:

- Signer: the entity that generates key pairs and signs a message.
- Verifier: the entity that verifies the signature. In the context of cryptocurrency, the verifier could be the recipient of the money or the miners executing the consensus algorithms.
- System developer: the entity that generates the system parameters, denoted as mpk. In practical scenarios, system developers typically justify their choice of parameters by aligning them with established standards or setting parameters as a hash of some significant string.

In this work, we scrutinize the security of each signer against other parties within the system. Following standard unforgeability requirements, the attacker can procure signatures for certain messages from the target signer. We stipulate that no probabilistic polynomial time (PPT) adversary can forge a signature on a novel message. An additional security requirement is collision resistance, which implies that no PPT adversary can generate two unique secret keys with identical addresses. In these two security models, we further permit the attacker to procure all randomness employed by the system developer in generating the system parameters. This simulates an attack on unforgeability and collision resistance, even against a system developer who is honest-but-curious by nature.

3.3.2 Unforgeability

Our security model is defined similarly to the concept of *multi-user existential unforgeability against chosen message attack* [54]. The original security

model in [54] does not take into account whether par is generated by a trusted third party. In this research, we consider this aspect and define *trapdoor-less multi-user existentially unforgeable under chosen message attack* (TMU-EUF-CMA). In the first step of this security game, the challenger executes $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda; \rho)$ with randomness ρ. The adversary is also provided with ρ in the initial step. This approach is inspired by the trustless setup assumption that is prevalent in most public blockchains.

It is straightforward to observe that the Schnorr signature is TMU-EUF-CMA secure, given that the system parameter par only includes a cyclic group $\mathbb{G}$ with prime order p, its generator g, and a cryptographic hash function H_{zp}. The generation of par does not entail a trapdoor, which aligns with the absence of a trusted third party in the model. In contrast, an RSA-based signature with a common modulus $N = pq$ does not provide security in this model. The concept of *existential unforgeability under chosen message attack* is defined as a game between a challenger and an adversary, as outlined in the following steps.

1. The challenger runs $\mathsf{mpk} \leftarrow \mathsf{Setup}(1^\lambda)$. The challenger runs $(\mathsf{addr}_i, \mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{mpk})$ for $i \in [1, q_k]$. He stores $(\mathsf{addr}_i, \mathsf{pk}_i, \mathsf{sk}_i)$. The adversary is given mpk and $(\mathsf{addr}_i, \mathsf{pk}_i)$ for $i \in [1, q_k]$, as well as the randomness used to run Setup.
2. The adversary can adaptively issue a sign oracle query on message m_j and pk_{i_j}, for $j \in [1, q_s]$. The challenger answers by $\sigma_j \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_{i_j}, m_j)$.
3. Finally, the adversary outputs an index $i^* \in [1, q_k]$, a message m^* and a signature σ^*.

The adversary wins the game if $(1, \cdot) \leftarrow \mathsf{Verify}(\mathsf{mpk}, \mathsf{addr}_{i^*}, m^*, \sigma^*)$ and $(i^*, m^*) \neq (i_j, m_j)$ for all $j \in [1, q_s]$.

Definition 3.1. An address-based signature scheme is (t, ϵ, q_k, q_s)-TMU-EUF-CMA if no adversary running in time t can win the above game with probability ϵ, with at most q_s query to the Sign oracle.

3.3.3 Collision Resistance

The second security requirement pertains to the collision resistance of the address. We mandate that it should be challenging to discover two secret keys that correspond to the same address. The importance of collision resistance is also explicitly stated in XIA [1]

The notion of *collision resistance* is defined as the following game between a challenger and an adversary.

1. The challenger runs $\mathsf{mpk} \leftarrow \mathsf{Setup}(1^\lambda)$. The adversary is given mpk, as well as the randomness used to run Setup.
2. Finally, the adversary outputs signatures σ_b^*, messages m_b^* and an address addr^*, for $b = 0/1$.

The adversary wins the game if $(1, \mathsf{pk}_b^*) \leftarrow \mathsf{Verify}(\mathsf{mpk}, \mathsf{addr}^*, m_b^*, \sigma_b^*)$ for $b = 0/1$ and $\mathsf{pk}_0^* \neq \mathsf{pk}_1^*$.

Definition 3.2. An address-based signature scheme is collision resistant if no PPT adversary can win the above game with non-negligible probability.

Modelling Signatures for Blockchain. It is clear that address-based signature is a more suitable notion for signature used in blockchain applications. Firstly, there is a system-wise public parameters mpk which is usually written in the source code of the blockchain application. Our security model allows the adversary to obtain the randomness used to generate mpk. Therefore, it demonstrates the security even against the developer of the blockchain system. It is also the key difference between our model and the multi-user security model of EUF-CMA. Secondly, the collision resistant property of the address is now formally captured into the security model.

3.4 Address-based ECDSA Signature

Some of the classical digital signature schemes allow *public key recovery* and can be used as address-based signature directly. One example is ECDSA [73]. Consider x as a secret key and g as a generator of a multiplicative ECC group $\mathbb{G}$, the public key is $X = g^x$. Denote $H : \mathbb{G} \to \{0,1\}^{160}$ and $H_{zp} : \{0,1\}^* \to \mathbb{Z}_p$ as cryptographic hash functions and m as a message. The address is $A = H(X)$.

The SEC standard introduced an ECDSA scheme with public key recovery [17]. Notably, the verification of the ECDSA signature does not necessitate the use of a public key. However, this comes at a cost of potentially having to repeat the verification computation up to four times (approximately 49.99% chance for one time, and another 49.99% for two times).

Ethereum implemented an address-based ECDSA signature as demonstrated in Algorithm 6. They incorporated an additional byte, denoted as v, to store information pertaining to the varying cases in verification.

Ethereum directly employed ECDSA as an address-based signature. That said, there are several well-known limitations associated with the ECDSA signature.

Algorithm 6: Address-based ECDSA Signature in Ethereum

Procedure SETUP(1^λ)**:**
- pick a cyclic EC group $\mathbb{G}$ of order p;
- pick a generator g of $\mathbb{G}$;
- pick hash functions $H : \mathbb{G} \to \{0,1\}^*$ and $H_{zp} : \{0,1\}^* \to \mathbb{Z}_p$;
- return $\mathsf{mpk} = (\mathbb{G}, p, g, H, H_{zp})$;

Procedure KEYGEN(mpk)**:**
- $x \leftarrow_s \mathbb{Z}_p$;
- $X = g^x$;
- $A = H(\mathsf{pk})$;
- return $(\mathsf{sk} = x, \mathsf{pk} = X, \mathsf{addr} = A)$;

Procedure SIGN(x, m)**:**
- $k \leftarrow_s \mathbb{Z}_p$;
- $K = g^k$ and K has coordinates (K_x, K_y);
- $r = K_x \mod p$;
- $c = H_{zp}(m)$;
- $s = k^{-1}(c + rx) \mod p$;
- **if** $K_x = r$ *and* K_y *is even* **then**
  - $v = 27$;
- **if** $K_x = r$ *and* K_y *is odd* **then**
  - $v = 28$;
- **if** $K_x = r + p$ *and* K_y *is even* **then**
  - $v = 29$;
- **if** $K_x = r + p$ *and* K_y *is odd* **then**
  - $v = 30$;
- return $\sigma = (r, s, v)$;

Procedure VERIFY($\mathsf{mpk}, \mathsf{addr} = A, m, \sigma = (r, s, v)$)**:**
- $c = H_{zp}(m)$;
- **if** $v = 27$ **then**
  - compute a point K having x-coordinate r and even y-coordinate ;
- **if** $v = 28$ **then**
  - compute a point K having x-coordinate r and odd y-coordinate ;
- **if** $v = 29$ **then**
  - compute a point K having x-coordinate $r + p$ and even y-coordinate ;
- **if** $v = 30$ **then**
  - compute a point K having x-coordinate $r + p$ and odd y-coordinate ;
- $X = (K^s g^{-c})^{1/r}$;
- **if** $A = H(X)$ **then**
  - stop with $(1, X)$;
- stop with $(0, \perp)$;

Firstly, there's a lack of a known security proof for ECDSA in either the standard model or the random oracle model. The EUF-CMA security of ECDSA has only been proven in the generic group model [15, 16, 70], or in the bijective random oracle model [38].

Secondly, the ECDSA signature is acknowledged to be malleable: if (r, s) is a valid signature from address A, then $(r, -s)$ is also a valid signature. This malleability of ECDSA is one of the contributing factors to transaction malleability in the Bitcoin system, leading to the discovery of several related attacks [30]. A common workaround is to exclusively use the smaller of s and $-s \bmod p$ during the signing phase.

Thirdly, as delineated in the SEC standard [17], the public key recovery process involves repetition due to four potential address choices. Consequently, the computational complexity in verification is, on average, 1.5 times higher (given the rarity of the last two cases). In their implementation, Ethereum used this address-based ECDSA and addressed the third issue by adding an extra byte of information about the x and y coordinates of K.[1]

3.5 Address-based Schnorr Signature

Similar to ECDSA, the original Schnorr signature [67] also supports public key recovery from the signature. Consequently, we can construct an address-based Schnorr signature as shown in Algorithm 7. For this construction, we use the same concepts and ECC group as ECDSA. However, it still lacks security under the strong known related-key attack, a flaw which is similar to the one identified in [79].

Security Analysis. It is straightforward that the collision resistant property of the address-based Schnorr signature follows the collision resistant property of H.

Theorem 3.3. *The address-based Schnorr signature is collision resistant if H is a collision resistant hash function.*

On the other hand, the theorem about the EUF-CMA security gives us a non-trivial result. The address-based Schnorr signature is secure if the standard Schnorr signature (whose signature is (c, z)) is EUF-CMA secure and the always second-preimage resistant property holds for H. There is no known

[1] The value $v = 27/28$ is frequently used in the original Ethereum protocol. The Ethereum Improvement Protocol 155 proposes alternative formulas for computing v. Other potential solutions to address this problem include using only the positive y coordinate of K, or using the y coordinate of K with a Jacobi symbol of 1 only.

Algorithm 7: Address-based Schnorr Signature

Procedure SETUP(1^λ)**:**
  pick a cyclic group $\mathbb{G}$ of order p;
  pick a generator g of $\mathbb{G}$;
  pick hash functions $H : \mathbb{G} \to \{0,1\}^*$ and $H_{zp} : \{0,1\}^* \to \mathbb{Z}_p$;
  return $\mathsf{mpk} = (\mathbb{G}, p, g, H, H_{zp})$;
Procedure KEYGEN(mpk)**:**
  $x \leftarrow_s \mathbb{Z}_p$;
  $X = g^x$;
  $A = H(\mathsf{pk})$;
  return $(\mathsf{sk} = x, \mathsf{pk} = X, \mathsf{addr} = A)$;
Procedure SIGN($\mathsf{mpk}, \mathsf{sk} = x, m$)**:**
  $r \leftarrow_s \mathbb{Z}_p$;
  $R = g^r$;
  $c = H_{zp}(R, m)$;
  $z = r + cx \mod p$;
  return $\sigma = (R, z)$;
Procedure VERIFY($\mathsf{mpk}, \mathsf{addr} = A, m, \sigma = (R, z)$)**:**
  $c = H_{zp}(R, m)$;
  $X = (g^z R^{-1})^{1/c}$;
  if $A = H(X)$ **then**
    stop with $(1, X)$;
  stop with $(0, \perp)$;

relationship between the collision resistant property and the always second-preimage resistant property as shown in [65]. Therefore, a secure address-based Schnorr signature requires both properties for the hash function H.

Theorem 3.4. *The address-based Schnorr signature is TMU-EUF-CMA secure if the standard Schnorr signature is TMU-EUF-CMA secure and H is an always second-preimage resistant hash function.*

Proof. Suppose $\mathcal{B}$ is the attacker trying to break the TMU-EUF-CMA security of the standard Schnorr signature (with signature (c, z)). $\mathcal{B}$ is given $\mathsf{par} = (\mathbb{G}, p, g, H_{zp})$, the randomness ρ' to generate par and $(\mathsf{pk}_1, \ldots, \mathsf{pk}_{q_k})$ from its challenger $\mathcal{C}$. $\mathcal{B}$ tries to forge the standard Schnorr signature by using an TMU-EUF-CMA adversary $\mathcal{A}$ on the address-based Schnorr signature.

Setup. $\mathcal{B}$ picks H from the family of always second-preimage resistant hash function, possibly with the use of some random tape ρ. (In the case of Bitcoin, the definition of H includes the random choice of the prefix representing the type of public key, and the prefix representing the network ID.)

$\mathcal{B}$ sets $\mathsf{mpk} = (\mathbb{G}, p, g, H, H_{zp})$. $\mathcal{B}$ sends mpk, $(\mathsf{pk}_1, \ldots, \mathsf{pk}_{q_k})$ and randomness $\rho||\rho'$ to the adversary $\mathcal{A}$.

Oracle Query. $\mathcal{B}$ answers the sign oracle queries as follows:

- Sign: On input a message m_j and a public key pk_{i_j}, $\mathcal{B}$ asks the signing oracle of its challenger $\mathcal{C}$ with input m_j and pk_{i_j}. $\mathcal{B}$ obtains (c, z) and computes $R = g^z {\mathsf{pk}_{i_j}}^{-c}$. $\mathcal{B}$ returns $\sigma = (R, z)$ to $\mathcal{A}$.

Output. Finally $\mathcal{A}$ outputs an index i^*, a message m^* and a signature $\sigma^* = (R^*, z^*)$. $\mathcal{B}$ can compute

$$c^* = H_{zp}(R^*, m^*), \qquad X' = (g^{z^*} R^{*-1})^{1/c^*},$$

such that $A_{i^*} = H(X')$. If the *always second-preimage resistance* property holds for H, then $X' = \mathsf{pk}_{i^*}$ for the random choice of ρ. Then we have

$$R^* = g^{z^*} {\mathsf{pk}_{i^*}}^{-c^*}.$$

Then $\mathcal{B}$ returns (i^*, c^*, z^*) as the forgery to the standard Schnorr signature. □

Drawbacks for Address-based Schnorr Signatures. It's acknowledged within the Bitcoin community that the Schnorr signature is not compatible with some Bitcoin Improvement Protocols (BIPs), such as the combination of non-hardened derivation from BIP32 with the BIP118 SIGHASH_NOINPUT flag. Recently, Yuen and Yiu [79] found that the Schnorr signature lacks security in the strong known related-key attack model. This model encapsulates the aforementioned attack within the blockchain system. Therefore, we would like to investigate if there are other possible solutions for address-based signatures.

3.6 Compact and Secure Address-based Signature

Given that both address-based ECDSA and Schnorr signatures exhibit specific issues when utilized in blockchain, it becomes pertinent to explore if we can devise a secure and efficient address-based signature. In this section, we provide our construction and substantiate the security of our scheme within the random oracle model. Subsequently, we present the construction of our address-based signature.

3.6.1 Our Construction

Our construction can be regarded as the middle ground between Schnorr signature and ECDSA. We eliminate the extraction of the x-coordinate in ECDSA, a feature that complicates the derivation of a standard security proof for ECDSA.

On the other hand, we have to prevent the strong known related-key attack [79] in Schnorr signature, while preserving the public key recovery property (unlike the key-prefixed Schnorr signature). As a result, we remove the linear structure of the Schnorr signature $z = r + cx$ and change it to $z = r^{-1}(c + x)$. The randomness r^{-1} is multiplied with the secret key x to prevent the strong known related-key attack.

Considering this requirement of the address-based system, the actual signature scheme is as follows.

Setup. On input the system parameter λ, it generates a cyclic group $\mathbb{G}$ of prime order p. with a generator $g \in \mathbb{G}$. Suppose $H : \mathbb{G} \to \{0,1\}^*$ and $H_{zp} : \{0,1\}^* \to \mathbb{Z}_p$ are cryptographic hash functions. It sets $\mathsf{mpk} = (p, \mathbb{G}, g, H, H_{zp})$.

KeyGen. On input mpk, it picks a random $x \in \mathbb{Z}_p$ and computes $X = g^x$. It outputs the secret key x, the public key X and the address $A = H(X)$.

Sign. On input mpk, the secret key x and a message m, it picks a random number $r \in \mathbb{Z}_p$ and computes:

$$R = g^r, \quad c = H_{zp}(R, m), \quad z = r^{-1}(c + x) \mod p.$$

It outputs the signature $\sigma = (R, z)$.

Verify. On input mpk, the address A, a message M and a signature $\sigma = (R, z)$, it computes $c = H_{zp}(R, M)$, $X' = R^z g^{-c}$ and outputs $(1, X')$ if $A = H(X')$. Otherwise, it outputs $(0, \perp)$.

Theorem 3.5. *Our address-based signature is collision resistant if H is a collision resistant hash function.*

Theorem 3.6. *Our address-based signature is TMU-EUF-CMA secure if the DL assumption holds in the random oracle model and H is an always second-preimage resistant hash function.*

Proof. Suppose $\mathcal{B}$ is given a DL problem X^* and $SI = (\mathbb{G}, p, g)$. $\mathcal{B}$ tries to solve the DL by using an TMU-EUF-CMA adversary $\mathcal{A}$.

Setup. $\mathcal{B}$ picks cryptographic hash function H_{zp} and H using randomness ρ. $\mathcal{B}$ prepares an empty list $\mathcal{H}_{zp}$. $\mathcal{B}$ sets $\mathsf{mpk} = (p, \mathbb{G}, g, H, H_{zp})$. $\mathcal{B}$ picks a random index $i' \in [1, q_k]$. $\mathcal{B}$ runs $(x_i, X_i, A_i) = \mathbf{KeyGen}(\mathsf{mpk})$ for all $i \in [1, q_k]$ except i'. $\mathcal{B}$ sets $X_{i'} = X^*$ and $A_{i'} = H(X^*)$. $\mathcal{B}$ sends mpk, (X_i, A_i) for $i \in [1, q_k]$ and ρ to the adversary $\mathcal{A}$.

Oracle Query. $\mathcal{B}$ answers the oracle queries as follows:

- Sign: On input a message m_j and public key X_{i_j}, if $i_j \neq i'$, $\mathcal{B}$ returns $\sigma = \mathbf{Sign}(\mathsf{mpk}, x_{i_j}, m_j)$.

  If $i_j = i'$, $\mathcal{B}$ picks some random $z, c \in \mathbb{Z}_p$ and computes $R = (g^c X^*)^{1/z}$. $\mathcal{B}$ puts $(c, (R, m))$ in the list $\mathcal{H}_{zp}$. (If the value of c is already set in $\mathcal{H}_{zp}$, $\mathcal{B}$ picks another c and repeats the previous step.) $\mathcal{B}$ returns $\sigma = (R, z)$.

- H_{zp}: On input (R, m), if $(c, (R, m))$ is in the list $\mathcal{H}_{zp}$, $\mathcal{B}$ returns c. Otherwise, $\mathcal{B}$ picks a random $c \in \mathbb{Z}_p$. $\mathcal{B}$ puts $(c, (R, m))$ in the list $\mathcal{H}_{zp}$ and returns c.

Output. Finally $\mathcal{A}$ outputs an index $i^* \in [1, q_k]$, a message m^* and a signature $\sigma^* = (R^*, z^*)$. If $i' \neq i^*$, $\mathcal{B}$ declares failure and exits. Otherwise, $\mathcal{B}$ can compute $c^* = H_{zp}(R^*, m^*)$ such that

$$X' = {R^*}^{z^*} g^{-c^*}, \quad A_{i^*} = H(X').$$

If the *always second-preimage resistance* property holds for H, then $X' = X^*$ for the random choice of ρ. Then:

$${R^*}^{z^*} = g^{c^*} X^*.$$

$\mathcal{B}$ rewinds H_{zp} to the point that (R^*, m^*) was queried, and returns a different $c' \neq c^*$. $\mathcal{B}$ eventually obtains another forgery (R^*, z') from $\mathcal{A}$. Therefore, we have

$$(g^{c^*} X^*)^{1/z^*} = (g^{c'} X^*)^{1/z'}.$$

It implies ${X^*}^{z'-z^*} = g^{c'z^* - c^*z'}$.

Next we argue that $z^* \neq z'$. Suppose on the contrary $z^* = z'$. Then $g^{c^*} X^* = g^{c'} X^*$. It leads to a contradiction with the setting that $c^* \neq c'$. Therefore we have $z^* \neq z'$.

From ${X^*}^{z'-z^*} = g^{c'z^* - c^*z'}$ and $z^* \neq z'$, we can extract $\log_g X^* = \frac{c'z^* - c^*z'}{z' - z^*}$ as the answer to the DL problem instance.

□

It is known that the Fiat-Shamir transform has non-malleability in the random oracle model [37]. Therefore, it is straightforward that our signature scheme has non-malleability. Finally, we show the security against strong known relate-key attack under the security model defined in [79].

Theorem 3.7. *Our address-based signature is* Φ^{aff}*-EUF-CM-sRKA secure for the class of affine function* Φ^{aff} *if the DL assumption holds in the random oracle model and* H *is an always second-preimage resistant hash function.*

Proof. The proof is similar to the proof of TMU-EUF-CMA. We sketch the differences below. In the setup phase, $\mathcal{B}$ samples a number of random affine functions $\phi_k(x) = a_k x + b_k$ for some $a_k, b_k \in_R \mathbb{Z}_p$. For the signing oracle query:

- Sign: On input a message m_j, public key X_{i_j} and index k, if $i_j \neq i'$, $\mathcal{B}$ returns $\sigma = \textbf{Sign}(\mathsf{mpk}, a_k x_{i_j} + b_k, m_j)$. If $i_j = i'$, $\mathcal{B}$ picks some random $z, c \in \mathbb{Z}_p$ and computes $R = (g^c (X^*)^{a_k} g^{b_k})^{1/z}$. $\mathcal{B}$ puts $(c, (R, m))$ in the list $\mathcal{H}_{zp}$. $\mathcal{B}$ returns $\sigma = (R, z)$.

The output phase is almost the same. □

3.7 Hash Functions for Address-based Signatures

From the security proofs in the previous sections, it has been demonstrated that our new address-based signature requires both the collision-resistant property and the always second-preimage resistant property of the hash function H. Although theoretically these two properties are not always the same [65], practically there are several possible instantiations of H. In this section, we contrast the differences encountered when we instantiate the hash functions differently in Bitcoin and Ethereum.

In Ethereum, the process begins by computing a keccak256 hash of the uncompressed X and storing the rightmost 20 bytes. The 20 bytes string is typically expressed as a hexadecimal string for Ethereum addresses. Importantly, Ethereum's address computation does not involve any prefix or key in the hash function. For unkeyed hash functions, the always second-preimage resistant property is equivalent to the traditional second-preimage resistant property. The latter is implied by the collision-resistant property [65]. Therefore, only the collision-resistant property is required for the security of the address-based signature with Ethereum's H function.

In Bitcoin, the public key $X = g^x$ can be expressed in either a compressed or uncompressed format. A byte of prefix (0x02/0x03/0x04) is placed in front of the public key (0x04 denotes using uncompressed X as the public key; 0x02 denotes using the x-coordinate of X and the y-coordinate is even; 0x03 denotes using the x-coordinate of X and the y-coordinate is odd). The resulting 65/33 bytes are first hashed by SHA256 and then by RIPEMD160 to obtain a 20-byte string. A byte of prefix is then added to denote the network ID (e.g., 0x00 represents Bitcoin main network, 0x6f represents a test network). This is followed by appending with 4 bytes of checksum on the previous 21 bytes. The 25-byte string is typically expressed as a base58 string for Bitcoin addresses. For Bitcoin, the choice of the prefix representing the type of public key and the prefix representing the network ID are selected by the Bitcoin core developers. Therefore, the security of the address-based signature with Bitcoin's H function requires both the collision-resistant property and the always second-preimage resistant property of H.

From a practical standpoint, the function H constructed from popular hash functions (e.g., SHA256 or keccak256) should satisfy both the collision-resistant property and the always second-preimage resistant property. However, from a theoretical perspective, the Bitcoin setting requires a slightly stronger assumption.

3.8 Generic Construction of Address-based Signatures

In this section, we present two generic constructions of address-based signatures derived from standard signatures. Subsequently, we analyze the signature size of these schemes when instantiated from other signatures, such as BLS, and BB signatures.

3.8.1 Generic Construction 1

A trivial way to construct address-based signature is to treat the encoding of the public key as address. Suppose (S.KeyGen, S.Sign, S.Verify) is a EUF-CMA secure signature scheme, in the multi-user setting. Therefore, the address function H is a simple encoding function. The first generic construction is given in Algorithm 8. For the security, the collision resistance of this generic address-based signature is trivial: it holds if H is collision resistant. The TMU-EUF-CMA security of this generic address-based signature is based on the TMU-EUF-CMA security of the underlying signature scheme. The major

disadvantage of this scheme is that the size of the address is at least as long as the public key.

Algorithm 8: Generic Construction 1

Procedure $\textsc{Setup}(1^\lambda)$**:**
  pick H as the encoding function for the public key space and H^{-1} is the reverse of H;
  return $\mathsf{mpk} = (1^\lambda, H, H^{-1})$;
Procedure $\textsc{KeyGen}(\mathsf{mpk})$**:**
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(1^\lambda)$;
  $A = H(\mathsf{pk})$;
  return $(\mathsf{sk}, \mathsf{addr} = A)$;
Procedure $\textsc{Sign}(\mathsf{mpk}, \mathsf{sk}, m)$**:**
  return $\sigma \leftarrow \mathsf{S.Sign}(\mathsf{sk}, m)$;
Procedure $\textsc{Verify}(\mathsf{mpk}, \mathsf{addr} = A, m, \sigma)$**:**
  $\mathsf{pk} = H^{-1}(A)$;
  if $1 \leftarrow \mathsf{S.Verify}(\mathsf{pk}, m, \sigma)$ **then**
    stop with $(1, \mathsf{pk})$;
  stop with $(0, \perp)$;

3.8.2 Generic Construction 2

Another generic construction of address-based signature is to send the classical signature σ_c and together with the signer public key pk to the verifier. The address is the hash of pk. The verifier checks the validity of (1) the classical signature σ_c with respect to pk and (2) pk with respect to the signer's address. Therefore, the actual data to be stored by a blockchain includes the signer's signature (σ_c, pk) and the recipient's address. The second generic construction is given in Algorithm 9. With regard to security, the collision resistance of the second generic address-based signature is straightforward: it holds if H is collision-resistant. The trapdoorless multi-user existential unforgeability under chosen message attacks (TMU-EUF-CMA) security of this generic address-based signature is predicated on the TMU-EUF-CMA security of the underlying signature scheme, as well as the always second-preimage resistant property of H. The proof is akin to the proof of our construction. The key drawback of this scheme is that the public key pk introduces an additional overhead, leading to higher transaction fees.

The case of Bitcoin. In Bitcoin, the most prevalent type of transaction is Pay-to-Public-Key-Hash (P2PKH). In a P2PKH transaction, the signature

Algorithm 9: Generic Construction 2

Procedure SETUP(1^λ)**:**
  pick hash function H for the public key space;
  return $\mathsf{mpk} = (1^\lambda, H)$;
Procedure KEYGEN(mpk)**:**
  $(\mathsf{pk}, \mathsf{sk}') \leftarrow \mathsf{S.KeyGen}(1^\lambda)$;
  $A = H(\mathsf{pk})$;
  return $(\mathsf{sk} = (\mathsf{pk}, \mathsf{sk}'), \mathsf{addr} = A)$;
Procedure SIGN($\mathsf{mpk}, \mathsf{sk} = (\mathsf{pk}, \mathsf{sk}'), m$)**:**
  $\sigma_c \leftarrow \mathsf{S.Sign}(\mathsf{sk}', m)$;
  return $\sigma = (\sigma_c, \mathsf{pk})$;
Procedure VERIFY($\mathsf{mpk}, \mathsf{addr} = A, m, \sigma = (\sigma_c, \mathsf{pk})$)**:**
  if $1 \leftarrow \mathsf{S.Verify}(\mathsf{pk}, m, \sigma_c)$ *and* $A = H(\mathsf{pk})$ **then**
    stop with $(1, \mathsf{pk})$;
  stop with $(0, \perp)$;

script contains an secp256k1 signature (Sig) and the full public key (PubKey), and it is concatenated to the pubkey script as follows:

<Sig> <PubKey> OP_DUP OP_HASH160 <PubkeyHash> OP_EQUALVERIFY OP_CHECKSIG

Therefore, the spender's public key (PubKey) is sent as a part of the input signature script [2]. It is the same as our generic construction 2.

3.8.3 Candidate Signature Schemes and their Drawbacks

We examine some popular digital signature schemes (other than Schnorr and ECDSA) that can be used with the generic constructions. This allows us to evaluate the overall performance of all combinations with generic constructions.

The key-prefixed variant of the Schnorr signature does not allow public key recovery as in the standard Schnorr signatures, since the computation of $c = H(R, M, X)$ requires knowledge of the public key X. The same is true for the EdDSA signature. The key-prefixed variant of the Schnorr signature is shown in Algorithm 10. Without public key recovery, the key-prefixed variant of the

[2] https://developer.bitcoin.org/devguide/transactions.html

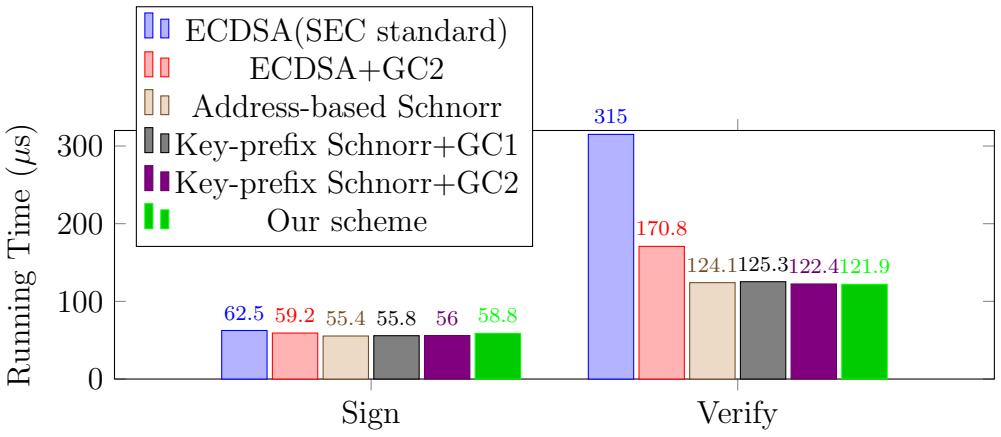


FIGURE 3.2: Comparison of running time of address-based signature.

Schnorr signature incurs a larger communication cost regardless of whether it's combined with generic construction 1 or 2, as shown in Table 3.1.

Algorithm 10: Key-prefix Schnorr Signature

Procedure SIGN(x, m)**:**
- $r \leftarrow_s \mathbb{Z}_p$;
- $R = g^r$;
- $X = g^x$;
- $c = H_{zp}(R, m, X)$;
- $z = r + cx \mod p$;
- return $\sigma = (c, z)$;

Procedure VERIFY$(X, m, \sigma = (c, z))$**:**
- $R = g^z X^{-c}$;
- **if** $c = H_{zp}(R, m, X)$ **then**
  - stop with 1;
- stop with 0;

There are a few short signature schemes based on pairings. Denote $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ as a pairing, and $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic group of prime order p. Denote g_1 and g_2 as the generator of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Consider x as a secret key, the public key is $X = g_1^x$. Denote $H_{G2} : \{0,1\}^* \rightarrow \mathbb{G}_2$ as a cryptographic hash function. For the BLS signature [13] and the BB signature [10], the verification of both signatures requires the public key. The algorithms are shown in Algorithm 4 and 5 respectively (in the appendix).

We can define $\mathbb{G}_2$ as an ECC group of 256-bit order p and obtains a compact signature of 256 bits only. However, the public key will then be defined as an ECC group of 512-bit order and the public key size will become 512 bits. As

shown in Table 3.1, the larger size of public key in BLS/BB offsets the saving in the signature size, no matter combining with generic construction 1 and 2.

3.9 Efficiency Analysis

We have implemented a variety of classical and address-based signatures. These implementations were written in Rust and tested on a MacBook with an Intel Core i5 1.4GHz processor and 16GB RAM. The results reflect the median running time for more than 300 executions.

For ECC-based schemes, they all use the same curve, secp256k1, which is the curve used in Bitcoin. For pairing-based schemes, they utilize the BN curves in the bn library in Rust.

Classical Signatures. We compare these schemes with our own and the classical signatures in Fig. 3.1. Our scheme outperforms ECDSA in both signing and verification. The Schnorr signature is the most efficient in terms of signing, since no modular inverse computation is involved. The verification time of our scheme is comparable to that of the Schnorr signature.

As indicated in Table 3.1, there are several drawbacks when using the Schnorr signature or ECDSA. Thus, our scheme presents a more favorable alternative when compared to the Schnorr signature or ECDSA.

Address-based Signatures. We have implemented several address-based signatures based on our generic constructions. The comparison with our scheme is shown in Fig. 3.2. For comparison, we also include the address-based Schnorr signature (provided in the Appendix). In general terms, the computational performance of our scheme aligns with the family of Schnorr signatures. Particularly, the ECDSA following the SEC standard for public key recovery is slower due to the repetition in the verification process.

The schemes constructed from BB and BLS signatures are significantly slower, and thus we do not include them in the figure. The running time for the BB signature is 2083 μs and 12503 μs for signing and verification, respectively. The running time for the BLS signature is 4610 μs and 13433 μs for signing and verification, respectively. The address-based versions (by applying GC1 or GC2) of BB and BLS signatures are slightly slower. Hence, they are at least 30 times slower than other schemes in signing and approximately 40 times slower in verification.

Advantages over existing CA approach. In the current Internet, a trusted certificate authority (CA) is employed to provide certificates to websites for safeguarded SSL/TLS connections. This certificate links the website's

address with its associated public key. In reality, there are instances where deceptive Certificate Authorities (CAs) or intermediary CAs grant certificates to hackers. The challenge of maintaining a secure chain of trust is a significant hurdle in the existing Internet framework. However, by leveraging the hash of the public key in an XIA architecture, we can achieve intrinsic security that doesn't hinge on the issuance of certificates by a trusted third party.

From an efficiency standpoint, the overhead of using a certificate with a standard signature is considerable. A typical X.509 certificate is around 2-4 Kb, which significantly exceeds the size of a single signature (approximately 512 bits for an ECDSA/Schnorr signature). By deploying address-based signatures within an XIA framework, we can substantially reduce the overhead associated with certificates.

3.10 Conclusion

In this work, we introduce the concept of an "address-based signature", which represents systems that employ a concise address for signature verification. Systems such as XIA and blockchain utilize the hash of the public key as the address.

We define the security models for address-based signatures and propose two generic constructions which can be instantiated by existing signature schemes like Schnorr, ECDSA, BB/BLS.

We propose a novel address-based signature, which outperforms the aforementioned schemes in terms of security or total output size (i.e., the size of the signature and address).

Chapter 4

Bandwidth-Efficient Zero-Knowledge Proofs for Threshold ECDSA

4.1 Motivation

Threshold signatures [51] enable multiple parties to collaboratively sign a message, and the output signature can be authenticated with a single public key. Threshold ECDSA signatures can be effortlessly integrated into existing blockchain systems that utilize ECDSA signatures. In Bitcoin, users can generate a "multisig" address composed of n public keys to facilitate threshold signing in a straightforward manner: t ECDSA signatures are produced, and the signatures are deemed valid only if they can be verified by any t out of n public keys in the address. However, this direct approach results in a large signature size ($O(t)$), leading to increased transaction costs. Therefore, threshold ECDSA signatures offer a valuable alternative to the existing "multisig" addresses in Bitcoin.

Many existing threshold ECDSA signatures [23, 23, 42, 43, 56, 78] use an additive homomorphic encryption with zero-knowledge proofs as the building block. Some schemes [42, 43, 56] use the Paillier encryption [64]. However, there is a mismatch between the size of the message space for the Paillier encryption and the order of the elliptic curve group utilized by ECDSA. Hence, a range proof is required to ensure the encrypted message remains within an appropriate range. This range proof, however, is bandwidth-intensive. A novel additive homomorphic encryption has been proposed by Castagnos and Laguillaumie [26] (referred to as CL encryption), which can be implemented in a class group of imaginary quadratic order. A key advantage of CL encryption

is that the size of the message space can be set to the order of the elliptic curve group. Some recent threshold ECDSA schemes [23, 78] have employed CL encryption to reduce the communication bandwidth, eliminating the need for an additional range proof. However, this approach does involve a trade-off, as it increases the computational load for the signers.

The increased complexity of the number of Zero-Knowledge (ZK) proofs in [21] is due to their ZK proofs (other than those for $\mathcal{R}_{\mathsf{mod}}$ and $\mathcal{R}_{\mathsf{prm}}$) needing to use the verifier's public Ring-Pedersen parameters. As a result, one relation proved to n different receivers generates n different ZK proofs. In contrast, such a concern doesn't exist in the CL setting, which further contributes to bandwidth reduction.

Another limiting factor of [21] is the complex Zero-Knowledge (ZK) proofs for $\mathcal{R}_{\mathsf{mod}}$ and $\mathcal{R}_{\mathsf{prm}}$, represented as $\pi_{\mathsf{mod}}, \pi_{\mathsf{prm}}$ respectively, which are used in both key generation and key refreshment. They require $m = 80$ challenge and verification cycles to achieve a soundness error of 2^{-80}, resulting in substantial computation and communication overheads. Moreover, π_{mod} must be executed twice due to their extraction requirements, a concern which the CL setting does not have.

4.2 Our Contributions

The contribution of this work is three-fold.

Contribution 1: Improve the zero-knowledge proof for the discrete logarithm and the well-formedness of a CL ciphertext. We improve the zero-knowledge proofs for discrete logarithm (and its generalization) and the well-formedness of a CL ciphertext in [78].

We use a single factorization to replace the two round factorization used in the argument of knowledge in [78]. Roughly speaking, the verifier picks a random prime $\ell < q$ as a challenge, and the prover computes d' and $e' \in [0, \ell - 1]$ such that $x = d'q\ell + e'$. The prover now sends $D' = g^{d'}$ and e' to the verifier to check if $D'^{q\ell} g^{e'} = w$. This method can also avoid adversary generating forged proof which can pass the verification mentioned in [78], without the need of doubling the proof size and the running time of [11]. This argument of knowledge can be extended to a zero-knowledge proof.

Consequently, the size of the proof is decreased by a minimum of 32%, and the running time for the prover is cut down by at least 29%. These innovative zero-knowledge proofs can be directly incorporated into the two threshold ECDSA settings in [78], resulting in enhanced efficiency.

The technical obstacle is that we need to handle the soundness proof differently. In previous works [11, 78], the witness x is extracted by rewinding multiple times to obtain (x_i, e_i, ℓ_i) such that $x_i \equiv e_i \bmod \ell_i$. By the Chinese Remainder Theorem (CRT), we can extract a unique witness $x < \prod_i \ell_i$ such that $x \equiv e_i \bmod \ell_i$. By using our single factorization proof, we get $x_i \equiv e'_i \bmod q\ell_i$. They cannot be combined by the CRT directly. We further split the equations into mod q and mod ℓ_i. Since q and ℓ_i are set as distinct primes, these split equations can be combined by the CRT to extract the required witness x. Hence we can achieve soundness in our ZK proof of DL relation in G. We also apply this technique in our ZK proof for the relation $\mathcal{R}_{\mathsf{enc}}$ (knowing the plaintext encrypted in a CL ciphertext) and its variant $\mathcal{R}_{\mathsf{log}}$ (knowing the plaintext of a CL ciphertext is equal to the discrete logarithm of an element in the ECC group).

Contribution 2: Present new zero-knowledge proof for the affine transformation over the CL ciphertext. We give new ZK proofs related to the affine transformation on a CL ciphertext. Denote C as a CL ciphertext for a plaintext m. Since the CL encryption is additive homomorphic, anyone can turn C into a ciphertext C' which is the encryption of $Am + B$, where A, B is an integer. The prover who perform this transformation can generate a ZK proof for the witness (A, B). We give a compact ZK proof for the affine transformation relation $\mathcal{R}_{\mathsf{aff}}$, as well as two related relations $\mathcal{R}_{\mathsf{aff-p}}$ and $\mathcal{R}_{\mathsf{aff-g}}$ defined in [21]. This new ZK proof can be used for threshold ECDSA with identifiable abort [43], or UC-secure threshold ECDSA [21].

Contribution 3: Applying our ZK into UC, non-interactive threshold ECDSA. We demonstrate the advantage of our bandwidth-efficient ZK proofs by transforming the Paillier-based threshold ECDSA [21] into CL-encryption-based[1]. The resulting scheme is UC-secure (composable security), non-interactive (one round online signing phase) and proactive (periodic key refresh).

By using the Paillier encryption [64], the threshold ECDSA in [21] uses a lot of ZK proofs related to the Paillier encryption in the pre-signing phase and the key refresh phase. In the pre-signing phase involving n parties, there are $n(n-1)$ ZK proofs for $\mathcal{R}_{\mathsf{enc}}$; and $2n(n-1)$ ZK proofs for the discrete logarithm in the Paillier group, i.e., $\mathcal{R}_{\mathsf{log}}$ in [21]. They can be directly changed to $3n$ ZK proofs in our CL-based counterpart if the CL encryption is employed. The bandwidth and computation complexity for these ZK proofs are lowered from $O(n^2)$ to $O(n)$. There are still $2n(n-1)$ ZK proofs for the affine transformation of the Paillier/CL ciphertext in the pre-signing phase. In the key refresh phase

[1] We note that [20] is the combination of UC-secure threshold ECDSA [21] and threshold ECDSA with identifiable abort [43]. Since the main goal of this work is about bandwidth efficiency, we do not include identifiable abort for optimal performance.

involving n parties, [21] used $2n$ ZK proofs for showing a modulus is Paillier-Blum denoted by $\mathcal{R}_{\mathsf{mod}}$, n ZK proofs that s belongs to the multiplicative group generated by t in the Paillier group denoted by $\mathcal{R}_{\mathsf{prm}}$, and $n(n-1)^2$ ZK proofs for the Paillier version of $\mathcal{R}_{\mathsf{log}}$. The overall complexity is $O(n^3)$. When adopting the CL encryption, they can be replaced by only $n(n-1)$ ZK proofs for $\mathcal{R}_{\mathsf{log}}$. The overall complexity is lowered to $O(n^2)$ for both the bandwidth as well as computation. Similarly, in the key generation phase, we replace the two ZK proofs for $\mathcal{R}_{\mathsf{mod}}$ and one ZK proof for $\mathcal{R}_{\mathsf{prm}}$ with a single ZK proof for $\mathcal{R}_{\mathsf{RepS}}$ described in section 3.1.

From a computation time perspective, we can significantly reduce the running time of the key generation and key refresh algorithms by substantially lowering the total number of zero-knowledge (ZK) proofs. However, we note that the computation time related to the group used by the CL encryption is higher than the computation time related to the Paillier group. Hence, the running time of our PreSign algorithm is 10+ times more expensive than the Paillier-based counterpart [21]. Nevertheless, the online signing time is still the same as [21]. The running time during the online phase is generally considered to be more critical for the online/offline signature.

4.3 ZK Proofs for HSM Group with Trustless Setup

In this section, we will provide a series of Zero-Knowledge (ZK) proofs for relations that are essential in the construction of threshold ECDSA. These proofs are primarily associated with the discrete logarithm and the ciphertext of the CL encryption.

4.3.1 ZK Proof for Multi-exponentiation

We now construct an argument of knowledge for the following *representation* relation:

$$\mathcal{R}_{\mathsf{RepS}} = \{w \in G; \vec{x} \in \mathbb{Z}^n : w = \prod_{i=1}^{n} g_i^{x_i}\},$$

where $g_1, \ldots, g_n \in G \setminus F$ are in the CRS (common reference string) $\mathcal{G}_{\mathsf{HSM}}$. This is the generalization of the discrete logarithm relation (in which $n = 1$). The ZK proof is given in Algorithm 11.

[2]To reduce the round complexity, we can set $c = H_1(R)$, $\ell = H_2(c)$, where H_1, H_2 are hash functions which output a number in $[0, q-1]$ and a suitable prime number respectively.

Algorithm 11: Protocol $\mathsf{ZKPoKRepS}$ for the relation $\mathcal{R}_{\mathsf{RepS}}$

Param: $\mathcal{G}_{\mathrm{HSM}} \leftarrow \mathsf{GGen}_{\mathrm{HSM},q}(1^\lambda)$, $g_1, \ldots, g_n \in G \setminus F$, $B = 2^{\epsilon_d+2\lambda} n q^2 \tilde{s}$ where $\epsilon_d = 80$.

Input: $w \in G$.

Witness: $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{Z}^n$.

1 Prover chooses $k_1, \ldots, k_n \xleftarrow{\$} [-B, B]$, $t \xleftarrow{\$} \mathbb{Z}_q$, sends $R = \prod_{i=1}^n g_i^{k_i}$ to verifier.

2 Verifier sends $c \xleftarrow{\$} [0, q-1]$ to the prover. Prover aborts if $c \notin [0, q-1]$.

3 Verifier sends $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$ to the prover.[2]

4 Prover computes $s_i = k_i + c x_i$ for $i \in [1, n]$. Prover finds $d_i \in \mathbb{Z}$ and $e_i \in [0, q\ell - 1]$ s.t. $s_i = d_i q\ell + e_i$ for $i \in [1, n]$. Prover sends $D = \prod_{i=1}^n g_i^{d_i}$ and $\vec{e} = (e_1, \ldots, e_n)$ to the verifier.

5 Verifier accepts if $e_1, \ldots, e_n \in [0, q\ell - 1]$ and $D^{q\ell} \prod_{i=1}^n g_i^{e_i} = R w^c$.

Theorem 4.1. *Protocol* $\mathsf{ZKPoKRepS}$ *is an argument of knowledge for* $\mathcal{R}_{\mathsf{RepS}}$ *in the generic group model.*

Proof. Let $(\ell_1, D_1, \vec{e}_1)$ and $(\ell_2, D_2, \vec{e}_2)$ be two accepting transcripts for $\mathsf{ZKPoKRepS}$. We first prove a claim.

Claim. With overwhelming probability for fixed (q, R, c) there exists $\vec{\alpha}$, $\vec{\beta}$ and $\vec{x} = (x_1, ..., x_n)$ such that $\vec{x} = \vec{\alpha} q\ell_1 + \vec{e}_1 = \vec{\beta} q\ell_2 + \vec{e}_2$ and $Rw^c = Rep(\vec{x}) = \prod_{i=1}^n g_i^{x_i}$, and each x_i for $i \in [1, n]$ is bounded by 2^{q_2}.

Proof. By the verification equation of this protocol, we have $D_1^{q\ell_1} Rep(\vec{e}_1) = D_2^{q\ell_2} Rep(\vec{e}_2) = Rw^c$. With overwhelming probability the generic group adversary knows $\alpha_1, ..., \alpha_m$ and $\beta_1, ..., \beta_m$ and $m > n$ such that $D_1 = f^\gamma \prod_{i=1}^m g_i^{\alpha_i}$ and $D_2 = f^\delta \prod_{i=1}^m g_i^{\beta_i}$. By lemma 2.3 with overwhelming probability $\gamma\ell_1 = \delta\ell_2, \alpha_i q\ell_1 + \vec{e}_1[i] = \beta_i q\ell_2 + \vec{e}_2[i]$ for each $i \leq n$ and $\alpha_i \ell_1 = \beta_i \ell_2$ for each $i \in [n+1, m]$ and thus $\ell_1 | \beta_i \ell_2$ for each $i \in [n+1, m]$. We note that ℓ_1 and ℓ_2 are co-prime unless $\ell_1 = \ell_2$ which happens with probability $\frac{\lambda \ln 2}{2^\lambda}$, and that $\alpha_i \leq 2^{q_2}$ and α_i is chosen before ℓ_2 is sampled. Hence, with overwhelming probability $\alpha_i = \beta_i = 0$ for each $i \in [n+1, m]$ in which case $Rw^c = \prod_{i=1}^n g_i^{\alpha_i q\ell_1 + \vec{e}_1[i]} = \prod_{i=1}^n g_i^{\beta_i q\ell_2 + \vec{e}_2[i]}$. Note that f is cancelled out by q-th exponentiation for D_1 and D_2. Setting $\vec{\alpha} = (\alpha_1, ..., \alpha_n)$ and $\vec{\beta} = (\beta_1, ..., \beta_n)$ we conclude with overwhelming probability $Rw^c = Rep(\vec{\alpha} q\ell_1 + \vec{e}_1) = Rep(\vec{\beta} q\ell_2 + \vec{e}_2)$. Finally, if $\mathcal{A}$ has made at most q_2 queries to $\mathcal{O}_2$ then $\alpha_i < 2^{q_2}$ and $\beta_i < 2^{q_2}$ for each i. The claim holds. □

Consider any third transcript, w.l.o.g. $(\ell_3, D_3, \vec{e}_3)$. Involving this claim again, there exists $\vec{\alpha'}, \vec{\beta'}, \vec{x'}$ in $\mathbb{Z}^n$ such that $\vec{x'} = \vec{\alpha'} q\ell_2 + \vec{e}_2 = \vec{\beta'} q\ell_3 + \vec{e}_3$. Thus with overwhelming probability $\vec{x'} - \vec{x} = (\vec{\alpha'} - \vec{\beta}) q\ell_2$. However, since ℓ_2 is sampled randomly from an exponentially large set of primes independent of $\vec{e}_1, \vec{e}_3, \ell_1$ and ℓ_3, (which fix the value of $\vec{x'} - \vec{x}$) there is a negligible probability that $\vec{x'} - \vec{x} = 0 \mod q\ell_2$, unless $\vec{x'} - \vec{x} = 0$. We draw a conclusion that by a simple union bound over the $\mathsf{poly}(\lambda)$ number of transcripts, for any polynomial number of accepting transcripts $\{(\ell_i, D_i, \vec{e}_i)\}_{i=1}^{\mathsf{poly}(\lambda)}$ there exists a single $\vec{x}$ such that $\vec{x} = \vec{e}_i \mod q\ell_i$ for all i.

Now we describe the extractor Ext:

1. Run $\mathcal{A}_0$ to get output (w, state).
2. Let $\mathcal{L} \leftarrow \{\}$. Run Step 1 of Protocol $\mathsf{ZKPoKRepS}$ with $\mathcal{A}_1$ on input (w, state).
3. Run Step 2-4 of Protocol $\mathsf{ZKPoKRepS}$ with $\mathcal{A}_1$, sampling fresh randomness c and ℓ for the verifier. If the transcript $(R, c, \ell, D, \vec{e})$ is accepting, set $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\vec{e}, \ell)\}$, and otherwise repeat this step.
4. Compute $\vec{e}'_0 = \vec{e}_1 \mod q$ and $\vec{e}'_i = \vec{e}_i \mod \ell_i$ for each $(\vec{e}_i, \ell_i) \in \mathcal{L}$. Compute by Chinese Remainder Theorem (CRT) $\vec{s} = (s_1, \ldots, s_n)$ such that $\vec{s} = \vec{e}'_0 \mod q$ and $\vec{s} = \vec{e}'_i \mod \ell_i$ for each $(\vec{e}_i, \ell_i) \in \mathcal{L}$. If $\prod_{i=1}^n g_i^{s_i} \neq Rw^c$, return to Step 3.
5. Consider the intermediate transcript as $(R, c, \vec{s})$. Run from step 2 for the second time and obtain $(R, c', \vec{s'})$.
6. Compute $\Delta_{s_i} = s_i - s'_i$ for $i \in [1, n]$ and $\Delta_c = c - c'$. Output $\vec{x} = (x_1, \ldots, x_n)$ for $x_i = \Delta_{s_i}/\Delta_c$.

Analysis for Step 4. Suppose that after some polynomial number of rounds the extractor has obtained M accepting transcripts $(R, c, \ell_i, D_i, \vec{e}_i)$ for independent values of $\ell_i \xleftarrow{\$} \mathsf{Primes}(\lambda)$, by the claim above, with overwhelming probability there exists $\vec{s} = (s_1, \ldots, s_n) \in \mathbb{Z}^n$ such that $\vec{s} = \vec{e}_i \mod q\ell_i$, and $\prod_{i=1}^n g_i^{s_i} = Rw^c$, $s_j < 2^{q_2}$ for $j \in [n]$. Hence, the CRT algorithm used in Step 4 will recover the required vector $\vec{s}$ once $|\mathcal{L}| > q_2$. Since a single round of interaction with $\mathcal{A}_1$ results in an accepting transcript with probability $\epsilon \geq 1/\mathsf{poly}(\lambda)$, in expectation the extractor obtains $|\mathcal{L}| > q_2$ accepting transcripts for independent primes ℓ_i after $q_2 \cdot \mathsf{poly}(\lambda)$ rounds. Hence, Ext output $\vec{s}$ such that $\prod_{i=1}^n g_i^{s_i} = Rw^c$ in expected polynomial time.

Analysis for Step 6. It remains to argue that Ext succeeds with overwhelming probability in Step 6. W.l.o.g., assume that $c > c'$, by Step 5, we have

$\prod_{i=1}^{n} g_i^{s_i} \cdot w^{-c} = \prod_{i=1}^{n} g_i^{s'_i} \cdot w^{-c'} = R$. Denote $\Delta_c = c' - c, \Delta_s = s_i - s'_i$ for $i \in [0, n]$. We have $\prod_{i=1}^{n} g_i^{\Delta_{s_i}} = w^{\Delta_c} = (\prod_{i=1}^{m} g_i^{\alpha'_i} \cdot f^{\gamma'})^{\Delta_c}$ for some $\alpha'_i \in \mathbb{Z}$ and $\gamma' \in \mathbb{Z}_q$ by Lemma 2.2. By Lemma 2.3, $\Delta_{s_i} = \alpha'_i \Delta_c$ for $i \in [1, n]$, $\alpha'_i = 0$ for $i \in [n+1, m]$ and $\gamma' = 0 \bmod q$ with overwhelming probability. If $\mu = \prod_{i=1}^{n} g_i^{\Delta_{s_i}/\Delta_c} \neq w$, then $\mu^{\Delta_c} = w^{\Delta_c}$. It follows that μ/w is an element of order $1 < \Delta_c < q$. By Corollary 2.5, the probability of finding a non-trivial order of $\mu/w \neq 1$ is negligible. Hence, $\mu = w$ with overwhelming probability. It implies that $\Delta_{s_i}/\Delta_c \in \mathbb{Z}$ for all i. Hence, the witness $\vec{x} = (x_1, \ldots, x_n)$ can be extracted as in Step 6. □

Theorem 4.2. *The protocol* ZKPoKRepS *is an honest-verifier statistically zero-knowledge argument of knowledge for relation* $\mathcal{R}_{\mathsf{RepS}}$ *in the generic group model.*

Proof. The simulator Sim picks a random challenge $c' \xleftarrow{\$} [0, q-1]$ and $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$. It picks random $d'_1, \ldots, d'_n \xleftarrow{\$} [0, B-1]$, $e'_1, \ldots, e'_n \xleftarrow{\$} [0, q\ell - 1]$. It computes: $D' = \prod_{i=1}^{n} g_i^{d'_i}, \quad R' = D'^{q\ell} \prod_{i=1}^{n} g_i^{e'_i} \cdot w^{-c'}$.

We argue that the transcript $(R', c', (D', \vec{e'} = (e'_1, \ldots, e'_n)), \ell')$ is indistinguishable from a real transcript between a prover and a verifier. Sim chooses ℓ', c' identically to the honest verifier. R' is uniquely determined by the other values such that the verification holds.

We must show that in the real protocol, independent of ℓ and c, the values in $\vec{e}$ have a negligible statistical distance from the uniform distribution over $[0, q\ell - 1]$ and each $g_i^{d_i}$ has a negligible statistical distance from uniform over G. In addition we must argue that D and $\vec{e}$ are independent. For this we use the following facts, which are easy to verify:

1. Fact 1: If Z is a uniform random variable over N consecutive integers and $m < N$, then $Z \bmod m$ has a statistical distance at most m/N from the uniform distribution over $[0, m-1]$.

2. Fact 2: For independent random variables X_1, X_2, Y_1, Y_2, the distance between the joint distributions (X_1, X_2) and (Y_1, Y_2) is at most the sum of statistical distances of X_1 from Y_1 and X_2 from Y_2. Similarly, if these variables are group elements in G, the statistical distance between $X_1 \cdot X_2$ and $Y_1 \cdot Y_2$ is no greater than the sum of statistical distances of X_1 from Y_1 and X_2 from Y_2.

3. Fact 3: Consider random variables X_1, X_2, Y_1, Y_2 with statistical distances $s_1 = \Delta(X_1, Y_1)$ and $s_2 = \Delta(X_2, Y_2)$, where $Pr(X_1 = a | X_2 = b) < Pr(X_1 = a) + \epsilon_1$ and $Pr(Y_1 = a | Y_2 = b) < Pr(Y_1 = a) + \epsilon_2$ for all values

> a, b. Then the joint distributions (X_1, X_2) and (Y_1, Y_2) have a statistical distance at most $s_1 + s_2 + \epsilon_1|\text{supp}(X_2)| + \epsilon_2|\text{supp}(Y_2)|$, where supp is the support.

Consider fixed values of $c, x_1, \ldots, x_n$ and ℓ. In the real protocol, the prover computes $s_i = k_i + cx_i$, where k is uniform in $[-B, B]$ and t is uniform in $\mathbb{Z}_q$, and sets $e_i = s_i \bmod q\ell$. By Fact 1, the value of s_i is distributed uniformly over a range of $2B + 1$ consecutive integers, thus e_i has a statistical distance at most $q\ell/(2B + 1)$ from uniform over $[0, q\ell - 1]$. This bounds the distance between the real e_i and the simulated e'_i which is uniform over $[0, q\ell - 1]$.

Next, we show that each $g_i^{d_i}$ is statistically indistinguishable from uniform in the subgroup generated by g_i (denoted as G_i). The distribution of $g_i^{d_i}$ over G_i is determined by the distribution of $d_i \bmod |G_i|$. Consider the distribution of $d_i = \lfloor \frac{s_i}{q\ell} \rfloor$ over the consecutive integers in $[\lfloor \frac{cx_i - B}{q\ell} \rfloor, \lfloor \frac{cx_i + B}{q\ell} \rfloor]$. Denote this by the random variable Z. The probability that $d_i = z$ is the probability that s_i falls in the interval $[zq\ell, (z+1)q\ell - 1]$. Hence, $Pr[d_i = z] = q\ell/(2B+1)$ for all $z \in Z$ if $zq\ell \geq cx_i - B$ and $(z + 1)q\ell - 1 \leq cx_i + B$. This probability may or may not hold for the two endpoints $E_1 = \lfloor \frac{cx_i - B}{q\ell} \rfloor$ and $E_2 = \lfloor \frac{cx_i + B}{q\ell} \rfloor$. Denote Y as the set of points with $Pr[d_i = z] = q\ell/(2B + 1)$ only. The distance of d_i from a uniform random variable U_Y over Y is largest when the number of possible s_i mapping to E_1 and E_2 are both $q\ell - 1$, i.e. $cx_i - B = 1 \bmod q\ell$ and $cx_i + B = q\ell - 2 \bmod q\ell$. In this case, d_i is one of the two endpoints outside Y with probability $\frac{2(q\ell-1)}{2B+1}$. As $|Y| = \frac{2B+3}{q\ell} - 3$, the statistical distance of d_i from U_Y is at most $\frac{1}{2}(|Y|(\frac{1}{|Y|} - \frac{q\ell}{2B+1}) + \frac{2(q\ell-1)}{2B+1}) = \frac{5q\ell-4}{2(2B+1)} < \frac{2q\ell}{B} \leq \frac{2^{\lambda+1}q}{B}$. Moreover, the statistical distance of $d_i \bmod |G_i|$ from $U_Y \bmod |G_i|$ is no larger. By Fact 1, $U_Y \bmod |G_i|$ has a statistical distance at most $\frac{|G_i|}{|Y|} \leq \frac{2^\lambda q|G|}{2B+3-3q\cdot 2^\lambda}$. By the triangle inequality, the statistical distance of $d_i \bmod |G_i|$ from uniform is at most $\frac{2^{\lambda+1}q}{B} + \frac{2^\lambda q|G|}{2B+3-3q\cdot 2^\lambda}$. This also bounds the distance of $g_i^{q_i}$ from uniform in G_i. The simulated value q'_i is uniformly chosen from a set of size B. Again by Fact 1, if $|G_i| < B$, then $d'_i \bmod |G_i|$ has a distance $|G_i|/B \leq |G|/B$ from uniform. The simulated value $g_i^{d'_i}$ has a distance at most $|G|/B$ from uniform in G_i. By the triangle inequality, the statistical distance of $g_i^{d_i}$ and $g_i^{d'_i}$ is at most: $\frac{2^{\lambda+1}q}{B} + \frac{2^\lambda q|G|}{2B+3-3q\cdot 2^\lambda} + \frac{|G|}{B} < \frac{(2^\lambda q+2)|G|+2^{\lambda+2}q}{2B+3-3q\cdot 2^\lambda} \leq \frac{1}{n2^{\epsilon_d+1}}$ if $B \geq n2^{\epsilon_d}(2^\lambda q + 2)|G| + nq \cdot 2^{\epsilon_d+\lambda+2} + 3q \cdot 2^{\lambda-1} - \frac{3}{2}$ for some distance parameter ϵ_d.

Finally, we consider the joint distribution of $g_i^{d_i}$ and e_i. Consider the conditional distribution of $d_i|e_i$. Note that $d_i = z$ if $(s_i - e_i)/(q\ell) = z$. We repeat a similar argument as above for bounding the distribution of d_i from uniform. For each possible value of z, there always exists a unique value of s_i such that $\lfloor \frac{s_i}{q\ell} \rfloor = z$ and $s_i = 0 \bmod q\ell$, except possibly at the two endpoints E_1, E_2 of the range of d_i. When e_i disqualifies the two points E_1 and E_2, then each of the remaining points $z \notin \{E_1, E_2\}$ still has an equal probability mass, and

thus the probability $Pr(d_i = z|e_i)$ increases by at most $\frac{1}{|Y|} - \frac{q\ell}{2B+1}$. The same applies to the variable $d_i|e_i \bmod |G_i|$ and hence the variable $g^{d_i}|e_i$.

We can compare the joint distribution $X_i = (g_i^{d_i}, e_i)$ to the simulated distribution $Y_i = (g_i^{d'_i}, e'_i)$ using Fact 3. Setting $\epsilon_1 = \frac{1}{|Y|} - \frac{q\ell}{2B+1}$ and $\epsilon_2 = 0$, the distance between these joint distributions is at most $\frac{1}{n2^{\epsilon_d+1}} + \frac{q\ell}{2B+1} + \epsilon_1 q\ell = \frac{1}{n2^{\epsilon_d+1}} + \frac{q^2\ell^2}{2B+3-3q\ell} + \frac{q\ell(1-q\ell)}{2B+1}$. Moreover, as each X_i is independent from X_j for $i \neq j$, we use Fact 2 to bound the distance between joint distributions $(g_1^{d_1}, \ldots, g_n^{d_n}, e_1, \ldots, e_n)$ and $(g_1^{d'_1}, \ldots, g_n^{d'_n}, e'_1, \ldots, e'_n)$ by the sum of individual distances between each X_i and Y_i, which is at most: $\frac{1}{2^{\epsilon_d+1}} + \frac{nq^2\ell^2}{2B+3-3q\ell} + \frac{nq\ell(1-q\ell)}{2B+1} < \frac{1}{2^{\epsilon_d+1}} + \frac{nq^2\ell^2}{2B+3-3q\ell} < \frac{1}{2^{\epsilon_d}}$, where the last equality holds if $B \geq 2^{\epsilon_d+2\lambda}nq^2 + 3q \cdot 2^{\lambda-1} - \frac{3}{2}$. Finally, this also bounds the distance between $(D, \vec{r})$ and $(D', \vec{r'})$, where $D = \prod_{i=1}^n g_i^{d_i}$ and $D' = \prod_{i=1}^n g_i^{d'_i}$. Combining the two requirements on B (Recall the first requirement is $B \geq n2^{\epsilon_d}(2^\lambda q + 2)|G| + nq \cdot 2^{\epsilon_d+\lambda+2} + 3q \cdot 2^{\lambda-1} - \frac{3}{2}$, which can be further required by $B \geq nq \cdot 2^{\epsilon_d+\lambda+1}|G|$; and $|G| = q \cdot s > q$), we obatin a simplified requirement $B \geq nq \cdot 2^{\epsilon_d+2\lambda}|G| = 2^{\epsilon_d+2\lambda}nq^2\tilde{s}$.

□

4.3.2 ZK Proof for the Well-formedness of a CL Ciphertext

Consider a prover encrypted a message $m \in \mathbb{Z}_q$ using a randomness $\rho \in [0, S]$, for some honestly generated public key $\mathsf{pk} \in G^q$[3]. We present a zero-knowledge proof of knowledge of the following relation:

$$\begin{aligned}\mathcal{R}_{\mathsf{Enc}} = &\{(\mathsf{pk}, C_1, C_2); (m, \rho) | C_1 \in G \setminus F; C_2, \mathsf{pk} \in G^q; \\ &\rho \in [0, S]; m \in \mathbb{Z}_q : C_1 = f^m \mathsf{pk}^\rho \wedge C_2 = g_q^\rho\}.\end{aligned}$$

For the relation $\mathcal{R}_{\mathsf{Enc}}$, we cannot apply the protocol ZKPoKRepS directly since $f \in F$. We propose a new ZK proof ZKPoKEnc for $\mathcal{R}_{\mathsf{Enc}}$ in Algorithm 12.

Theorem 4.3. *The protocol* ZKPoKEnc *is an argument of knowledge in the generic group model.*

Proof. We rewind the adversary on fresh challenges ℓ so that each accepting transcript outputs an (e_ρ, ℓ), where $\rho^* = e_\rho \bmod q\ell$ by ZKPoKRepS with overwhelming probability. We consider the following two cases:

[3]This condition holds if there is another explicit ZK proof of knowing $\log_{g_q} \mathsf{pk}$ generated by the owner of the decryption key, or pk is honestly generated by the verifier himself.

Algorithm 12: Protocol $\mathsf{ZKPoKEnc}$ for the relation $\mathcal{R}_{\mathsf{Enc}}$

Param: $\mathcal{G}_{\mathrm{HSM}} \leftarrow \mathsf{GGen}_{\mathrm{HSM},q}(1^\lambda)$, $B = 2^{\epsilon_d+\lambda+2}q\tilde{s}$ where $\epsilon_d = 80$.
Input: $C_1 \in G \setminus F; C_2, \mathsf{pk} \in G^q$.
Witness: $\rho \in [0, S], m \in \mathbb{Z}_q$, where $S = \tilde{s} \cdot 2^{\epsilon_d}$.

1 Prover chooses $s_\rho \xleftarrow{\$} [-B, B]$, $s_m \xleftarrow{\$} \mathbb{Z}_q$ and computes:

$$S_1 = \mathsf{pk}^{s_\rho} f^{s_m}, \quad S_2 = g_q^{s_\rho}.$$

Prover sends (S_1, S_2) to the verifier.

2 Verifier sends $c \xleftarrow{\$} [0, q-1]$ to the prover. Prover aborts if $c \notin [0, q-1]$.

3 Verifier sends $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$ to the prover.

4 Prover computes:

$$u_\rho = s_\rho + c\rho, \quad u_m = s_m + cm \mod q.$$

Prover finds $d_\rho \in \mathbb{Z}$ and $e_\rho \in [0, q\ell - 1]$ s.t. $u_\rho = d_\rho q\ell + e_\rho$. Prover computes:

$$D_1 = \mathsf{pk}^{d_\rho}, \quad D_2 = g_q^{d_\rho}.$$

Prover sends (u_m, D_1, D_2, e_ρ) to the verifier.

5 Verifier accepts if $e_\rho \in [0, q\ell - 1]$ and:

$$D_1^{q\ell}\mathsf{pk}^{e_\rho} f^{u_m} = S_1 C_1^c, \quad D_2^{q\ell} g_q^{e_\rho} = S_2 C_2^c.$$

Case 1. If $\mathsf{pk}^{\rho^*} \neq S_1 C_1^c f^{-u_m}$ and $(\mathsf{pk}^{\rho^*})^q \neq (S_1 C_1^c f^{-u_m})^q$, then we have $\mathsf{pk}^{\rho^*} \neq S_1 C_1^c f^{-u_m} = D_1^{q\ell}\mathsf{pk}^{e_\rho}$. Let $\gamma_\rho = \frac{e_\rho - \rho^*}{\ell}$. Then $D_1^q \mathsf{pk}^{\gamma_\rho}$ is an ℓ-th root of $(S_1 C_1^c f^{-u_m})/\mathsf{pk}^{\rho^*} \neq 1$. This breaks the adaptive root subgroup assumption since $(S_1 C_1^c f^{-u_m})^q/(\mathsf{pk}^{\rho^*})^q \neq 1$.

Case 2. If $\mathsf{pk}^{\rho^*} \neq S_1 C_1^c f^{-u_m}$ and $(\mathsf{pk}^{\rho^*})^q = (S_1 C_1^c f^{-u_m})^q$, then $S_1 C_1^c = \mathsf{pk}^{\rho^*} f^{\delta'}$ for some $\delta' \neq u_m \in \mathbb{Z}_q$. Then we have $S_1 C_1^c f^{-u_m} = \mathsf{pk}^{\rho^*} f^{\delta' - u_m} = D_1^{q\ell}\mathsf{pk}^{e_\rho}$. Observe that $\mathsf{pk}^{\rho^*} f^{\delta' - u_m}$ cannot cancel element f because $|\delta' - u_m| < q$. Instead, $D_1^{q\ell}\mathsf{pk}^{e_\rho}$ cancel out f and thus lies in G^q, which leads to contradiction.

Hence, by Corollary 2.4, $\mathsf{pk}^{\rho^*} f^{u_m} = S_1 C_1^c$ with overwhelming probability.

By rewinding, the extractor obtains a pair of accepting transcripts with (ρ^*, u_m, c) and (s'_ρ, u'_m, c'). The extractor can compute $\Delta_{\rho^*} = \rho^* - s'_\rho$ and $\Delta_{u_m} = u_m - u'_m \mod q$. We denote $\rho = \frac{\Delta_{\rho^*}}{\Delta_c}$ and $m = \frac{\Delta_{u_m}}{\Delta_c} \mod q$. Hence we have $C_1^{\Delta_c} = (\mathsf{pk}^\rho f^m)^{\Delta_c}$. If $C_1 \neq \mathsf{pk}^\rho f^m$, then $\frac{\mathsf{pk}^\rho f^m}{C_1}$ is a non-trivial element of order $\Delta_c < q$ which contradicts Corollary 2.5.

Note that our scheme includes a sub-protocol $\mathsf{ZKPoKRepS}$ on input C_2 w.r.t. bases $g_q \in G \setminus F$. Since $\mathsf{ZKPoKRepS}$ is an argument of knowledge, there exists an extractor to extract the same ρ such that $C_2 = g_q^\rho$.

Hence the extractor can output (m, ρ) such that $C_1 = \mathsf{pk}^\rho f^m$, $C_2 = g_q^\rho$.

□

Theorem 4.4. *The protocol* $\mathsf{ZKPoKEnc}$ *is an honest-verifier statistically zero-knowledge argument of knowledge for relation* $\mathcal{R}_{\mathsf{Enc}}$ *in the generic group model.*

Proof. The simulator Sim randomly picks a challenge $c' \in [0, q-1]$ and a prime $\ell' \in \mathsf{Prime}(\lambda)$. It picks randomly $u'_m \in \mathbb{Z}_q$, $d'_\rho \in [0, B-1]$ and $e'_\rho \in [0, q\ell' - 1]$.

It computes: $D'_1 = \mathsf{pk}^{d'_\rho}, D'_2 = g_q^{d'_\rho}, S'_1 = D_1'^{q\ell'} \mathsf{pk}^{e'_\rho} f^{u'_m} C_1^{-c'}, S'_2 = D_2'^{q\ell'} g_q^{e'_\rho} C_2^{-c'}$.

We argue that the simulated transcript $(S'_1, S'_2, c', u'_m, D'_1, D'_2, e'_\rho, \ell')$ is indistinguishable from a real transcript $(S_1, S_2, c, u_m, D_1, D_2, e_\rho, \ell)$ between a prover and a verifier. Sim chooses (ℓ', c') identically to the honest verifier. Both u_m and u'_m are uniformly distributed in $\mathbb{Z}_q$. (S'_1, S'_2) is uniquely defined by the other values such that the verification holds.

For simulated transcript (D'_1, D'_2, e'_ρ) and real transcript (D_1, D_2, e_ρ), we want to prove that the simulator produces statistically indistinguishable transcripts in that: independent of ℓ and c, the values e_ρ has a negligible statistical distance from the uniform distribution over $[0, q\ell - 1]$ and each one of pk^{d_ρ}, $g_q^{d_\rho}$ has negligible statistical from uniform over $G_k = \langle \mathsf{pk} \rangle$, G^q respectively; D_1, D_2 and e_ρ are independent.

Consider fixed values of c, ρ and ℓ. In the real protocol, the prover computes $u_\rho = c\rho + s_\rho$ where s_ρ is uniform in $[-B, B]$ and sets $e_\rho = u_\rho \bmod q\ell$. By Fact 1, the value of u_ρ is distributed uniformly over a range of $2B + 1$ consecutive integers, thus e_ρ has a statistical distance at most $q\ell/(2B + 1)$ from uniform over $[0, q\ell - 1]$. This bounds the distance between the real e_ρ and the simulated e'_ρ, which is uniform over $[0, q\ell - 1]$.

Next, we show that $g_q^{d_\rho}$ is statistically indistinguishable from uniform in G^q. The distribution of $g_q^{d_\rho}$ over G^q is determined by the distribution of $d_\rho \bmod |G^q|$. Similar to *Theorem 2*, we conclude the that the statistical distance of d_ρ from U_Y is at most: $\frac{1}{2}[Y(\frac{1}{Y} - \frac{q\ell}{2B+1}) + \frac{2(q\ell-1)}{2B+1}] = \frac{5q\ell-4}{2(2B+1)} < \frac{2q\ell}{B} \leq \frac{2^{\lambda+1}q}{B}$ and the statistical distance of $d_\rho \bmod |G^q|$ from $U_Y \bmod |G^q|$ will not exceed. By Fact 1, $U_Y \bmod |G^q|$ has a statistical distance at most $\frac{|G^q|}{|Y|} \leq \frac{2^\lambda q|G^q|}{2B+3-3q\cdot 2^\lambda}$. Thus, by triangle inequality, we have the statistical distance of $d_\rho \bmod |G^q|$ from uniform is at most $\frac{2^{\lambda+1}q}{B} + \frac{2^\lambda q|G^q|}{2B+3-3q\cdot 2^\lambda}$. This also bounds the distance of

$g_q^{d_\rho}$ from uniform in G^q. The simulated value d'_ρ is uniformly chosen from a set of size B. Again by Fact 1, if $|G^q| < B$, then $d'_\rho \bmod |G^q|$ has a distance $|G^q|/B$ from uniform. The simulated value $g_q^{d'_\rho}$ has a distance at most $|G^q|/B$ from uniform in G^q. By the triangle inequality again, the statistical distance of $g_q^{d_\rho}$ and $g_q^{d'_\rho}$ is at most $\frac{2^{\lambda+1}q}{B} + \frac{2^\lambda q|G^q|}{2B+3-3q\cdot 2^\lambda} + \frac{|G^q|}{B} < \frac{(2^\lambda q+2)|G^q|+2^{\lambda+2}q}{2B+3-3q\cdot 2^\lambda} \leq \frac{1}{2^{\epsilon_d+2}}$ if $B \geq 2^{\epsilon_d+1}(2^\lambda q+2)|G^q| + 2^{\epsilon_d+\lambda+3}q + 3q\cdot 2^{\lambda-1} - \frac{3}{2}$ for some distance parameter ϵ_d. Similarly, the same argument holds for the distances of pk^{d_ρ} and $\mathsf{pk}^{d'_\rho}$. By using Fact 3, the distance between the joint distribution $X_\rho = (\mathsf{pk}^{d\rho}, g_q^{d_\rho})$ and the simulated distribution $Y_\rho = (\mathsf{pk}^{d'_\rho}, g_q^{d'_\rho})$ is at most $\frac{1}{2^{\epsilon_d+1}}$.

Finally, we consider the joint distribution of $(\mathsf{pk}^{d\rho}, g_q^{d_\rho})$ and e_ρ. Consider the conditional distribution of $d_\rho|e_\rho$. Note that $r_\rho = z$ if $(s_\rho - e_\rho)/q\ell = z$. We repeat a similar argument as above for bounding the distribution of d_ρ from uniform. For each possible value of z, there always exists a unique value of s_ρ such that $\lfloor \frac{s_\rho}{q\ell} \rfloor = z$ and $s_\rho = 0 \bmod q\ell$, expect possibly at the two endpoints E_1 and E_2 of the range of d_ρ. When e_ρ disqualifies the two points E_1 and E_2, then each of the remaining points $z \notin \{E_1, E_2\}$ still have equal probability mass, and thus the probability $Pr(d_\rho = z|e_\rho)$ increases by at most $\frac{1}{|Y|} - \frac{q\ell}{2B+1}$. The same applies to the variable $(\mathsf{pk}^{d\rho}, g_q^{d_\rho})|e_\rho$.

We can compare the joint distribution $X_\rho = (\mathsf{pk}^{d_\rho}, g_q^{d_\rho}, e_\rho)$ to the simulated distribution $Y_\rho = (\mathsf{pk}^{d'_\rho}, g_q^{d'_\rho}, e'_\rho)$ using Fact 3. Setting $\epsilon_1 = \frac{1}{|Y|} - \frac{q\ell}{2B+1}$ and $\epsilon_2 = 0$, the distance between these joint distributions is at most $\frac{1}{2^{\epsilon_d+1}} + \frac{q\ell}{2B+1} + \epsilon_1 q\ell = \frac{1}{2^{\epsilon_d+1}} + \frac{q^2\ell^2}{2B+3-3q\ell} + \frac{q\ell(1-q\ell)}{2B+1} < \frac{1}{2^{\epsilon_d+1}} + \frac{q\ell}{2B+3-3q\ell} < \frac{1}{2^{\epsilon_d}}$, where the last equality holds if $B > 2^{\epsilon_d+\lambda}q + 3q\cdot 2^{\lambda-1} - \frac{3}{2}$. This bounds the distance between (D_1, D_2, e_ρ) and (D'_1, D'_2, e'_ρ).

Combining the two requirements on B (Recall the first requirement is $B \geq 2^{\epsilon_d+1}(2^\lambda q+2)|G^q| + 2^{\epsilon_d+\lambda+3}q + 3q\cdot 2^{\lambda-1} - \frac{3}{2}$ for some distance parameter ϵ_d), we can simplify the requirement as $B \geq 2^{\epsilon_d+\lambda+2}q\tilde{s}$.

□

4.3.3 ZK Proof for Affine Transformation for CL Ciphertext

We want to prove a relation between two CL ciphertext c and $\tilde{c}$. Here c is the encryption of k using randomness r, but (k, r) are not treated as witness of the ZK proof. The ciphertext c is transformed to a ciphertext $\tilde{c}$ for a message $k\gamma + \beta$ by using the additive homomorphic property of the CL encryption.

Algorithm 13: Protocol $\mathsf{ZKPoKAff}$ for the relation $\mathcal{R}_{\mathsf{Aff}}$

Param: $\mathcal{G}_{\mathrm{HSM}} \leftarrow \mathsf{GGen}_{\mathrm{HSM},q}(1^\lambda)$; $B = 2^{\epsilon_d+\lambda+3}q^2\tilde{s}$ where $\epsilon_d = 80$.
Input: $C_1, \tilde{C}_1 \in G \setminus F; C_2, \tilde{C}_2, \mathsf{pk} \in G^q$.
Witness: $\rho \in [0, S], \gamma, \beta \in \mathbb{Z}_q$, where $S = \tilde{s} \cdot 2^{\epsilon_d}$

1 Prover chooses $s_\rho, s_\gamma \xleftarrow{\$} [-B, B]$, $s_\beta \xleftarrow{\$} \mathbb{Z}_q$ and computes:

$$S_1 = C_1^{s_\gamma} f^{s_\beta} \mathsf{pk}^{s_\rho}, \quad S_2 = C_2^{s_\gamma} g_q^{s_\rho}.$$

Prover sends (S_1, S_2) to the verifier.

2 Verifier sends $c \xleftarrow{\$} [0, q-1]$ to the prover. Prover aborts if $c \notin [0, q-1]$.

3 Verifier sends $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$ to the prover.

4 Prover computes:

$$u_\beta = s_\beta + c\beta \mod q, \quad u_\rho = s_\rho + c\rho, \quad u_\gamma = s_\gamma + c\gamma.$$

Prover finds $d_\rho, d_\gamma \in \mathbb{Z}$ and $e_\rho, e_\gamma \in [0, q\ell - 1]$ s.t. $u_\rho = d_\rho q\ell + e_\rho$ and $u_\gamma = d_\gamma q\ell + e_\gamma$. Prover computes:

$$D_1 = \mathsf{pk}^{d_\rho}, \quad D_2 = g_q^{d_\rho}, \quad E_1 = C_1^{d_\gamma}, \quad E_2 = C_2^{d_\gamma}.$$

Prover sends $(u_\beta, D_1, D_2, E_1, E_2, e_\rho, e_\gamma)$ to the verifier.

5 Verifier accepts if $e_\rho \in [0, q\ell - 1]$ and:

$$(D_1E_1)^{q\ell}\mathsf{pk}^{e_\rho}C_1^{e_\gamma} f^{u_\beta} = S_1\tilde{C}_1^c, \quad (D_2E_2)^{q\ell} g_q^{e_\rho} C_2^{e_\gamma} = S_2\tilde{C}_2^c.$$

Define ρ as the randomness of encrypting β.

$$\begin{aligned}
c &= \mathsf{Encrypt}_{\mathsf{pk}}(k; r) = (C_1 = f^k\mathsf{pk}^r, C_2 = g_q^r) \\
\tilde{c} &= \mathsf{Encrypt}_{\mathsf{pk}}(k\gamma + \beta) \\
&= \mathsf{EvalAdd}(\mathsf{EvalScal}(c, \gamma), \mathsf{Encrypt}_{\mathsf{pk}}(\beta; \rho)) \\
&= \mathsf{EvalAdd}((C_1^\gamma, C_2^\gamma), (f^\beta\mathsf{pk}^\rho, g_q^\rho)) \\
&= (\tilde{C}_1 = C_1^\gamma f^\beta \mathsf{pk}^\rho, \tilde{C}_2 = C_2^\gamma g_q^\rho)
\end{aligned}$$

So we obtain the following relation:

$$\begin{aligned}
\mathcal{R}_{\mathsf{Aff}} = \{&(\mathsf{pk}, C_1, C_2, \tilde{C}_1, \tilde{C}_2); (\gamma, \beta, \rho) | \mathsf{pk}, C_2 \in G^q; \\
&C_1 \in G \setminus F; \gamma, \beta \in \mathbb{Z}_q; \rho \in [0, S] : \\
&\tilde{C}_1 = C_1^\gamma f^\beta \mathsf{pk}^\rho \wedge \tilde{C}_2 = C_2^\gamma g_q^\rho\}.
\end{aligned}$$

Theorem 4.5. *The protocol* ZKPoKAff *is an argument of knowledge for* $\mathcal{R}_{\mathsf{Aff}}$ *in the generic group model.*

Proof. We rewind the adversary on fresh challenges ℓ so that each accepting transcript outputs an (e_ρ, e_γ, ℓ), where $\rho^* = e_\rho \bmod q\ell$ and $\gamma^* = e_\gamma \bmod q\ell$ with overwhelming probability. We consider the following two cases:

Case 1. If $\mathsf{pk}^{\rho^*} C_1^{\gamma^*} \neq S_1 \tilde{C}_1^c f^{-u_\beta}$ and $(\mathsf{pk}^{\rho^*} C_1^{\gamma^*})^q \neq (S_1 \tilde{C}_1^c f^{-u_\beta})^q$, then we have $\mathsf{pk}^{\rho^*} C_1^{\gamma^*} \neq S_1 \tilde{C}_1^c f^{-u_\beta} = D_1^{q\ell} E_1^{q\ell} \mathsf{pk}^{e_\rho} C_1^{e_\gamma}$. Let $\chi_\rho = \frac{e_\rho - \rho^*}{\ell}$ and $\chi_\gamma = \frac{e_\gamma - \gamma^*}{\ell}$. Then $D_1^q E_1^q \mathsf{pk}^{\chi_\rho} C_1^{\chi_\gamma}$ is an ℓ-th root of $(S_1 \tilde{C}_1^c f^{-u_\beta})/(\mathsf{pk}^{\rho^*} C_1^{\gamma^*}) \neq 1$. This breaks the adaptive root subgroup assumption since $(S_1 \tilde{C}_1^c f^{-u_\beta})^q / (\mathsf{pk}^{\rho^*} C_1^{\gamma^*})^q \neq 1$.

Case 2. If $\mathsf{pk}^{\rho^*} C_1^{\gamma^*} \neq S_1 \tilde{C}_1^c f^{-u_\beta}$ and $(\mathsf{pk}^{\rho^*} C_1^{\gamma^*})^q = (S_1 \tilde{C}_1^c f^{-u_\beta})^q$, then $S_1 \tilde{C}_1^c = \mathsf{pk}^{\rho^*} C_1^{\gamma^*} f^{\delta'}$ for some $\delta' \neq u_\beta \in \mathbb{Z}_q$. Then we have $S_1 \tilde{C}_1^c f^{-u_\beta} = \mathsf{pk}^{\rho^*} C_1^{\gamma^*} f^{\delta' - u_\beta} = D_1^{q\ell} E_1^{q\ell} \mathsf{pk}^{e_\rho} C_1^{e_\gamma}$. By Lemma 2.2, write $C_1 = f^v \prod_{i=1}^{m_0} g_i^{\alpha_i}$. Consider the f parts in both sides, we have $f^{v \cdot \gamma^* + \delta' - u_\beta} = f^{v \cdot e_\gamma}$, namely $v \cdot (\gamma^* - e_\gamma) = u_\beta - \delta' \bmod q$. Since $0 < |u_\beta - \delta'| < q$ and $\gamma^* - e_\gamma = 0 \bmod q$, contradiction happens. Hence, by Corollary 2.4, $\mathsf{pk}^{\rho^*} C_1^{\gamma^*} f^{u_\beta} = S_1 \tilde{C}_1^c$ with overwhelming probability.

By rewinding, the extractor obtains a pair of accepting transcripts with $(\rho^*, \gamma^*, u_\beta, c)$ and $(s'_\rho, s'_\gamma, u'_\beta, c')$. The extractor can compute $\Delta_{\rho^*} = \rho^* - s'_\rho$, $\Delta_{\gamma^*} = \gamma^* - s'_\gamma$ and $\Delta_{u_\beta} = u_\beta - u'_\beta \bmod q$. We denote $\rho = \frac{\Delta_{\rho^*}}{\Delta_c}, \gamma = \frac{\Delta_{\gamma^*}}{\Delta c}$ and $\beta = \frac{\Delta_{u_\beta}}{\Delta_c} \bmod q$. Hence we have $\tilde{C}_1^{\Delta_c} = (\mathsf{pk}^\rho C_1^\gamma f^\beta)^{\Delta_c}$. If $\tilde{C}_1 \neq \mathsf{pk}^\rho C_1^\gamma f^\beta$, then $\frac{\mathsf{pk}^\rho f^\beta C_1^\gamma}{\tilde{C}_1}$ is a non-trivial element of order $\Delta_c < q$ which contradicts Corollary 2.5.

Note that our scheme includes a sub-protocol ZKPoKRepS on input $\tilde{C}_2$ w.r.t. bases $g_q \in G \setminus F$. Since ZKPoKRepS is an argument of knowledge, there exists an extractor to extract the same (γ, ρ) such that $\tilde{C}_2 = C_2^\gamma g_q^\rho$.

Hence the extractor can output (β, γ, ρ) such that $\tilde{C}_1 = C_1^\gamma f^\beta \mathsf{pk}^\rho$ and $\tilde{C}_2 = C_2^\gamma g_q^\rho$.

□

Theorem 4.6. *The protocol* ZKPoKAff *is an honest-verifier statistically zero-knowledge argument of knowledge for relation* $\mathcal{R}_{\mathsf{Aff}}$ *in the generic group model.*

Proof. The simulator Sim randomly picks a challenge $c' \in [0, q-1]$ and a prime $\ell' \in \mathsf{Prime}(\lambda)$. It picks randomly $u'_\beta \in \mathbb{Z}_q$, $d'_\rho, d'_\gamma \in [0, B-1]$ and $e'_\rho, e'_\gamma \in [0, q\ell' - 1]$.

It computes:

$$D'_1 = \mathsf{pk}^{d'_\rho}, \quad D'_2 = g_q^{d'_\rho}, \quad E'_1 = C_2^{d'_\gamma}, \quad E'_2 = C_2^{d'_\gamma}$$

$$S'_1 = {D'_1}^{q\ell'} \mathsf{pk}^{e'_\rho} f^{u'_\beta} C_1^{e'_\gamma} \tilde{C}_1^{-c'}, \quad S'_2 = {D'_2}^{q\ell'} g_q^{e'_\rho} C_2^{e'_\gamma} \tilde{C}_2^{-c'}$$

We argue that the simulated transcript $(S'_1, S'_2, c', u'_\beta, D'_1, D'_2, e'_\rho, e'_\gamma, \ell')$ is indistinguishable from a real transcript $(S_1, S_2, c, u_\beta, D_2, e_\rho, e_\gamma D_1, \ell)$ between a prover and a verifier. Sim chooses (ℓ', c') identically to the honest verifier. Both u_β and u'_β are uniformly distributed in $\mathbb{Z}_q$. (S'_1, S'_2) is uniquely defined by the other values such that the verification holds. Next, we compare the simulated transcript $(D'_1, D'_2, E'_1, E'_2, e'_\rho, e'_\gamma)$ and the real transcript $(D_1, D_2, E_1, E_2, e_\rho, e_\gamma)$.

Similar to the proof of Theorem 4, we know that e_ρ or e_γ has a statistical distance at most $q\ell/(2B+1)$ from uniform over $[0, q\ell - 1]$. This bounds the distance between the real e_ρ (resp. e_γ) and the simulated e'_ρ (resp. e'_γ), which is uniform over $[0, q\ell - 1]$.

By similar argument in Theorem 2, the statistical distances of $(g_q^{d_\rho}, g_q^{d'_\rho})$,$(\mathsf{pk}^{d_\rho}, \mathsf{pk}^{d'_\rho})$ and $(C_2^{d_\gamma}, C_2^{d'_\gamma})$ are all at most $\mathsf{dist}_1 = \frac{2^{\lambda+1}q}{B} + \frac{2^\lambda q|G^q|}{2B+3-3q\cdot 2^\lambda} + \frac{|G^q|}{B}$; the statistical distance of $(C_1^{d_\gamma}, C_1^{d'_\gamma})$ is at most $\mathsf{dist}_2 = \frac{2^{\lambda+1}q}{B} + \frac{2^\lambda q|G|}{2B+3-3q\cdot 2^\lambda} + \frac{|G|}{B}$. We have $\mathsf{dist}_1 \leq \frac{1}{2^{\epsilon_d+3}}$ if $B \geq 2^{\epsilon_d+2}(2^\lambda q + 2)|G^q| + 2^{\epsilon_d+\lambda+4}q + 3q\cdot 2^{\lambda-1} - \frac{3}{2}$ for some distance parameter ϵ_d; and $\mathsf{dist}_2 \leq \frac{1}{2^{\epsilon_d+3}}$ if $B \geq 2^{\epsilon_d+2}(2^\lambda q + 2)|G| + 2^{\epsilon_d+\lambda+4}q + 3q\cdot 2^{\lambda-1} - \frac{3}{2}$ for some distance parameter ϵ_d; By Fact 3, the distance between the joint distribution $X_\rho = (\mathsf{pk}^{d_\rho}, g_q^{d_\rho})$ and the simulated distribution $Y_\rho = (\mathsf{pk}^{d'_\rho}, g_q^{d'_\rho})$ is at most $\frac{1}{2^{\epsilon_d+2}}$; the distance between the joint distribution $X_\gamma = (C_1^{d_\gamma}, C_2^{d_\gamma})$ and the simulated distribution $Y_\gamma = (C_1^{d'_\gamma}, C_2^{d'_\gamma})$ is at most $\frac{1}{2^{\epsilon_d+2}}$. Moreover, we can obtain that the probability $Pr(d_\rho = z|e_\rho)$ or $Pr(d_\gamma = z|e_\gamma)$ increases by at most $\frac{1}{|Y|} - \frac{q\ell}{2B+1}$. The same applies to the variable $(\mathsf{pk}^{d_\rho}, g_q^{d_\rho})|e_\rho$ (resp. $(C_1^{d_\gamma}, C_2^{d_\gamma})|e_\gamma$).

We can compare the joint distributions $X'_\rho = (\mathsf{pk}^{d_\rho}, g_q^{d_\rho}, e_\rho)$ to the simulated distribution $Y'_\rho = (\mathsf{pk}^{d'_\rho}, g_q^{d'_\rho}, e'_\rho)$ using Fact 3. Setting $\epsilon_1 = \frac{1}{|Y|} - \frac{q\ell}{2B+1}$ and $\epsilon_2 = 0$, the distance between these joint distributions is at most $\frac{1}{2^{\epsilon_d+2}} + \frac{q\ell}{2B+1} + \epsilon_1 q\ell = \frac{1}{2^{\epsilon_d+2}} + \frac{q^2\ell^2}{2B+3-3q\ell} + \frac{q\ell(1-q\ell)}{2B+1} < \frac{1}{2^{\epsilon_d+2}} + \frac{q\ell}{2B-3q\ell+3} \leq \frac{1}{2^{\epsilon_d+1}}$, where the last equality holds if $B \geq 2^{\epsilon_d+\lambda+1}q + 3q\cdot 2^{\lambda-1} - \frac{3}{2}$. Similarly, we have the distance between the joint distributions $X'_\gamma = (C_1^{d_\gamma}, C_2^{d_\gamma}, e_\gamma)$ and $Y'_\gamma = (C_1^{d'_\gamma}, C_2^{d'_\gamma}, e'_\gamma)$ at most $\frac{1}{2^{\epsilon_d+1}}$ if $B \geq 2^{\epsilon_d+\lambda+1}q + 3q\cdot 2^{\lambda-1} - \frac{3}{2}$. Using Fact 2, we have the statistical distance between $X = (\mathsf{pk}^{d_\rho}, g_q^{d_\rho}, C_1^{d_\gamma}, C_2^{d_\gamma}, e_\gamma, e_\gamma)$ and $Y = (\mathsf{pk}^{d'_\rho}, g_q{}^{d'_\rho}, C_1^{d'_\gamma}, C_2^{d'_\gamma}, e'_\rho, e'_\gamma)$ at most $\frac{1}{2^{\epsilon_d}}$. Combining the above requirements on B, we have a simplified requirement that $B \geq 2^{\epsilon_d+\lambda+3}q^2\tilde{s}$. □

Remarks. The analyses for argument of knowledge for $\mathsf{ZKPoKEnc}$ and $\mathsf{ZKPoKAff}$ are similar but they achieve contradiction in Case 2 in different ways. The

TABLE 4.1: Comparison on proof size in bits where $\lambda = 128, ||q|| = 256, ||\Delta|| = 2339$.

Protocol	[78]	This work	Change
ZKPoKRepS (n=1)	7401	5062	⇑ 31.6%
ZKPoKEnc	14674	9996	⇑ 31.9%
ZKPoKAff	×	15058	-

former utilizes the fact that in its first verification equation of the last step $D_1^{q\ell}\mathsf{pk}^{e_\rho}$ cancels f with the assumption $\mathsf{pk} \in G^q$. The latter cannot follow the same method since there is one C_1 lying in $G \backslash F$ involved, in which case we lead to contraction through analyzing the exponent of f parts in the verification equation.

4.3.4 Comparison with ZK Proofs in PKC-2021

Yuen *et al.* [78] proposed two compact ZK proofs for ZKPoKRepS and ZKPoKEnc. In the previous section, we further improve these two protocols, which can be directly plugged into the two threshold schemes proposed in [78]. We also give a ZK Proof for ZKPoKAff. We now show the improvement in terms of bandwidth and running time.

Parameter setting. For computing class group size, according to [47], each reduced class group represented by (a, b, Δ) satisfies that $-a < b \leq a$ and $a < \sqrt{|\Delta|/3}$. Let $||\Delta||$ denote the bit length of Δ. Then, a and b can be denoted by a $\lceil \frac{||\Delta||-1}{2} \rceil$-bit string and a $\lceil \frac{||\Delta||-1}{2} \rceil + 1$-bit string ($b$ needs one more bit to represent its sign). Since Δ is already store in the common public key which is available for every party, we directly use $2 \times \lceil \frac{||\Delta||-1}{2} \rceil + 1$ to represent the bit size of one class group element (also equivalent to $\log_2 |\Delta|$ bits). According to [23], we require $||\Delta|| = ||q^2 \Delta_K|| = 512 + 1827 = 2339$ in 128-bit security level.

Result. We implement our ZK proofs in Rust language using a MacBook Pro laptop with 16G RAM and Intel Core i5. We use the Class[4] library in Rust. The running time and bandwidth are shown in Table 5.6 and 4.2. The proof generation is much faster than the counterparts in [78] for ZKPoKRepS and ZKPoKEnc. The verification time does not vary much. The ZKPoKAff is relatively costly in proving but efficient in verification.

[4]https://github.com/ZenGo-X/class

TABLE 4.2: Comparison on the running time for both prover and verifier (128-bit security level).

	Prover			Verifier		
	[78]	Ours	change	[78]	Ours	change
ZKPoKRepS (n=1)	488.7 ms	412.5 ms	⇑ 15.6%	168.6 ms	164.2 ms	⇑ 2.6%
ZKPoKEnc	969.2 ms	684.5 ms	⇑ 29.4%	333.2 ms	324.6 ms	⇑ 2.6%
ZKPoKAff	×	1715.2 ms	-	×	441.9 ms	-

4.4 Application to UC Non-interactive, Proactive Threshold ECDSA

Canetti *et al.* [21] proposed a UC non-interactive, proactive threshold ECDSA which introducing a global random oracle and an enhanced ECDSA assumption. Their implementation involves zero-knowledge proofs in order to validate the information among the interaction. In this section, we broaden the application of ZK proofs as used in [21], and propose enhancements using our newly developed ZK proofs. Additionally, we demonstrate that our modifications maintain the UC property, and we present a performance analysis comparing our implementation and the one provided in [21].

4.4.1 ZK Proofs in Threshold ECDSA

We generalize the ZK proofs from [21] in a high level as follows.

$$
\begin{aligned}
\mathcal{R}_{\mathsf{enc}} =& \{(\mathsf{pk} \in G^q, K \in \mathbb{C}); (k, \rho \in \mathbb{Z}) : K = \mathsf{Enc}_{pk}(k; \rho)\} \\
\mathcal{R}_{\mathsf{log}} =& \{(\mathsf{pk} \in G^q, \hat{P}, \hat{X} \in \hat{G}, C \in \mathbb{C}); (x, \rho \in \mathbb{Z}) : \\
& \hat{X} = \hat{P}^x, C = \mathsf{Enc}_{pk}(x; \rho)\} \\
\mathcal{R}_{\mathsf{aff\text{-}g}} =& \{(Y, C, D \in \mathbb{C}, \mathsf{pk}_1, \mathsf{pk}_2 \in G^q, \hat{X} \in \hat{G}); \\
& (x, y, \rho_y, \rho \in \mathbb{Z}) : \hat{X} = \hat{P}^x, Y = \mathsf{Enc}_{\mathsf{pk}_1}(y; \rho_y), \\
& D = \mathsf{EvalAdd}(\mathsf{EvalScal}(C, x), \mathsf{Enc}_{\mathsf{pk}_2}(y; \rho)\} \\
\mathcal{R}_{\mathsf{aff\text{-}p}} =& \{(X, Y, C, D \in \mathbb{C}, \mathsf{pk}_1, \mathsf{pk}_2 \in G^q); \\
& (x, y, \rho_x, \rho_y, \rho \in \mathbb{Z}) : X = \mathsf{Enc}_{\mathsf{pk}_1}(x; \rho_x), \\
& Y = \mathsf{Enc}_{\mathsf{pk}_1}(y; \rho_y), D = \mathsf{EvalAdd}(\mathsf{EvalScal}(C, x), \\
& \mathsf{Enc}_{\mathsf{pk}_2}(y; \rho)\}
\end{aligned}
$$

We've transformed them into CL implementations and have labeled them as ZKPoKEnc (consistent with Algorithm 2), ZKPoKLog, ZKPoKAff-g, and

TABLE 4.3: Sample ranges in different settings.

ZKP	B
ZKPoKRepS	$2^{\epsilon_d+2\lambda}nq^2\tilde{s}$
ZKPoKEnc	$2^{\epsilon_d+\lambda+2}q\tilde{s}$
ZKPoKAff	$2^{\epsilon_d+\lambda+3}q^2\tilde{s}$
ZKPoKAff-p	$2^{\epsilon_d+\lambda+4}q(5+q\tilde{s})$
ZKPoKAff-g	$2^{\epsilon_d+\lambda+2}q(5+q\tilde{s})$
ZKPoKLog	$2^{\epsilon_d+\lambda+2}q\tilde{s}$

ZKPoKAff-p, as illustrated in algorithms 14, 15, 16. These will be utilized to construct a CL-based bandwidth-efficient threshold ECDSA scheme (adaptation of Canetti's UC non-interactive proactive threshold ECDSA in a class group setting), achieving a bandwidth-optimal version without sacrificing its UC security, non-interactivity, and proactive security. We want to exphasize that the sampling bounds B for ZKPoKAff-p and ZKPoKAff-g differ from those of other ZK proofs. ZKPoKLog shares the same B as ZKPoKEnc. We've summarized the sampling ranges for each ZK in Table 4.3.

4.4.2 Construct CL-based Bandwidth-efficient Threshold ECDSA

We present our CL-based bandwidth-efficient threshold ECDSA, including key generation, key refreshment, pre-signing and online signing respectively in Table 4.4, 4.5, (4.6 & 4.7) and 4.8, where communication through point-to-point channel and synchronized broadcast channel are denoted by $\rightarrow$ and $\Rightarrow$ respectively. Note that for Table 4.4, it should include both ECDSA key generation (Figure 5 in [20]) and running auxiliary info according to the description in Figure 4 in [20]. We summarize the major changes from Canetti's Paillier based threshold ECDSA to our CL-based threshold ECDSA as follows.

- For KeyGen and KeyRefresh, we replace the two proofs for $\mathcal{R}_{\mathsf{mod}}$ (denoted by ZKPoKMod) and one proof $\mathcal{R}_{\mathsf{prm}}$ (denoted by ZKPoKPrm), with one single ZKPoKRepS ($n = 1$) for each party.
- For Paillier-related algorithms including encryption key pair generation, $\mathsf{Enc}, \mathsf{Dec}, \mathsf{EvalAdd}$ and $\mathsf{EvalScal}$ are updated with CL encryption alternatives; all ZK proofs are transformed from Paillier version to corresponding CL version, as the blue parts shown in Table 4.4, 4.5 and (4.6 & 4.7).

Algorithm 14: Protocol $\mathsf{ZKPoKAff\text{-}p}$ for the relation $\mathcal{R}_{\mathsf{aff-p}}$

Param: $\mathcal{G}_{\mathrm{HSM}} \leftarrow \mathsf{GGen}_{\mathrm{HSM},q}(1^\lambda)$; $B = 2^{\epsilon_d+\lambda+4}q(5+q\tilde{s})$ where $\epsilon_d = 80$.
Input: $X_1, X_2, Y_1, Y_2, C_1, C_2, D_1 D_2, \mathsf{pk}_1, \mathsf{pk}_2 \in G^q$.
Witness: $\rho, \rho_x, \rho_y \in [0, S], x, y \in \mathbb{Z}_q$, where $S = \tilde{s} \cdot 2^{\epsilon_d}$.

1 Prover chooses $s_\rho, s_x, s'_x, s'_y \xleftarrow{\$} [-B, B]$, $s_y \xleftarrow{\$} \mathbb{Z}_q$ and computes: $S_1 = f^{s_x}\mathsf{pk}_1^{s'_x}, S_2 = g_q^{s'_x}, S_3 = f^{s_y}\mathsf{pk}_1^{s'_y}, S_4 = g_q^{s'_y}, S_5 = C_1^{s_x} f^{s_y}\mathsf{pk}_2^{s_\rho}, S_6 = C_2^{s_x} g_q^{s_\rho}$
Prover sends $(S_1, S_2, S_3, S_4, S_5, S_6)$ to the verifier.

2 Verifier sends $c \xleftarrow{\$} [0, q-1]$ to the prover. Prover aborts if $c \notin [0, q-1]$.

3 Verifier sends $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$ to the prover.

4 Prover computes: $u_y = s_y + cy$
mod $q, u_x = s_x + cx, u_\rho = s_\rho + c\rho, u'_x = s'_x + c\rho_x, u'_y = s'_y + c\rho_y$. Prover finds $d_x, d_\rho, d'_x, d'_y \in \mathbb{Z}$ and $e_x, e_\rho, e'_x, e'_y \in [0, q\ell - 1]$ s.t.
$u_x = d_x q\ell + e_x, u_\rho = d_\rho q\ell + e_\rho, u'_x = d'_x q\ell + e'_x, u'_y = d'_y q\ell + e'_y$, Prover computes: $E_1 = \mathsf{pk}_1^{d'_x}, E_2 = g_q^{d'_x}, E_3 = \mathsf{pk}_1^{d'_y}, E_4 = g_q^{d'_y}$,
$E_5 = \mathsf{pk}_2^{d_\rho}, E_6 = g_q^{d_\rho}, H_1 = C_1^{d_x}, H_2 = C_2^{d_x}, F = f^{d_x}$. Prover sends $(u_y, E_1, E_2, E_3, E_4, E_5, E_6, H_1, H_2, F, e_\rho, e_x, e'_x, e'_y)$ to the verifier.

5 Verifier accepts if $e_\rho \in [0, q\ell - 1]$ and: $(E_1 F)^{q\ell}\mathsf{pk}_1^{e'_x} f^{e_x} = S_1 X_1^c, \quad E_2^{q\ell} g_q^{e'_x} = S_2 X_2^c, E_3^{q\ell}\mathsf{pk}_1^{e'_y} f^{u_y} = S_3 Y_1^c, \quad E_4^{q\ell} g_q^{e'_y} = S_4 Y_2^c, (E_5 H_1)^{q\ell}\mathsf{pk}_2^{e_\rho} C_1^{e_x} f^{u_y} = S_5 D_1^c, (E_6 H_2)^{q\ell} g_q^{e_\rho} C_2^{e_x} = S_6 D_2^c$.

Algorithm 15: Protocol $\mathsf{ZKPoKAff\text{-}g}$ for the relation $\mathcal{R}_{\mathsf{aff-g}}$

Param: $\mathcal{G}_{\mathrm{HSM}} \leftarrow \mathsf{GGen}_{\mathrm{HSM},q}(1^\lambda)$; $B = 2^{\epsilon_d+\lambda+2}q(5+q\tilde{s})$ where $\epsilon_d = 80$.
Input: $Y_1, Y_2, C_1, C_2, D_1, D_2, \mathsf{pk}_1, \mathsf{pk}_2 \in G^q, \hat{X} \in \hat{G}$.
Witness: $\rho, \rho_x, \rho_y \in [0, S], x, y \in \mathbb{Z}_q$, where $S = \tilde{s} \cdot 2^{\epsilon_d}$

1 Prover chooses $s_\rho, s_x, s'_y \xleftarrow{\$} [-B, B]$, $s_y \xleftarrow{\$} \mathbb{Z}_q$ and computes:
$S_1 = f^{s_y}\mathsf{pk}_1^{s'_y}, S_2 = g_q^{s'_y}, R_1 = C_1^{s_x} f^{s_y}\mathsf{pk}_2^{s_\rho}, R_2 = C_2^{s_x} g_q^{s_\rho}, \hat{S} = \hat{P}^{s_x}$ Prover sends $(S_1, S_2, R_1, R_2, \hat{S})$ to the verifier.

2 Verifier sends $c \xleftarrow{\$} [0, q-1]$ to the prover. Prover aborts if $c \notin [0, q-1]$.

3 Verifier sends $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$ to the prover.

4 Prover computes: $u_y = s_y + cy$
mod $q, u_x = s_x + cx, u_\rho = s_\rho + c\rho, u'_y = s'_y + c\rho_y$. Prover finds $d_\rho, d_x, d'_y \in \mathbb{Z}$ and $e_x, e_\rho, e'_y \in [0, q\ell - 1]$ s.t.
$u_x = d_x q\ell + e_x, u_\rho = d_\rho q\ell + e_\rho, u'_y = d'_y q\ell + e'_y$, Prover computes:
$H_1 = C_1^{d_x}, H_2 = C_2^{d_x}, E_1 = \mathsf{pk}_1^{d'_y}, E_2 = g_q^{d'_y}, F_1 = \mathsf{pk}_2^{d_\rho}, F_2 = g_q^{d_\rho}, \hat{D} = \hat{P}^{d_x}$.
Prover sends $(u_y, H_1, H_2, E_1, E_2, F_1, F_2, \hat{D}, e_\rho, e_x, e'_y)$ to the verifier.

5 Verifier accepts if $e_\rho \in [0, q\ell - 1]$ and:
$\hat{D}^{q\ell}\hat{P}^{e_x} = \hat{S}\hat{X}^c, \quad E_2^{q\ell} g_q^{e'_y} = S_2 Y_2^c, E_1^{q\ell}\mathsf{pk}_1^{e'_y} f^{u_y} = S_1 Y_1^c, (H_1 F_1)^{q\ell}\mathsf{pk}_2^{e_\rho} C_1^{e_x} f^{u_y} = R_1 D_1^c, (H_2 F_2)^{q\ell} g_q^{e_\rho} C_2^{e_x} = R_2 D_2^c$.

Algorithm 16: Protocol $\mathsf{ZKPoKLog}$ for the relation $\mathcal{R}_{\mathsf{log}}$

Param: $\mathcal{G}_{\mathrm{HSM}} \leftarrow \mathsf{GGen}_{\mathrm{HSM},q}(1^\lambda)$, $B = 2^{\epsilon_d+\lambda+2}q\tilde{s}$, where $\epsilon_d = 80$.
Input: $\hat{P}, \hat{X}, C_1, C_2, \mathsf{pk} \in G^q$.
Witness: $\rho \in [0, S], m \in \mathbb{Z}_q$, where $S = \tilde{s} \cdot 2^{\epsilon_d}$.

1 Prover chooses $s_\rho \xleftarrow{\$} [-B, B]$, $s_m \xleftarrow{\$} \mathbb{Z}_q$ and computes: $S_1 = \mathsf{pk}^{s_\rho} f^{s_m}, S_2 = g_q^{s_\rho}, \hat{S} = \hat{P}^{s_m}$. Prover sends $(S_1, S_2, \hat{S})$ to the verifier. Prover aborts if $c \notin [0, q-1]$.

2 Verifier sends $c \xleftarrow{\$} [0, q-1]$ to the prover.

3 Verifier sends $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$ to the prover.

4 Prover computes: $u_\rho = s_\rho + c\rho, u_m = s_m + cm \mod q$. Prover finds $d_\rho \in \mathbb{Z}$ and $e_\rho \in [0, q\ell - 1]$ s.t. $u_\rho = d_\rho q\ell + e_\rho$. Prover computes: $D_1 = \mathsf{pk}^{d_\rho}, D_2 = g_q^{d_\rho}$. Prover sends (u_m, D_1, D_2, e_ρ) to the verifier.

5 Verifier accepts if $e_\rho \in [0, q\ell - 1]$ and: $\hat{P}^{u_m} = \hat{S}\hat{X}^c, \quad D_1^{q\ell}\mathsf{pk}^{e_\rho} f^{u_m} = S_1 C_1^c, D_2^{q\ell} g_q^{e_\rho} = S_2 C_2^c$.

- Further, we've optimized the complexity involved in generating and verifying ZK proofs. Please note that in the Key Refresh phase, the operations in the orange box are required $(n-1)^2$ times, and in the Pre-signing phase, they're needed $(n-1)$ times for each party. In our CL setting, however, they are required only $(n-1)$ times and once, respectively.

4.4.3 Security Claim

Recap security proof in [21]. We recall first the security analysis in the Paillier-version non-interactive threshold ECDSA from [21] following the universal composable (UC) framework [19]. In traditional non-UC (standalone) security framework, it allows the existence of rewinding technique to extract adversary's secrets. On the contrary, UC framework introduces one more role called *environment* and the analyzed protocol is called safe if the *environment* cannot tell the differences between an ideal execution and a real execution. However, this augmented UC framework brings the technical obstacle of disabling the rewinding technique. But the security proof in [21] bypasses this obstacle by defining a *generic* ideal threshold signature functionality $\mathcal{F}_{\mathsf{tsig}}$ (c.f. Figure 14 of [21]), instead of an ECDSA functionality. In this way, they well capture the required proactive security and successfully reintroduce the rewinding technique which greatly simplifies the security analysis. They also define another *global random oracle* functionality $\mathcal{H}$ which is accessible to both real and ideal systems. Moreover, they formalize an *enhanced unforgeability* of ECDSA which is used to prove the non-interactive threshold ECDSA protocol

TABLE 4.4: Key Generation

KeyGen (param)

$\mathcal{P}_i$	Round 1	All players $\{\mathcal{P}_j\}_{j\neq i}$
$(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{CL.KeyGen}(1^\lambda, \mathsf{crs})$ $\pi_i^{\mathsf{RepS}} := \mathsf{ZKPoKRepS}(\mathsf{pk}_i; \mathsf{sk}_i : \mathsf{pk}_i = g_q^{\mathsf{sk}_i})$ $u_i \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$ and $Q_i := u_i P$ $\mathsf{srid}_i \leftarrow \{0,1\}^\kappa$; $\tau_i \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$ and $A_i := \tau_i P$ $u_i' \leftarrow \{0,1\}^\kappa$; $V_i = \mathcal{H}(\mathsf{srid}_i, Q_i, A_i, u_i')$	$\xrightarrow{\mathsf{pk}_i, \pi_i^{\mathsf{Reps}}, V_i}$	Abort if π_i^{Reps} fails.
$\mathcal{P}_i$	Round 2	All players $\{\mathcal{P}_j\}_{j\neq i}$
	$\xrightarrow{\mathsf{srid}_i, Q_i, A_i, u_i'}$	Abort if $\mathcal{H}(\mathsf{srid}_i, Q_i, A_i, u_i') \neq V_i$.
$\mathcal{P}_i$	Round 3	All players $\{\mathcal{P}_j\}_{j\neq i}$
Perform (t-n)-VSS share of u_i: $p_i(X) = u_i + \sum_{k=1}^{t} a_{i,k} X^k \bmod q$ Denote $\{\sigma_{i,j} := p_i(j)\}_{j\in[0,n]}$ and $\{V_{i,k} := a_{i,k} P\}_{k\in[t]}$ $c_i := \mathcal{H}(V_i, \mathsf{srid})$ where $\mathsf{srid} = \oplus_j \mathsf{srid}_j$ $z_i := \tau_i + c_i u_i \bmod q$	$\xrightarrow{\sigma_{i,j}}$ $\xrightarrow{\{V_{i,k}\}_{k\in[t]}, z_i}$	Abort if $\sigma_{i,j} \cdot P \neq \sum_{k=0}^{t} (j^k \cdot V_{i,j})$ or $\mathcal{H}(\mathsf{srid}_i, Q_i, \tilde{A}_i, u_i) \neq V_i$ where $\tilde{A}_i = z_i P - \mathcal{H}(i, \mathsf{srid}) \cdot Q_i$.
$\mathcal{P}_i$	Output	All players $\{\mathcal{P}_j\}_{j\neq i}$
$\{\sigma_{k,i}\}_k$ are additive shares of $x_i := \sum_{k\in[n]} p_k(i)$ where $\{x_i\}_{i\in[n]}$ are (t,n) Shamir shares of x. Output $Q = \prod_j Q_j$		

UC-realizes the functionality $\mathcal{F}_{\mathsf{tsig}}$. We note that this *enhanced unforgeability* unconditionally holds in generic group model (c.f. sec 1.2.5, [21]). Equipped with the above arsenals, they proved by reduction that if their non-interactive threshold ECDSA protocol cannot UC-realize functionality $\mathcal{F}_{\mathsf{tsig}}$, there exists a PPT distinguisher $\mathcal{R}_1$ who can break Paillier's semantic security or a PPT forger $\mathcal{R}_2$ who can win the enhanced ECDSA experiment (c.f. E.1, [21]).

Security of our proposed scheme. The UC security analysis in [21] is transferrable to our CL setting except for some changes during simulation. We

TABLE 4.5: Key Refresh and Auxiliary Info

Key Refresh (param)

$\mathcal{P}_i$	Round 1	All players $\{\mathcal{P}_j\}_{j\neq i}$
$(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{CL.KeyGen}(1^\lambda)$ $x_{i,1}, ..., x_{i,i-1}, x_{i,i+1}, ...x_{i,n} \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$ s.t. $\sum_{j\neq i} x_{i,j} = 0$ $\mathbf{Y}_i = \{Q_{i,j}\}_{j\in[n],j\neq i}$ where $Q_{i,j} := x_{i,j} \cdot P$ $u'_i \leftarrow \{0,1\}^\kappa$ $Vi \leftarrow \mathcal{H}(sid; i; \mathbf{Y}_i; u'_i)$ $\pi_i^{\mathsf{RepS}} := \mathsf{ZKPoKRepS}($ $\mathsf{pk}_i; \mathsf{sk}_i : \mathsf{pk}_i = g_q^{\mathsf{sk}_i})$	$\xrightarrow{\mathsf{pk}_i, V_i, \pi_i^{\mathsf{RepS}}}$	Abort if π_i^{RepS} fails.
$\mathcal{P}_i$	Round 2	All players $\{\mathcal{P}_j\}_{j\neq i}$
$\rho_j \xleftarrow{\$} [0, S]$ $C_{i,j,j\neq i} := \mathsf{Enc}(\mathsf{pk}_j, x_{i,j}; \rho_j)$	$\xrightarrow{\{C_{i,j}\}_{j\in[n],j\neq i}, \mathbf{Y}_i, u'_i}$	Abort if $\sum_{k\in[n],k\neq i} Q_{i,k} \neq 0_{\mathbb{G}}$ or $V_i \neq \mathcal{H}(sid; i; \mathbf{Y}_i; u'_i)$
$\pi_{i,j,j\neq i}^{\mathsf{log}} := \mathsf{ZKPoKLog}((x_{i,j}, \rho_j) :$ $((\mathsf{pk}_j, g, Q_{i,j}, C_{i,j}); (x_{i,j}, \rho_j)) \in \mathcal{R}_{\mathsf{log}})$	$\xrightarrow{\{\pi_{i,j}^{\mathsf{log}}\}_{j\in[n],j\neq i}}$	Abort if $\pi_{i,j}^{\mathsf{log}}$ fails
$\mathcal{P}_i$	Output	All players $\{\mathcal{P}_j\}_{j\neq i}$
$u_i^{\mathsf{new}} := u_i + \sum_{j\in[n],j\neq i} \mathsf{Dec}(\mathsf{sk}_i, C_{i,j}) \mod q$ $Q_i^{\mathsf{new}} := Q_i + \sum_{j\in[n],j\neq i} Q_{j,i}$ Erase previously computed pre-signatures and all Key Refresh data except $u_i^{\mathsf{new}}, Q, \mathsf{pk}_{j,j\in[n]}, \mathsf{sk}_i$		

list below the updates of the 5 simulators, 2 UC-simulators $\mathcal{R}_{\{1,2\}}$ and 3 non-UC simulators $\mathcal{S}^{\{1,2,3\}}$, and omit the full proof here. Then, we demonstrate that our modifications towards [21] do not affect the UC security. Note that in the following descriptions we omit some indices for simplicity.

- **Cancel out ring pederson parameters and its ZKs.** Across the 5 simulators, we unifiably cancel out every (s, t) parameters and its ZK simulator $\mathcal{S}^{\mathsf{prm}}$ since ring pederson commitment is not required in the CL setting.

- **Encryption from Paillier to CL** First, the $\mathcal{R}_1$ is converted to CL distinguisher. Accordingly, the CL distinguisher is parameterized with CL public keys and ciphertexts instead of the Paillier ones. Second, all the encryption/decryption keys, encryption/decryption algorithms, ciphertexts across the three non-UC simulators $\mathcal{S}^{\{1,2,3\}}$ are converted from

TABLE 4.6: Modifications to the Pre-Signing (Part 1)). Note that $S = \tilde{s} \cdot 2^{\epsilon_d}$.

Pre-Signing (Part 1) (param)

$\mathcal{P}_i$	Round 1	All players $\{\mathcal{P}_j\}_{j \neq i}$
$k_i, \gamma_i \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$; $\rho_i, \nu_i \xleftarrow{\$} [0, S]$ $K_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, k_i; \rho_i)$; $G_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \gamma_i; \nu_i)$ $\pi_i^{\mathsf{Enc}} := \mathsf{ZKPoKEnc}((k_i, \rho_i),$ $((\mathsf{pk}_i, K_i); (k_i, \rho_i)) \in \mathcal{R}_{\mathsf{Enc}})$	$\xrightarrow{K_i, G_i, \pi_i^{\mathsf{Enc}}}$	Abort if the π_i^{Enc} fails.
$\mathcal{P}_i$	Round 2	All players $\{\mathcal{P}_j\}_{j \neq i}$
$\Gamma_i = g^{\gamma_i}$ $\beta_{i,j}, \hat{\beta}_{i,j} \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$ $r_{i,j}, s_{i,j}, \hat{r}_{i,j}, \hat{s}_{i,j} \xleftarrow{\$} [0, S]$; $D_{j,i} \leftarrow \mathsf{EvalAdd}(\mathsf{EvalScal}(\gamma_i,$ $K_j), \mathsf{Enc}(\mathsf{pk}_j, -\beta_{i,j}; s_{i,j}))$ $\hat{D}_{j,i} \leftarrow \mathsf{EvalAdd}(\mathsf{EvalScal}(x_i,$ $K_j), \mathsf{Enc}(\mathsf{pk}_j, -\hat{\beta}_{i,j}; \hat{s}_{i,j}))$ $F_{j,i} \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \beta_{i,j}; r_{i,j})$; $\hat{F}_{j,i} \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \hat{\beta}_{i,j}; \hat{r}_{i,j})$ $\pi_i^{\mathsf{log}} := \mathsf{ZKPoKLog}((\gamma_i, \nu_i) :$ $((\mathsf{pk}_i, g, \Gamma_i, G_i); (\gamma_i, \nu_i)) \in \mathcal{R}_{\mathsf{log}})$ $\pi_{j,i}^{\mathsf{Aff\text{-}p}} := \mathsf{ZKPoKAff\text{-}p}($ $(\gamma_i, \beta_{i,j}, \nu_i, r_{i,j}, s_{i,j}) :$ $((G_i, F_{j,i}, K_j, D_{j,i}, \mathsf{pk}_i, \mathsf{pk}_j);$ $(\gamma_i, \beta_{i,j}, \nu_i, r_{i,j}, s_{i,j})) \in \mathcal{R}_{\mathsf{Aff\text{-}p}})$ $\pi_{j,i}^{\mathsf{Aff\text{-}g}} := \mathsf{ZKPoKAff\text{-}g}($ $(x_i, \hat{\beta}_{i,j}, \hat{r}_{i,j}, \hat{s}_{i,j}) :$ $((G_i, F_{j,i}, K_j, D_{j,i}, \mathsf{pk}_i, \mathsf{pk}_j, g^{x_i});$ $(x_i, \hat{\beta}_{i,j}, \hat{r}_{i,j}, \hat{s}_{i,j})) \in \mathcal{R}_{\mathsf{Aff\text{-}g}})$	$\xrightarrow{\pi_i^{\mathsf{log}}, \Gamma_i}$ $\xrightarrow{D_{j,i}, \hat{D}_{j,i}, F_{j,i}, \hat{F}_{j,i}, \pi_{j,i}^{\mathsf{Aff\text{-}g}}, \pi_{j,i}^{\mathsf{Aff\text{-}p}}}$	Abort if any proof fails.

TABLE 4.7: Modifications to the Pre-Signing (Part 2)).

Pre-Signing (Part 2) (param)		
$\mathcal{P}_i$	Round 3	All players $\{\mathcal{P}_j\}_{j\neq i}$
$\Gamma := \prod_j \Gamma_j$; $\quad \Delta_i = \Gamma^{k_i}$ $\alpha_{i,j} \leftarrow \mathsf{Dec}(\mathsf{sk}_i, D_{i,j})$; $\hat{\alpha}_{i,j} \leftarrow \mathsf{Dec}(\mathsf{sk}_i, \hat{D}_{i,j})$ $\delta_i := \gamma_i k_i + \sum_{j\neq i}(\alpha_{i,j} + \beta_{i,j}) \mod q$ $\chi_i := x_i k_i + \sum_{j\neq i}(\hat{\alpha}_{i,j} + \hat{\beta}_{i,j}) \mod q$ $\pi_i^{\mathsf{log}'} := \mathsf{ZKPoKLog}((k_i, \rho_i) :$ $((\mathsf{pk}_i, \Gamma, \Delta_i, K_i); (k_i, \rho_i))) \in \mathcal{R}_{\mathsf{log}})$ Erase all items in memory except for the stored state.	$\xrightarrow{\delta_i, \Delta_i, \pi_i^{\mathsf{log}'}}$	Abort if $\pi_i^{\mathsf{log}'}$ fails.
$\mathcal{P}_i$	Output	All players $\{\mathcal{P}_j\}$
$\delta := \sum_j \Delta_j$ Verify $g^\delta = \prod_j \Delta_j$, abort otherwise Set $R := \Gamma^{\delta^{-1}}$ Output (R, k_i, χ_i). Erase all items except the store state.		

TABLE 4.8: Signing protocol.

Signing (param)		
$\mathcal{P}_i$	Round 1	All players $\{\mathcal{P}_j\}_{j\neq i}$
$r = R\|_{\text{x-axis}}$ $\sigma_i = km + r\chi$ Erase (R, k, χ).	$\xrightarrow{\sigma_i}$	$\sigma = \sum_j \sigma_j$ Verify (r, σ) is a valid signature, return (m, r, σ) if valid, abort otherwise.

Paillier setting to CL setting. For example, each (p, q) is transformed to sk (CL secret key).

- **ZK-simulators from Paillier to CL**. Across the five simulators, first, redefine the proofs π^{enc}, π^{log} and π^{aff} in CL setting instead of Paillier setting, equivalently for their ZK simulators $\mathcal{S}^{\mathsf{enc}}$, $\mathcal{S}^{\mathsf{log}}$ and $\mathcal{S}^{\mathsf{aff}}$; second, since we have already replaced all the Paillier secret key (p, q) to CL secret key sk, the ZK proof π^{mod} and ZK simulator $\mathcal{S}^{\mathsf{mod}}$ should be updated to π^{RepS} and $\mathcal{S}^{\mathsf{RepS}}$ respectively.

We emphasize here that our above updates in security analysis has no affections violating the requirements in analysis of [21], as shown in (i) and (ii);

and that some assumptions should be tuned in claim 5.5 and 5.6 in [21] as shown in (iii).

(i) The construction of standalone (non-UC) simulators $\mathcal{S}^{\{1,2,3\}}$ treat each ZK as a module without specifying its inner setting as in in Sec 5.5 of [21]. Thus, our updates in ZK proofs, simulators and also encryption schemes do not have material influence during the simulation.

(ii) The transition from π^{mod} to π^{RepS} keeps the *extractability requirement* in the UC simulators $\mathcal{R}^{\{1,2\}}$; and further, generic group model is required in the extractability analysis of π^{RepS} but still the enhanced ECDSA assumption unconditionally holds in generic group model as aforementioned;

(iii) In the output phase of simulator $\mathcal{S}^1$, it should fulfill an environment secrets extraction and the secrets are the encrypted plaintexts. This kind of extractability is promised by the special soundness of their encryption well-formedness ZK proof. More specifically, it relies on the *strong RSA assumption*. Switch to our setting, the special soundness of our ZKPoKEnc is assuerd by three assumptions: adaptive root subgroup assumption, Corollary 2.1 and Corollary 2.2. Hence, without changing much of the proof sketch of sec 5.3.1 of [21], we simply adapt these three assumptions to replace the original *strong RSA assumption*, more specifically, for Lemma 5.4 and Claim 5.5 in [21]. Also, it is trivial to replace the semantic security of Paillier encryption with the semantic security with CL encryption, which produces the noticeable winning probability of a CL distinguisher $\mathcal{R}^1$.

Hence, we conclude the security claim of our new threshold ECDSA scheme in Theorem 4.7, where we also follow the definition of proactive, ideal threshold signature functionality $\mathcal{F}_{\mathsf{tsig}}$ which follows the same definition of Figure 14, [21].

Theorem 4.7. *Assuming semantic security of the CL cryptosystem, adaptive root subgroup assumption, Corollary 2.1 and Corollary 2.2, and enhanced existential unforgeability of ECDSA, it holds that the non-interactive threshold ECDSA modified in this work UC-realizes the functionality $\mathcal{F}_{\mathsf{tsig}}$, in the presence of the global random oracle functionality H.*

4.4.4 Bandwidth Analysis

In this subsection, we analyse the theoretical complexity of our ZK proofs and modified ECDSA. We compare the communication bandwidth of our protocol with the one from Canetti *et al.* [21] corresponding to the level of security 128, and under soundness error 2^{-80}.

Zero-knowledge Proofs. We compare the communication bandwidth of the

TABLE 4.9: Message Sizes in bits under soundness error of 2^{-80}.

Message Type	[21]	This work
Public key	9216	2339
Ciphertext	6144	4678
ZKPoKRepS (n=1)	-	5062
ZKPoKMod	494752	-
ZKPoKPrm	491520	-
ZKPoKEnc	19971	9996
ZKPoKLog	20227	10252
ZKPoKAff-g	42244	25310
ZKPoKAff-p	51204	36877

TABLE 4.10: Bandwidth Analysis in bits under n-party setting in 128-bit security level.

Protocol	[21]	This work
Key generation	$256\ n^2 + (1497920 + 256t)\ n$	$256\ n^2 + (8681 + 256t)\ n$
Key refreshment	$20227\ n^3 - 34054\ n^2 + 1009955\ n$	$20504\ n^2 - 12591\ n$
Pre-signing	$118024\ n^2 - 44543\ n$	$80899\ n^2 - 40275\ n$

ZK proofs proposed in [21] and this work. We observe that, for the public key and ciphertext, the communication bandwidth is reduced by around 74.6% and 23.9%, respectively. For the ZK proofs related to encryption, the communication bandwidth is improved between the range of 28% and 50%. A detailed communication bandwidth analysis is shown in Table 4.9.

Components in threshold ECDSA. We compare the communication bandwidth of the key generation, key refreshment, and pre-signing components in threshold ECDSA. The results are shown in Table 4.10 and Figure 4.1. As the number of parties increased, the communication bandwidth of the threshold ECDSA proposed by Canetti *et al.* [21] grows much higher than that of ours, thus the modification from ours save a large amount of communication bandwidth. Note that in the picture for key generation, the two curves seem linear and like a mismatch with Table 4.10. This is due to that when n is not large enough, the trend will be dominated by the coefficient of n instead of the coefficient of n^2.

FIGURE 4.1: Bandwidth in 128-bit security level where t is set $n-1$

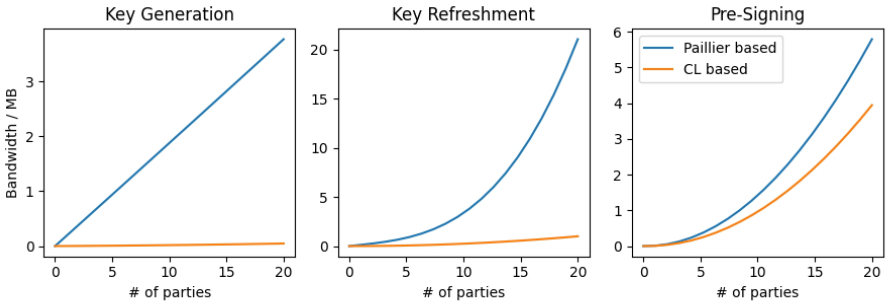


TABLE 4.11: Running time.

(t,n)	Protocol	KeyGen	KeyRefresh	PreSign	OnlineSign
(1,3)	[21]	36.5 s	39.4 s	6.2 s	1.2 ms
	our scheme	3.7 s	13.9 s	80.5 s	1.2 ms
(2,4)	[21]	54.1 s	62.2 s	11.9 s	1.6 ms
	our scheme	5.9 s	26.9 s	155.2 s	1.6 ms
(2,5)	[21]	72.1 s	89.9 s	19.7 s	2.1 ms
	our scheme	8.4 s	42.7 s	253.8 s	2.1 ms

4.4.5 Implementation

We implemented both the Paillier-based threshold ECDSA [21] and our CL-based threshold ECDSA in Rust using a MacBook Pro laptop with 16G RAM and Intel Core i5. We use the Bitcoin secp256k1 curve[5] and number theory library[6] for large integer operations. Our program is yet to be parallelized and instead, we linearly execute each party's operations in each round without considering the network issue. We choose the (t,n) settings with (1,3), (2,4) and (2,5) which are the most popular settings in Bitcoin's P2SH transactions.

As can be seen from Table 4.11, regarding running time, key generation is reduced by up to a factor of 10, and KeyRefresh is also improved by approximately 50%. However, the running time of pre-signing is over 10 times longer than its Paillier counterpart (although parallel computation can reduce this time by dividing the number of parties). This is the sole cost of enhancing the bandwidth across all phases and improving the running time in key generation and key refreshment. It's less critical in a non-interactive signing scenario where the focus is on optimal online signing, even if the computation assumption in pre-signing can be relaxed. We plan to further optimize this less efficient CL-based pre-signing in future work. Regarding online signing, our scheme aligns with [21].

[5]https://github.com/rust-bitcoin/rust-secp256k1
[6]https://docs.rs/rust-gmp/0.5.0/gmp/mpz/struct.Mpz.html

4.5 Conclusions

In this research, we've optimized existing ZK proofs in CL-based threshold ECDSA in terms of both proof size and proving time. We have also introduced a novel ZK proof for the affine transformation in CL ciphertext. Finally, we have extended these three components to create tailored ZK proofs for constructing a bandwidth-optimal, UC secure, non-interactive, and proactive threshold ECDSA. This outperforms its Paillier-based counterpart in terms of bandwidth across all phases and running time during key generation and refreshment, with the trade-off of a slower pre-signing process.

Chapter 5

Trapdoorless GQ Multi-Signature with Identifiable Abort

5.1 Motivation

Guillou-Quisquater signature, for simplicity called GQ, was proposed in 1988 [46]. GQ has some applications in cryptographic protocols such as forward-secure signature [52], identity-based signature with bounded life-span [32], distributed certificate status protocol [80], distributed authentication algorithm for mobile ad-hoc network [76], GQ1 (identity-based) and GQ2 schemes in ISO/IEC 14888-2 standard (certificate based) [50] and etc. GQ has already been used to construct distributed signing protocols, including multi-signature schemes [6, 31, 72] and threshold signature schemes [27, 57, 74]. However, when compared to Schnorr and ECDSA—which are the most widely used digital signature schemes due to Schnorr's remarkable simplicity and ECDSA's application in blockchains like Bitcoin and Ethereum—the application scenarios and research discussions surrounding GQ are still rather limited.

Drawbacks of RSA-based GQ. One outstanding drawback of GQ is that its RSA-based system design incurs some trust problems. The system parameter cannot be developed or chosen by non-trusted developers, otherwise system developers will hold a trapdoor ($n = pq$) which can be utilized to crack the whole cryptosystem, unlike Schnorr and ECDSA where they can securely resort to a non-trusted system developer to select the trapdoorless cyclic group like secp256k1 curve. For the aforementioned schemes using GQ, distributed or identity based GQ signings expect for [31] either require trusted dealer or require servers be honest majority, which is unrealistic to be applied

in trustless development environments, such as a public blockchain or a digital wallet. We remark that the scheme in [31] possibly the first trustless GQ multi-signature setting, at a cost of introducing one more entity called combiner, the malicious behaviour of which can be detected by signers. Another plausible way to achieve trustless setup is to use distributed RSA key generation methods like in [12, 29, 40]. The reason why we did not adopt them to construct GQ multi-signature is that they require most participated parties are honest. This contradicts the trustless setting. Next, we introduce a more intuitive way without using additional entity or requiring honest majority to achieve trustless GQ.

In 2000, Hamdy and Möller [47] informally suggested that class groups of imaginary quadratic fields (IQC) proposed by Buchmann and Williams [18] could be utilized in GQ signatures. This insight illuminated a pathway to eliminating the RSA trapdoor in the GQ signature scheme, specifically by replacing the RSA group in GQ signatures with a class group. However, such a class group-based GQ signature currently lacks a formal definition, as well as a rigorous security proof for Existential Unforgeability under Chosen Message Attack (EUF-CMA) under a suitable hardness assumption.

Another limitation of GQ protocols lies in their bandwidth efficiency, particularly in multi-user settings. Given that all elements in an RSA group of order n have to be represented by a 3072-bit string to achieve 128-bit security, the bandwidth requirements can be substantial. On the other hand, in the realm of class groups, a group element only requires a tuple (a, b), which can be denoted by a 1665-bit string, alongside a 1665-bit discriminant Δ which only needs to be declared once. Therefore, transitioning from an RSA group to a class group can reduce the bandwidth by 45.8% for each group element. This makes the use of GQ in a trustless distributed setting considerably more attractive. Moreover, to achieve a trustless distributed GQ signing in highly malicious environment, removing the RSA trapdoor is still not enough, because any signer may perform maliciously and finally lead to a collaborative signature invalid. It is essential to allow honest signers detect and reject incorrect/malicious messages.

Intuitions. In this work, we focus on constructing a *trustless* multi-signature scheme, allowing *key aggregation* and *identifiable abort* properties.

1. *Trustless property* requires a non-trusted setup and security against the existence of any number of malicious participants during all phases (for both key generation or signing).

2. System abort is not considered as a violation of the security definition in [45]. Consequently, a malicious adversary could easily launch a Denial

of Service (DoS) attack on the system. As such, we require an *identifiable abort property*, as defined in [49]. This property ensures that the identities of malicious participants leading to a system abort can be detected by any participants or external entities. This detection capability is crucial for identifying broken or hacked devices, or misbehaving banks or institutions, which could cause the failure of joint signing.

3. Moreover, we aim for our scheme to support *key aggregation*. This implies that a signer, rather than using a full list of public keys (or key shares), only needs an aggregated public key for everyone to verify a signature. This approach saves computational resources and storage, making it particularly beneficial for devices with limited computing capabilities.

5.2 Our Contributions

(1) Formal definition and security proof for class group based GQ signature (CL-GQ). (1) The application of a class group to a GQ signature can render GQ trapdoorless, as suggested in [47], but no formal definition has been put forward. We begin by formalizing the definition of a GQ signature over the class group of imaginary quadratic fields. We then identify the appropriate hardness assumption, the *prime root assumption*, for CL-GQ. We also provide a proof of existential unforgeability under a chosen message attack (EUF-CMA) in the random oracle model (ROM) under the *prime root assumption*, which is implied by the *root assumption* in a generic group as detailed in [28].

(2) Compact one-round NIZK proofs to resist malicious adversaries and achieve identifiable abort. In order to detect the malicious behaviour during the multi-party signing and the protocol can abort once misbehaving is detected once the malicious message is recieved (a timely identifiable abort with attributability to the exact malicious message), we design two tailored ZK proofs including ZKPoKRoot and ZKPoKSig following the 3 moves in the traditional Σ-protocol. They promise any messages sent during interactions are verifiable. Our Zero-Knowledge (ZK) proofs are remarkably efficient, as the adoption of a Bezout trick eliminates the need for repetition, even though the ZK proofs operate in an unknown order class group. This differs from the binary challenge-based ZK proofs presented in [22, 23]. The Bezout trick beautifully addresses the challenge of accelerating the ZK proof of Paillier ciphertext utilised in Yi's blind ECDSA [77].

(3) Provably secure trustless CL-GQ multi-signature in dishonest majority model. We extend CL-GQ to accommodate a multi-user setting,

and integrate the non-malleable equivocable commitment used in [23, 42] along with our ZK proofs to construct our trustless CL-GQ multi-signature scheme. Our scheme does not depend on any common reference string (CRS) generated by a trusted party. We link the unforgeability of our new multi-signature scheme in the *dishonest majority model* to the EUF-CMA of CL-GQ under ROM. Different from the *double-forking technique* that necessitates a two-layer rewinding framework used in [59, 63], we have no Hash oracle query rewinding when reducing the CL-GQ multi-signature to CL-GQ, which saves the reduction loss. Furthermore, our proof is more succinct than the ECDSA schemes [23, 42] as our simulator does not need to distinguish any non semi-correct executions.

(4) Implementation and efficiency analysis. We implement our protocol in Rust[1] to demonstrate the practical efficiency. One signer only needs 2.1/3.6 seconds to sign a document for 112/128-bit security level in a 5-user setting. We also analyze the concrete bandwidth needed in our scheme. In 128-bit security, our protocol only costs 6 kB (kilobytes) and 10 kB bandwidth for the interactive key generation and interactive signing phases respectively in a 5-signer setting. For signing, the bandwidth of our scheme is about one-third of the bandwidth in [42] since we do not have expensive range proofs led by Paillier encryption or tedious MtA (Multiplication-to-Addition) protocol led by the non-linear structure of ECDSA. Both running time and bandwidth are promising.

5.3 GQ Signature Scheme without Trapdoor (CL-GQ)

When we replace the RSA group by class group of imaginary quadratic field $CL(\Delta)$, the group order and thus factoring of group order are unknown even to the authority or user who generates the group. Hence, this $n = pq$ trapdoor is perfectly removed. The GQ signature based on class group is portraited below. The main difference between GQ and CL-GQ is in the KeyGen phase, where v has to be a prime and the group is initialized by a prime Δ. Procedures in sign and verification are basically the same as GQ's. But group operations in class group eliminates the necessity of computing modulo. We now describe the details.

- KeyGen. Given the security parameter λ, find a λ-bit prime $-\Delta$ s.t. $\Delta \equiv 1 \mod 4$ and a δ-bit prime v. Randomly sample a generator B from class group of imaginary quadratic field $CL(\Delta)$. Compute $J =$

[1] https://www.rust-lang.org/

B^{-v}. Notice that all the multiplication and exponentiation in class group should be finalized to a reduced form. It is for the unity of representation and to lower computation cost. Choose a hash function $H : \{0,1\}^* \rightarrow H : \{0,1\}^{\delta-1}$. Set $PK = (\Delta, v, J, H)$ and $SK = (B)$.

- Sign. On input the secret key B and a message M, randomly selects r from $CL(\Delta)$, then compute $T = r^v$, $h = H(M, T)$ and $t = rB^h$. Output signature $\sigma = (t, h)$.
- Verify. Upon receiving a signature $\sigma = (t, h)$ of message M, compute $T' = t^v J^h$ and $h' = H(M, T')$. If $h' = h$, output 1; otherwise, output 0.

Security. Damgård and Koprowski defined *root assumption* [28] working in generic group model, as a generalization of RSA assumption, by describing that given a group element $x \in G$ and a number $e > 1$, finding a group element y s.t. $y^e = x$ is intractable, where G is a finite Abelian group in which the inverse and multiplication can be efficiently computed. Thus, we define a *prime root assumption* as below, working in class group, which rules out composite exponent and can be directly implied by *root assumption*. By Theorem 1, the EUF-CMA security of CL-GQ can be reduced to *prime root assumption* in ROM.

Definition 5.1 (Prime root assumption)**.** We say that a class group of imaginary quadratic fields satisfies *prime root assumption* for any efficient $\mathcal{A}$ if

$$\Pr\left[u^v = g : u \leftarrow \mathcal{A}(\Delta, g, v), v \leftarrow \mathsf{Primes}(\delta), g \xleftarrow{\$} CL(\Delta), \Delta \xleftarrow{\$} \mathsf{Primes}^*(\lambda)\right]$$

is negligible in λ.

where $\mathsf{Primes}(\delta)$ is the set of primes less than 2^δ, here we require $\delta \geq \eta(\lambda)$ and $\eta(\lambda)$ is the output length of secure Hash function used in signature scheme which will be given in the later section; and $\mathsf{Primes}^*(\lambda)$ is the set of λ-bit primes which are equal to 3 modulo 4.

Theorem 5.2. *If prime root assumption holds and H is a random oracle, the CL-GQ signature is provably secure in the EUF-CMA model.*

Proof. Suppose $\mathcal{B}$ is given a prime root problem instance (Δ, J^*, v), J^* is a group member in $CL(\Delta)$ and v is a prime. $\mathcal{B}$ tries to find a B^* from $CL(\Delta)$ s.t. $B^{*v} = J^*$ by using an EUF-CMA adversary $\mathcal{A}$ against the CL-GQ signature scheme.

Setup. $\mathcal{B}$ prepares an empty list $\mathcal{H}$, set p as the length of each element in $\mathcal{H}$. $\mathcal{B}$ sends (Δ, v, J^*, H) to adversary $\mathcal{A}$ as the public key.

Oracle Query. $\mathcal{B}$ answers the oracle queries as follows:

- Sign: On input a message M, $\mathcal{B}$ picks some random $t \in CL(\Delta), h \in \mathbb{Z}_p$ and computes $T = t^v J^{*h}$. $\mathcal{B}$ puts (h, T, M) in the list $\mathcal{H}$. (If the value of h is already set in $\mathcal{H}$, $\mathcal{B}$ picks another h and repeats the previous step.) $\mathcal{B}$ returns $\sigma = (t, h)$.
- H: On input (T, M), if (h, T, M) is in the list $\mathcal{H}$, $\mathcal{B}$ returns h. Otherwise, $\mathcal{B}$ picks a random $h \in \mathbb{Z}_p$. $\mathcal{B}$ puts (h, T, M) in the list $\mathcal{H}$ and returns h.

Output. Finally $\mathcal{A}$ outputs an a message M^* and a forged signature $\sigma^* = (t^*, h^*)$. $\mathcal{B}$ can compute $h^* = H(T^*, M^*)$ s.t. $T^* = t^{*v} J^{*h^*}$.

$\mathcal{B}$ rewinds H to the point that (T^*, M^*) was queried, and returns a different $h' \neq h^*$. $\mathcal{B}$ eventually obtains another forgery (t', h') from $\mathcal{A}$. Therefore, we have $t^{*v} J^{*h^*} = t'^v J^{*h'}$ and it can be transformed into $J^{*h^*-h'} = (t'/t^*)^v$.

According to Bezout formula, there exists a unique pair of non-zero integers (k, m) where $0 \leq |k| \leq v - 1$ and $0 \leq |m| \leq |h^* - h'| - 1$ which is easily computed by Euclidean algorithm s.t.:

$$mv - k(h^* - h') = gcd(v, h^* - h') = 1.$$

Raise equation $J^{*h^*-h'} = (t'/t^*)^v$ to power k, we have:

$$\begin{aligned} J^{*k(h^*-h')} &= (t'/t^*)^{vk} \\ J^{*mv-1} &= (t'/t^*)^{vk} \\ J^* &= \{J^{*m}(t^*/t')^k\}^v \end{aligned}$$

Hence, $\mathcal{B}$ successfully extracts $B^* = J^{*m}(t^*/t')^k$ to solve the problem instance. □

5.4 Our Multi-Signature Scheme

In this section, we give the construction of our multi-signature scheme, which is a trustless GQ multi-signature with identifiable abort, secure in dishonest majority model. Both distributed key generation and distributed signing have six phases, they will either abort or output a CRS and a valid signature in each phase. We also utilize two zero-knowledge proofs ZKPoKRoot and ZKPoKSig in our protocol, which will be described in details in next section.

The proposed multi-signature scheme operates under a dishonest majority model allowing static corruption, as used in [23, 42, 43, 55]. Following [44], we use a game-based definition of security analogous to EUF-CMA (Existential

TABLE 5.1: Interactive Key Generation Protocol IKeyGen

	$\mathsf{IKeyGen}(\lambda)$	
P_i		All users $\{P_j\}, i \neq j$
$\delta_i \xleftarrow{\$} \{0,1\}^\lambda$		
$v_i \xleftarrow{\$} \{0,1\}^{\eta(\lambda)+1}$		
$[c_i, d_i] \leftarrow \mathsf{Com}(\delta_i)$		
$[\hat{c_i}, \hat{d_i}] \leftarrow \mathsf{Com}(v_i)$	$\xrightarrow{c_i, \hat{c_i}}$	
	$\xrightarrow{d_i, \hat{d_i}}$	$\delta_i \leftarrow \mathsf{Reveal}(c_i, d_i)$
		$v_i \leftarrow \mathsf{Reveal}(\hat{c_i}, \hat{d_i})$
$\Delta = \mathsf{NextPrime}^*(\oplus_{i=1}^n \delta_i)$		
$v = \mathsf{NextPrime}(\oplus_{i=1}^n v_i)$		
$B_i \xleftarrow{\$} CL(\Delta)$		
$J_i = B_i^{-v}$		
$[c_i^*, d_i^*] \leftarrow \mathsf{Com}(J_i)$	$\xrightarrow{c_i^*}$	
	$\xrightarrow{d_i^*}$	$J_i \leftarrow \mathsf{Reveal}(c_i^*, d_i^*)$
$\pi_i = \mathsf{ZKPoKRoot}((J_i, v) : B_i \mid J_i = B_i^{-v})$	$\xleftrightarrow{\pi_i}$	Abort if proof π fails
$J = \prod_{i=1}^n J_i$		
Set $CRS = (\Delta, v, J, H)$,		
and $PK_i = J_i; SK_i = B_i$		

Unforgeability under Chosen Message Attacks): multi-signature unforgeability under chosen message attacks (MU-CMA), as stated in Section 2.14.

Parameters and notations. For the security level of 80/112/128-bit security, we set λ (the bit length of the discriminant Δ of class group) 958/1208/1665 according to Appendix C and set $\eta(\lambda)$=160/224/256 bits. Considering the requirement in [46] that h is smaller than v, h and v are set $\eta(\lambda)$ and $\eta(\lambda)+1$ bits respectively. $\mathsf{NextPrime}(x)$ (resp. $\mathsf{PrevPrime}(x)$) is a function using Miller-Rabin prime test to generate the next (resp. previous) nearest prime. $\mathsf{NextPrime}^*(x)$ (resp. $\mathsf{PrevPrime}^*(x)$) is a function using Miller-Rabin prime test to generate the next (resp. previous) nearest prime r such that $r \equiv 1 \mod 4$ after the input integer x. $\mathsf{Com}(x)$ is a non-malleable commitment for a committed value x and $\mathsf{Reveal}(c, d)$ opens the underlying committed value of the non-malleable equivocal commitment where c is a commitment and d is a decommitment.

5.4.1 Distributed Key Generation

Our distributed key generation algorithm (Table 5.1) will either abort or output a CRS. ZKPoKRoot is used to promise that public key J_i broadcasted by

TABLE 5.2: Interactive Signing Protocol ISign

	$\mathsf{ISign}(\lambda, SK, M)$	
P_i		All users $\{P_j\}, i \neq j$
$r_i \xleftarrow{\$} CL(\Delta)$		
$T_i = r_i^v$		
$[c_i, d_i] \leftarrow \mathsf{Com}(T_i)$	$\xrightarrow{c_i}$	
	$\xrightarrow{d_i}$	$T_i \leftarrow \mathsf{Reveal}(c_i, d_i)$
$\pi_i = \mathsf{ZKPoKRoot}((T_i, v) : r_i \vert T_i = r_i^v)$	$\xleftrightarrow{\pi_i}$	Abort if proof π fails
$T = \prod_{i=1}^n T_i$		
$h = \mathsf{H}(M, T)$		
$t_i = r_i B_i^h$		
$[\hat{c}_i, \hat{d}_i] \leftarrow \mathsf{Com}(t_i)$	$\xrightarrow{\hat{c}_i}$	
	$\xrightarrow{\hat{d}_i}$	$t_i \leftarrow \mathsf{Reveal}(\hat{c}_i, \hat{d}_i)$
$\hat{\pi}_i = \mathsf{ZKPoKSig}((T_i, J_i, t_i, h, v) : (r_i, B_i) \vert$		
$t_i = r_i B_i^h, T_i = {r_i}^v, J_i = {B_i}^{-v})$	$\xleftrightarrow{\hat{\pi}_i}$	Abort if proof $\hat{\pi}$ fails
$t = \prod_{i=1}^n t_i$		
Output $\sigma = (t, h)$		

party P_i is correctly generated. We describe the details as follows.

Phase 1. Each party P_i picks $\delta_i \xleftarrow{\$} \{0,1\}^\lambda$ and $v_i \xleftarrow{\$} \{0,1\}^{\eta(\lambda)+1}$. P_i computes the commitment $[c_i, d_i] \leftarrow \mathsf{Com}(\delta_i)$ and $[\hat{c}_i, \hat{d}_i] \leftarrow \mathsf{Com}(v_i)$. Each P_i broadcasts to all other parties the commitment $(c_i, \hat{c}_i)$.

Phase 2. Each P_i broadcasts the decommitment $(d_i, \hat{d}_i)$ to all other parties.

Phase 3. After each P_i received all the (δ_j, v_j) generated by every $P_j (j \neq i)$, a collaboratively generated (Δ, v) is computed by $\Delta = \mathsf{NextPrime}^*(\oplus_{i=1}^n \delta_i)$ and $v = \mathsf{NextPrime}(\oplus_{i=1}^n v_i)$. Then, each P_i generate its key pair (B_i, J_i) by $B_i \xleftarrow{\$} CL(\Delta)$ and $J_i = B_i^{-v}$. P_i computes the commitment $[c_i^*, d_i^*] \leftarrow \mathsf{Com}(J_i)$ and broadcasts to all other parties the commitment c_i^*.

Phase 4. Each P_i broadcasts the decommitment d_i^* along with a non-interactive zero-knowledge proof π_i for the relation $\{(J_i, v) : B_i \vert J_i = B_i^{-v}\}$ to all other parties.

Phase 5. Upon receiving π_i from $P_j (j \neq i)$, each P_i checks the validity of π_j. If passing the check, P_i accepts π_j; otherwise, abort.

Phase 6. After each P_i received all the π_j generated by every $P_j (j \neq i)$ and every π_j's validity is proved, a common J is computed by $J = \prod_{i=1}^n J_i$. Each party P_i sets $CRS = (\Delta, v, J), PK_i = J_i; SK_i = B_i$.

5.4.2 Distributed Signing

Our distributed signing algorithm (Table 5.2) will either abort or output a valid signature. We use ZKPoKRoot to ensure the well-formedness of commitment T_i and use ZKPoKSig to ensure the well-formedness of response t_i, thus preventing malicious behaviors during the signing phase. We describe the details as follows.

Phase 1. Each party P_i picks $r_i \xleftarrow{\$} CL(\Delta)$ and compute $T_i = r_i^v$. P_i computes the commitment $[c_i, d_i] \leftarrow \mathsf{Com}(T_i)$. Each P_i broadcasts to all other parties the commitment c_i.

Phase 2. Each P_i broadcasts the decommitment d_i along with a non-interactive zero-knowledge proof π_i for the relation $\{(T_i, v) : r_i | T_i = r_i^v\}$ to all other parties.

Phase 3. Upon receiving π_j from $P_j (j \neq i)$, P_i checks the validity of each π_j. If it is valid, P_i accepts π_j ; otherwise, abort.

Phase 4. After each P_i received all the T_j and π_j generated by every $P_j (j \neq i)$ and π_j is proved valid, a common $T = \prod_{i=1}^{n} T_i$ is computed. Then, calculate $h = \mathsf{H}(M, T)$. Each P_i computes $t_i = r_i B_i^h$ and the commitment $[\hat{c}_i, \hat{d}_i] \leftarrow \mathsf{Com}(t_i)$. Each P_i broadcasts to all other parties the commitment $\hat{c}_i$.

Phase 5. Each P_i broadcasts the decommitment $\hat{d}_i$ along with a non-interactive zero-knowledge proof $\hat{\pi}_i$ for the relation $\{(T_i, J_i, t_i, h, v) : (r_i, B_i) | t_i = r_i B_i^h, T_i = {r_i}^v, J_i = {B_i}^{-v}\}$ to all other parties.

Phase 6. Upon receiving $\hat{\pi}_j$ from $P_j (j \neq i)$, each P_i checks the validity of $\hat{\pi}_i$. If it is valid, P_i accepts $\hat{\pi}_i$; otherwise, abort. Each party computes $t = \prod_{i=1}^{n} t_i$. Output the collaborative signature $\sigma = (t, h)$.

5.4.3 Verification

When receiving a signature $\sigma = (t, h)$ for the message M, the verification is similar to the original GQ signature scheme. Accept if and only h is equal to $\mathsf{H}(M, T')$ where $T' = t^v J^h$. The correctness follows by $T' = t^v J^h = (\prod_{i=1}^{n} t_i)^v (\prod_{i=1}^{n} J_i)^h = (\prod_{i=1}^{n} r_i B_i^h)^v (\prod_{i=1}^{n} B_i^{-v})^h = (\prod_{i=1}^{n} r_i)^v = r^v = T$. Since the operation is based on an unknown order class group and the results produced by class group multiplication and exponentiation is normalized when output, we do not need to modulo the result by any integer. Since the validity of the signature can be checked by any P_j, it is possible for P_i to send P_j the signature if it confirms the validity of this signature. This will not affect security at all. Moreover, non-malleable commitments and zero-knowledge

proofs promise that each party cannot deny the message it broadcasts to the network and each message contributing to collaboratively generated signature is well-formed, and thus no malicious behaviors can affect the joint signing. Note that, the verification phase only needs the aggregated key $J = \prod_{i=1}^{n} J_i$, not the full list of signers' public keys $\{J_i\}_{i\in[1,n]}$.

5.4.4 Rogue-Key Attack Resistant

In the IKeyGen phase, an adversary, P_{j^*} for example, cannot choose its PK_{j^*} after seeing the public keys of other parties to initiate rogue-key attack. More specifically, he cannot set his public key as $J_{j^*} = B_{j^*}^{-v}(\prod_{i=1,i\neq j^*}^{n} J_i)^{-1}$ and thus make the aggregated key equal his arbitrarily selected public key $B_{j^*}^{-v}$, in which case he can forge valid multi-signature by himself easily, since he cannot prove the knowledge of the discrete logarithm of J_{j^*} by submitting valid ZKPoKRoot. This rules out the possibility of rogue-key attack following the KOSK assumption.

5.5 Security Proof of Our Multi-Signature Scheme

The security proof of our multi-signature scheme is a reduction to the unforgeability of CL-GQ. If there is a PPT adversary $\mathcal{A}$ which breaks our multi-party CL-GQ, then we can construct a forger $\mathcal{F}$ to use $\mathcal{A}$ to break CL-GQ. $\mathcal{F}$ must simulate the environment of $\mathcal{A}$. Namely, when $\mathcal{A}$ corrupts $\{P_j\}$ where $j \neq 1$, we can construct a $\mathcal{F}$ to simulate honest party P_1 s.t. $\mathcal{A}$'s view of interaction with $\mathcal{F}$ is indistinguishable from $\mathcal{A}$'s view of interaction with P_1. Let $\mathcal{F}$ have the public key (Δ, v, J, H) of CL-GQ and owns the access to the signing oracle of its choice. After a series of queries from $\mathcal{F}$, it can output a forgery signature $\sigma = (t, h)$ for a message M chosen by itself which has never been queried. Different from the security proof of the multiparty ECDSA in [23], $\mathcal{F}$ does not need to distinguish a semi-correct or non semi-correct execution of $\mathcal{A}$ (δ_i in Phase 3, Fig 5 in [23] sent from adversary can be malicious) which makes our proof more concise.

5.5.0.1 Simulating P_1 in IKeyGen.

$\mathcal{F}$ obtains a public key (Δ, v, J, H) from its CL-GQ challenger and he must set up in its simulation with $\mathcal{A}$ this same public key (Δ, v, J, H). This will allow $\mathcal{F}$ to subsequently simulate interactively signing messages with $\mathcal{A}$, using the output of its CL-GQ signing oracle. $\mathcal{F}$ repeats the following steps by

rewinding $\mathcal{A}$ until $\mathcal{A}$ sends the correct decommitments for $P_2, ..., P_n$ on both iterations.

1. $\mathcal{F}$ randomly selects $\delta_1 \in \{0,1\}^\lambda$ and $v_1 \in \{0,1\}^{\eta(\lambda)+1}$, computes $[c_1, d_1] \leftarrow \mathsf{Com}(\delta_1)$ and $[\hat{c_1}, \hat{d_1}] \leftarrow \mathsf{Com}(v_1)$ and broadcasts $(c_1, \hat{c_1})$. $\mathcal{F}$ receives $\{c_j, \hat{c_j}\}_{j\in[n], j\neq 1}$.

2. $\mathcal{F}$ broadcasts $(d_1, \hat{d_1})$ and receives $\{d_j, \hat{d_j}\}_{j\in[n], j\neq 1}$. For $i \in [n]$, let $\delta_i \leftarrow \mathsf{Reveal}(c_i, d_i)$ and $v_i \leftarrow \mathsf{Reveal}(\hat{c_i}, \hat{d_i})$.

3. $\mathcal{F}$ randomly selects $\delta_1', v_1' \in \{0,1\}^\lambda$, subject to the condition $\Delta = \mathsf{NextPrime}^*$ $(\delta_1' \oplus (\oplus_2^n \delta_i))$ and $v = \mathsf{NextPrime}(v_1' \oplus (\oplus_2^n v_i))$. Then $\mathcal{F}$ computes equivocated decommitment $(d_1', \hat{d_1'})$ which reveal δ_1', v_1', rewinds $\mathcal{A}$ to step 2 and broadcasts $(d_1', \hat{d_1'})$.

4. All parties compute the common output $\Delta = \mathsf{NextPrime}^*(\delta_1' \oplus (\oplus_2^n \delta_i))$ and $v = \mathsf{NextPrime}(v_1' \oplus (\oplus_2^n v_i))$.

5. $\mathcal{F}$ randomly selects $B_1 \in CL(\Delta)$ and computes $J_1 = B_1^{-v}$. Then $\mathcal{F}$ computes $[c_1^*, d_1^*] \leftarrow \mathsf{Com}(J_1)$ and broadcasts to all other parties the commitment c_1^*. $\mathcal{F}$ receives $\{c_j^*\}_{j\neq i}$.

6. $\mathcal{F}$ broadcasts d_1^* and performs a ZKPoKRoot for relation $\{(J_1, v) : B_1 : |J_1 = B_1^{-v}\}$. $\mathcal{F}$ then receives $\{d_j^*\}_{j\neq i}$. For $i \in [n]$, let $J_i \leftarrow \mathsf{Reveal}(c_i^*, d_i^*)$ be the opened commitment value of each party.

7. $\mathcal{F}$ rewinds $\mathcal{A}$ to step 6 and equivocates P_1's commitment to $d_1^{*'}$ so that the revealed value now is $J_1' = J(\prod_{i=2}^n J_i)^{-1}$ and broadcasts $d_1^{*'}$. Then $\mathcal{F}$ simulates ZKPoKRoot.

8. If all the proofs and commitments are correct the protocol continues with $J' = J_1' \prod_{i=2}^n J_i = J$.

Theorem 5.3. *If the commitment scheme is non-malleable and equivocal and ZKPoKRoot is honest verifier zero-knowledge proof of knowledge, then the* IKeyGen *simulation above is indistinguishable from a real execution in the view of potentially corrupted parties $P_2, P_3, \ldots, P_n$. Moreover, when the simulation does not abort, all parties output Δ, v in step 4 and J in step 8.*

Proof. In simulation, $\mathcal{F}$ does not know the δ_1 and v_1 chosen in real execution, but it chooses a δ_1' and v_1' such that $\Delta = \mathsf{NextPrime}^*(\delta_1' \oplus (\oplus_2^n \delta_i))$ and $v = \mathsf{NextPrime}(v_1' \oplus (\oplus_2^n v_i))$. Let $D = \oplus_2^n \delta_i$ and $V = \oplus_2^n v_i$. Let $S_\delta = \{x \in \{0,1\}^\lambda : \mathsf{PrevPrime}^*(\Delta) < x \oplus D < \Delta - 1\}$ be the set of all element x such that $\Delta = \mathsf{NextPrime}^*(x \oplus D)$ and $S_v = \{x \in \{0,1\}^\lambda : \mathsf{PrevPrime}(v) < x \oplus V < v - 1\}$ be the set of all element x such that $v = \mathsf{NextPrime}(x \oplus V)$. Since

δ_1 and v_1 belong to S_δ and S_v respectively and they are chosen uniformly at random, and δ_1' and v_1' are chosen uniformly at random in the same sets, simulation and real execution are indistinguishable in simulation setp 1-4. The only difference in step 5-8 is that $\mathcal{F}$ computes J_1' instead of using J_1. J_1 and $J(\prod_{i=2}^{n} J_i)^{-1}$ follow the same distribution. Hence, simulation and real execution are indistinguishable.

Moreover, the simulation may fail due to that someone may refuse to decommit after rewinding in step 2 and 7 and that some π_i fails. Since the commitment scheme is non-malleable and equivocal, in step 2 $\mathcal{F}$ can rewind and equivocate the commitment to δ_1 and v_1, and if there are not aborts, all parties decommit to their correct values. As a consequence, all parties output δ and v at the end of step 4. In step 7, all parties compute the correct J using δ and v from the deterministic setup of CL, if not there is an abort caused by the soundness of the proof π_i corresponding to the corrupted P_i. Finally, if no abort has occurred, $\mathcal{F}$ can equivocate the decommitment to J_1 and all parties decommit to the correct values thanks to the non-malleability of the scheme. If no party refuses to decommit after rewinding, the protocol ends with $J' = J_1' \prod_{i=2}^{n} J_i = J$.

□

5.5.0.2 Simulating P_1 in ISign Phase.

1. As in a real execution, $\mathcal{F}$ randomly selects $r_1 \in CL(\Delta)$ and computes $T_1 = r_1^v$. Then $\mathcal{F}$ computes $[c_1, d_1] \leftarrow \mathsf{Com}(T_1)$ and broadcasts to all other parties the commitment c_1. $\mathcal{F}$ receives $\{c_j\}_{j \neq i}$.

2. $\mathcal{F}$ broadcasts d_1 and performs a ZKPoKRoot for relation $\{(T_1, v) : r_1 : |T_1 = r_1^v\}$. $\mathcal{F}$ then receives $\{d_j\}_{j \neq i}$. For $i \in [n]$, let $T_i \leftarrow \mathsf{Reveal}(c_i, d_i)$ be the opened commitment value of each party.

3. $\mathcal{F}$ requests a signature (t, h) for a message M from its CL-GQ signing oracle and computes $T = t^v J^h$ (note that $h = H(M, T)$).

4. $\mathcal{F}$ rewinds $\mathcal{A}$ to step 2 and equivocates P_1's commitment to d_1' so that the revealed value now is $T_1' = T(\prod_{i=2}^{n} T_i)^{-1}$ and broadcasts d_1'. Then $\mathcal{F}$ simulates ZKPoKRoot.

5. If all the proofs and commitments are correct, all parties compute $T' = T_1' \prod_{i=2}^{n} T_i = T$, $h' = H(M, T) = h$. $\mathcal{F}$ computes $t_1 = r_1 B_1^{h'}$. and $[\hat{c}_1, \hat{d}_1] \leftarrow \mathsf{Com}(t_1)$. $\mathcal{F}$ broadcasts to all other parties the commitment $\hat{c}_1$. $\mathcal{F}$ receives $\{\hat{c}_j\}_{j \neq i}$.

6. $\mathcal{F}$ broadcasts $\hat{d}_1$ and performs a ZKPoKSig for relation $\{(T_1, J_1, t_1, h) : (r_1, B_1)|t_1 = r_1 B_1^h, T_1 = {r_1}^v, J_1 = {B_1}^{-v}\}$. $\mathcal{F}$ then receives $\{\hat{d}_j\}_{j\neq i}$. For $i \in [n]$, let $t_i \leftarrow \mathsf{Reveal}(\hat{c}_i, \hat{d}_i)$ be the opened commitment of each party.

7. $\mathcal{F}$ rewinds $\mathcal{A}$ to step 5 and equivocates P_1's commitment to $\hat{d'}_1$. The revealed value is $t'_1 = t(\prod_{i=2}^n t_i)^{-1}$ and broadcasts $\hat{d'}_1$. Then $\mathcal{F}$ simulates ZKPoKSig.

8. If all the proofs and commitments are correct, all parties compute $t' = t'_1 \prod_{i=2}^n t_i = t$ and output $\sigma = (t', h)$.

Theorem 5.4. *If the commitment scheme is non-malleable and equivocal and ZKPoKRoot and ZKPoKSig are honest verifier zero-knowledge proof of knowledge, then the* ISign *simulation above is indistinguishable from a real execution in the view of potentially corrupted parties $P_2, P_3, ..., P_n$ and on input M the simulation outputs a valid signature $\sigma = (t, h)$ or aborts.*

Proof. The only difference in this simulation is that $\mathcal{F}$ computes t'_1 and T'_1 instead of using t_1 and T_1. Since t_1 and $t(\prod_{i=2}^n t_i)^{-1}$ follow the same distribution; T_1 and $T(\prod_{i=2}^n T_i)^{-1}$ follow the same distribution. Hence, simulation and real execution are indistinguishable.

Let (t, h) be the signature that $\mathcal{F}$ receives from its signing oracle in step 5. This is a valid signature for message M. We prove that if the protocol terminates, it does so with output $(t' = t, h)$, which is due to the non-malleability property of commitment scheme. Indeed, the revealing should be the same with overwhelming probability if the adversary decommits correctly. □

Finally, we capture the security of our protocol by Theorem 5.5.

Theorem 5.5. *Assuming standard CL-GQ is an existentially unforgeable signature scheme; the ZKPoKRoot and ZKPoKSig are honest verifier zero-knowledge proof of knowledge; and the commitment scheme is non-malleable and equivocable, then our GQ multi-signature protocol (IKeyGen, ISign) is an existentially unforgeable multi-signature scheme.*

Proof. By Theorem 5.3 and 5.4, $\mathcal{F}$ always knows how to simulate $\mathcal{A}$'s view and all simulations are indistinguishable of real executions of the protocol. Moreover if A, having corrupted up to $n-1$ parties in the GQ multi-signining protocol, outputs a forgery, since $\mathcal{F}$ set up with $\mathcal{A}$ the same public key as it received from its' CL-GQ challenger, $\mathcal{F}$ can use this signature as its own forgery, thus breaking the existential unforgeability of standard CL-GQ.

Denoting $\mathsf{Adv}_{\Pi,\mathcal{A}}^{mu-cma}$, $\mathcal{A}$'s advantage in breaking the existential unforgeability of our multi-signature protocol, and $\mathsf{Adv}_{cl-gq,\mathcal{A}}^{euf-cma}$ the forger $\mathcal{F}$'s advantage in

breaking the existential unforgeability of standard CL-GQ, from Theorem 5.3 and 5.4 it holds that if ZKPoKRoot and ZKPoKSig are zero-knowledge and the commitment scheme is non-malleable and equivocable then $|\mathsf{Adv}^{mu-cma}_{\Pi,\mathcal{A}} - \mathsf{Adv}^{euf-cma}_{cl-gq,\mathcal{A}}|$ is negligible in λ. Under the security of the CL-GQ signature scheme proved in Theorem 1, $\mathsf{Adv}^{euf-cma}_{cl-gq,\mathcal{A}}$ is negligible, which implies that $\mathsf{Adv}^{mu-cma}_{\Pi,\mathcal{A}}$ is negligible as well, contradicting the assumption that $\mathcal{A}$ has non-negligible advantage of forging a signature for our protocol. Hence the theorem holds. □

5.6 Zero-knowledge Proofs

In this section, we give the detailed construction of ZKPoKRoot and ZKPoKSig which are used in our multi-signature protocol. At the first glance, both ZK proofs seem easy to construct. But one problem of ZK proofs in an unknown order group is that it requires that the challenge is a binary string and thus should be repeated for many rounds to achieve an acceptable soundness error, like the one-bit challenge ZK proofs in [22, 77]. We observe an interesting thing that the Bezout formula utilized in the EUF-CMA of CL-GQ can also be adopted when proving the special soundness of our ZK proofs, which accordingly waive the repetition of our protocol, the additional constraint is that the length of the challenge space should be smaller than $||v||$. This trick also answers the problem in Yi's blind ECDSA scheme [77], that how to speed up their ZK proof of Paillier ciphertext and in Appendix E we give a slightly modified version of the ZK proof they used, which waives any repetition.

5.6.1 Zero-knowledge Proof for the $-v$-th Root

We define a relation for the $-v$-th root of a class group element x where v is a prime:

$$\mathcal{R}_{\mathsf{root}} = \{(X, v) : x | X = x^{-v}\}.$$

We put forward a zero-knowledge proof of knowledge (ZKPoK) protocol named ZKPoKRoot (Table 5.3) which is needed in our multi-signature scheme. It should run for only one round to achieve a soundness error of $2^{-\gamma}$ where γ is the length of the challenge space we set in the ZKPoKRoot protocol, additionally required that $1 \leq \gamma \leq \eta(\lambda)$. x and X are class group elements and v is a prime.

Theorem 5.6. *The protocol ZKPoKRoot is an honest verifier zero-knowledge proof of knowledge with soundness error* $2^{-\gamma}$ *where* $1 \leq \gamma \leq \eta(\lambda)$.

TABLE 5.3: Zero-knowledge Proof ZKPoKRoot for relation $\mathcal{R}_{\mathsf{root}}$

ZKPoKRoot(X, v)		
P_i		$P_j (j \neq i)$
$r \xleftarrow{\$} CL(\Delta)$		
$t = r^v$	$\xrightarrow{t}$	
	$\xleftarrow{k}$	$k \xleftarrow{\$} \{0,1\}^\gamma$
$u = x^{-k} r$	$\xrightarrow{u}$	Check: $u^v = X^k t$

Proof. We prove completeness, special soundness and honest verifier zero-knowledge of our ZKPoKRoot protocol.

Completeness. The equation $X^k t = x^{-vk} r^v = (x^{-k} r)^v = u^v$ always holds for any $((X, v) : x) \in \mathcal{R}_{\mathsf{root}}$.

Special soundness. Assuming that the extractor $\mathcal{E}$ can send two challenges k and k' respectively to the prover for a same commitment t, it receives two responses u and u'. We have $u^v = X^k t$ and $u'^v = X^{k'} t$. We have $X^{k'-k} = (u/u')^v$. According to Bezout formula, there exists a unique pair of (α, β) where $0 \leq |\alpha| \leq |k' - k| - 1$ and $0 \leq |\beta| \leq v - 1$ which is easily computed by Euclidean algorithm s.t.:

$$\alpha v - \beta(k' - k) = gcd(v, k' - k) = 1$$

The rightmost equation holds since v is a prime much larger than either k or k'.

Raise equation $X^{k'-k} = (u/u')^v$ to power β, we extract the witness x by:

$$\begin{aligned} X^{\beta(k'-k)} &= (u/u')^{\beta v} \\ X^{\alpha v - 1} &= (u/u')^{\beta v} \\ X &= \{X^\alpha (u'/u)^\beta\}^v \\ x &= X^\alpha (u'/u)^\beta \end{aligned}$$

This has a soundness error of $1/2^\gamma$ for running one round. The γ is usually set to 40, 60, 80 for different soundness requirements, but in any case γ is smaller than the bit size of v.

Honest verifier zero-knowledge. Given $\mathcal{S}$ randomly chooses $\tilde{u} \in CL(\Delta), \tilde{k} \in \{0,1\}^\gamma$. the simulator $\mathcal{S}$ computes $\tilde{t} \leftarrow \tilde{u}^v / X^{\tilde{k}}$. Clearly, the distribution of t in a real execution is statistically close to $\tilde{t}$.

□

TABLE 5.4: Zero-knowledge Proof ZKPoKSig for relation $\mathcal{R}_{\mathsf{sig}}$

$\mathsf{ZKPoKSig}(T_i, J_i, t_i, h, v)$		
P_i		$P_j(j \neq i)$
$\rho_1, \rho_2 \xleftarrow{\$} CL(\Delta)$		
$\tau_1 = \rho_1^v$		
$\tau_2 = \rho_2^v$		
$\tau_3 = \rho_1^{-h}\rho_2$	$\xrightarrow{\tau_1,\tau_2,\tau_3}$	
	$\xleftarrow{k}$	$k \xleftarrow{\$} \{0,1\}^\gamma$
$u_1 = B_i^{-k}\rho_1$		
$u_2 = r_i^k\rho_2$	$\xrightarrow{u_1,u_2}$	Check: $u_1^v = J_i^k\tau_1$
		Check: $u_2^v = T_i^k\tau_2$
		Check: $u_1^{-h}u_2 = t_i^k\tau_3$

5.6.2 Zero-knowledge Proof of a CL-GQ Signature

We need another one-round ZKPoK protocol named ZKPoKSig (Table 5.4) for the following relation, where T_i, J_i, B_i are class group elements, h is a positive integer and v is a prime. We set γ as the challenge space which can be used to adjust the soundness error of ZKPoKSig, additionally required that $1 \leq \gamma \leq \eta(\lambda)$.

$$\mathcal{R}_{\mathsf{sig}} = \{(T_i, J_i, t_i, h, v) : (r_i, B_i) | t_i = r_i B_i^h, T_i = {r_i}^v, J_i = {B_i}^{-v}\}$$

Theorem 5.7. *The protocol ZKPoKSig is an honest verifier zero-knowledge proof of knowledge with soundness error* $2^{-\gamma}$ *where* $1 \leq \gamma \leq \eta(\lambda)$.

Proof. We prove completeness, special soundness and honest verifier zero-knowledge of ZKPoKSig protocol.

Completeness. For any $((T_i, J_i, t_i, h, v) : (r_i, B_i)) \in \mathcal{R}_{\mathsf{sig}}$. The following equations always hold:

$$\begin{aligned} J_i^k\tau_1 &= B_i^{-vk}{\rho_1}^v = (B_i^{-k}\rho_1)^v = u_1^v; \\ T_i^k\tau_2 &= {r_i}^{vk}{\rho_2}^v = (x^k\rho_2)^v = u_2^v; \\ t_i^k\tau_3 &= (r_i B_i^h)^k\rho_1^{-h}\rho_2 = (B_i^{-k}\rho_1)^{-h}r_i^k\rho_2 = u_1^{-h}u_2. \end{aligned}$$

Special soundness. Assuming that the extractor $\mathcal{E}$ can send two challenges k and k' respectively to the prover for a same commitment (τ_1, τ_2, τ_3), it receives two responses (u_1, u_2) and (u_1', u_2'). We have $u_1^v = J_i^k\tau_1, u_2^v = T_i^k\tau_2, u_1^{-h}u_2 =$

$t_i^k\tau_3$ and $u'^v_1 = J_i^{k'}\tau_1, u'^v_2 = T_i^{k'}\tau_2, u'^{-h}_1 u'_2 = t_i^{k'}\tau_3$. Observe the τ_1, τ_2, τ_3 are the same in both groups of equations, we have:

$$J_i^{k'-k} = (u'_1/u_1)^v = (u_1/u'_1)^{-v}; \quad (1)$$

$$T_i^{k'-k} = (u'_2/u_2)^v; \quad (2)$$

$$t_i^{k'-k} = [(u'_1/u_1)^{-h}(u'_2/u_2)] = (u_1/u'_1)^h(u'_2/u_2). \quad (3)$$

According to Bezout formula, there exists a unique pair of (α, β) where $0 \leq |\alpha| \leq |k'-k|-1$ and $0 \leq |\beta| \leq v-1$ and (W.L.O.G. assuming $k \geq k'$) which is easily computed by Euclidean algorithm s.t.:

$$\alpha v - \beta(k'-k) = gcd(v, k'-k) = 1$$

Raise equations (1) and (2) to power β:

$$\begin{aligned} J_i^{\beta(k'-k)} &= (u_1/u'_1)^{-\beta v} \\ J_i^{\alpha v-1} &= (u'_1/u_1)^{\beta v} \\ J_i &= \{J_i^{-\alpha}(u'_1/u_1)^{\beta}\}^{-v} \end{aligned} \quad (4)$$

$$\begin{aligned} T_i^{\beta(k'-k)} &= (u'_2/u_2)^{\beta v} \\ T_i^{\alpha v-1} &= (u'_2/u_2)^{\beta v} \\ T_i &= \{T_i^{\alpha}(u_2/u'_2)^{\beta}\}^{v} \end{aligned} \quad (5)$$

Apply the $u_1/u'_1 = J_i^{(k-k')/v}$ and $u'_2/u_2 = T_i^{(k'-k)/v}$ implied by (1) and (2) and the results of (4) and (5), we imply t_i by the following:

$$\begin{aligned} t_i^{(k'-k)} &= (u_1/u'_1)^h(u'_2/u_2) \\ t_i^{(k'-k)} &= (J_i^{(k-k')/v})^h T_i^{(k'-k)/v} \\ t_i &= J_i^{\frac{-h}{v}} T_i^{\frac{1}{v}} \\ t_i &= \{J_i^{-\alpha}(u'_1/u_1)^{\beta}\}^{-v\frac{-h}{v}}\{T_i^{\alpha}(u_2/u'_2)^{\beta}\}^{v\frac{1}{v}} \\ t_i &= T_i^{\alpha}(u_2/u'_2)^{\beta}\{J_i^{-\alpha}(u'_1/u_1)^{\beta}\}^h \end{aligned}$$

Hence, we extract the witness (r_i, B_i):

$$r_i = T_i^{\alpha}(u_2/u'_2)^{\beta}; \quad B_i = J_i^{-\alpha}(u'_1/u_1)^{\beta}.$$

The extraction has a soundness error of $1/2^{\gamma}$ for running one round.

Honest verifier zero-knowledge. Given $\tilde{u_1}, \tilde{u_2} \in CL(\Delta), \tilde{k} \in \{0,1\}^\gamma$, the simulator $\mathcal{S}$ computes $\tilde{\tau_1} \leftarrow \tilde{u_1}^v / J_i^{\tilde{k}}, \tilde{\tau_2} \leftarrow \tilde{u_2}^v / T_i^{\tilde{k}}, \tilde{\tau_3} \leftarrow (\tilde{u_1}^{-h}\tilde{u_2})/t_i^{\tilde{k}}$. Clearly, the distribution of (τ_1, τ_2, τ_3) in a real execution is statistically close to $(\tilde{\tau_1}, \tilde{\tau_2}, \tilde{\tau_3})$.

□

Remarks. To reduce the unnecessary interactions, we adopt Fiat-Shamir transformation [39] to make both ZKPoKRoot and ZKPoKSig non-interactive by replacing the challenge k in each ZKPoK with $H(t)$ and $H(\tau_1, \tau_2, \tau_3)$ respectively where H is a secure hash function.

5.6.3 Limitations of ZKPoK with LCM trick

In the threshold ECDSA protocol of [22, 23], they also involved zero-knowledge proofs in an unknown order group. The difference with our GQ is that the witness is in an exponent instead of in a base like in GQ setting. Without Bezout trick, our ZKPoK also cannot avoid a binary challenge problem which incurs a series of execution of ZK protocols to achieve suitable soundness error. For the ZK proofs in threshold ECDSA, it adopt a trivial binary challenge fashion in [22], but it is improved in [23] by using a special technique called LCM *(lowest common multiple)* trick. We observe that this trick can also be used in proving our ZK relations to avoid the binary challenge problem. We give the description for reader's reference.

Recall our ZK proofs without the additional requirement of challenge size is smaller than v where we cannot use Bezout trick, the challenge space size γ can only be set 1 bit to construct the successful extractor. Hence, ℓ repetitions of either ZKPoKRoot or ZKPoKSig are compulsory when we want to achieve a soundness $2^{-\ell}$ where ℓ is a positive integer. The massive running time undermines its practical application. To adopt this LCM trick, we need to modify the original ZKPoK protocols in two places: i) change the challenge space of k from $\{0,1\}$ to $\{0,1\}^C$ for some positive integer C and ii) change the repeat time from ℓ to ℓ/C. Through the revisited ZKPoK protocols, the relations, where $y = \mathsf{lcm}(1, 2, 3, ..., 2^C)$, are proved.

$$\mathcal{R}'_{\mathsf{root}} = \{x : X^z = (x^y)^v\}$$

$$\mathcal{R}'_{\mathsf{sig}} = \{(T_i, J_i, t_i, h, v) : (r_i, B_i) | t_i^z = r_i^y (B_i^y)^h, T_i^z = (r_i^y)^v, J_i^z = (B_i{}^y)^v\}$$

Caveats. One concern of such an LCM trick is that the modified relation is a loosed relation and thus it is questionable if we can initiate any potential attacks, more specifically, forge a witness which holds in the loosed relation but does not hold in the standard relation and this issue is not well discussed

in [23]. In the mean time, although it will execute less rounds of protocols, parameter C has to be carefully selected, since a larger C will lead to a higher exponentiation cost which can make the computational cost even worse.

5.7 Implementation and Evaluation

We implemented the original GQ signature, the CL-GQ signature, and our multi-party GQ signature without trusted setup in Rust language. We use the Rust library Class[2] to conduct the class group operations, including sampling, reduction, exponentiation and multiplication. It should be noted that this Rust library calls the C library Pari and thus it basically ensures the efficiency of the heavy arithmetic computations for class groups, but can still be improved. We benchmark the running times of both KeyGen and Sign for three schemes. All the programs are executed in a single thread on a MacBook Pro with Intel Core i5 1.4GHz and 16GB RAM.

5.7.1 Description of Security Level

To depict the security level of class group-based cryptographic system, Safuat and Bodo in [47] gave the expected number of MIPS (Million Instruction Per Second) years using GFNS algorithm [71] to factor large integer and using Cl-MPQS algorithm [53] to compute discrete logarithms in class groups should use, as shown in Table 5.5. They provided an algorithm to compute the order of a class group element, after which both roots and discrete logarithms can be computed by Pohlig-Hellman algorithm, thus showing the equivalence of hardnesses of computing roots and discrete logarithms in class groups. According to [3], the difficulties of factoring $|n| = 2^{1024}, 2^{2048}, 2^{3072}$ are equivalent to 80, 112 and 128 bits asymmetric security level. Thus we obtain that the difficulties of computing the discrete logarithms in class group under $|\Delta| = 2^{687}, 2^{1208}, 2^{1665}$ are respectively 80, 112 and 128 bits secure.

5.7.2 Details of Computing Bandwidth

For computing class group size, according to [47], we directly use $Cl = 2 \times \lceil \frac{\lambda-1}{2} \rceil + 1$ to represent the bit size of one class group element.

Recall that denote by $\eta(\lambda)$ the length of h under our security parameter λ, i.e., the bit size of discriminant Δ. More precisely, we have $\eta(1208) = 224$ and

[2]It is a library for building cryptography based on class groups of imaginary quadratic orders. https://github.com/ZenGo-X/class.

TABLE 5.5: Expected computational cost of factoring integers and computing discrete logarithms in class groups

n	Δ	Expected number of MIPS-years
768 bits	540 bits	4.99×10^7
1024 bits	687 bits	6.01×10^{10}
1536 bits	958 bits	5.95×10^{15}
2048 bits	1208 bits	7.05×10^{19}
3072 bits	1665 bits	2.65×10^{26}
4096 bits	2084 bits	5.87×10^{31}

TABLE 5.6: Comparison between GQ and CL-GQ in various security levels.

Sec. Level	GQ's $\lvert\sigma\rvert$	GQ KeyGen	GQ Sign	CL-GQ's $\lvert\sigma\rvert$	CL-GQ KeyGen	CL-GQ Sign
80-bit	1184 bits	30.375 ms	96.130 us	847 bits	221.77 ms	99.250 ms
112-bit	2272 bits	147.94 ms	472.44 us	1433 bits	2.0269 s	300.61 ms
128-bit	3328 bits	455.42 ms	1.1299 ms	1921 bits	6.9179 s	564.09 ms

$\eta(1665) = 256$ corresponding to the 112-bit and 128-bit security respectively. We use the rightmost $\eta(\lambda)$ bits of SHA256 as our hash function H and as the commitment of which the blind factor is also a $\eta(\lambda)$-bit binary string. For zero-knowledge proofs, we require a soundness error of 2^{-40}. Fiat-Shamir transformation is used to make our ZKPoKRoot and ZKPoKSig non-interactive. For bandwidth, each broadcast message or received message is counted as one transmission and we compute the total bandwidth for one participant. We set v as $\eta(\lambda)+1$ bits when running our protocol. The bit lengths of one ZKPoKRoot proof and one ZKPoKSig proof can be represented by $2 \times Cl$ bits and $5 \times Cl$ bits respectively, in the non-interactive setting where the challenge is computed by the verifier with the predefined hash function and the received commitments, and thus is not included in the proof size.

5.7.3 Standard GQ v.s. CL-GQ

We compare the standard GQ and CL-GQ in three security levels: 80-bit, 112-bit, 128-bit security, where 80-bit security is insecure and over 112-bit is generally deemed as secure. We set v as $\eta(\lambda)+1$ bits for both GQ and CL-GQ schemes. We compare the signature sizes, running times of both schemes. As observed from results in Table 5.6, removing the RSA trapdoor is obviously a trade-off of computational efficiency. CL-GQ is much slower for both KeyGen and Sign due to the complicated arithmetic operations for class group in CL-GQ. For signature size, our CL-GQ is much shorter than GQ. Details of computing bandwidth are given in Appendix D.

TABLE 5.7: Comparison with existing multiparty signing schemes. rds is the abbreviation of rounds; n denotes the number of sigining parties;each round allowing broadcasting and a point-to-point message sending is considered one round.

scheme	range proof	key aggregation	identifiable abort*	signing rounds
ECDSA (CCS 18) [42]	√	√	×	9
ECDSA (CCS 18) [55]	√	√	×	8
ECDSA (S&P 19) [36]	×	√	×	6+log(n)
ECDSA (PKC 20) [23]	×	√	×	8
ECDSA (PKC 21) [78]	×	√	×	8
ECDSA (G.G. 20) [43]	√	√	√	7
ECDSA (CCS 20) [20]	√	√	×	4
ECDSA (G.K.S.S. 20)[41]	×	√	(√)	13
Schnorr (CCS 06) [5]	×	×	×	3
Schnorr (DCC 19) [59]	×	√	×	3
Schnorr (CCS 20) [63]	×	√	×	2
Schnorr (N.R.S. 20) [62]	×	√	×	2
GQ (CT-RSA 06) [6]	×	-	×	3
GQ (This work)	×	√	√	4

5.7.4 Comparison with other multiparty signatures

We give a comparison between our proposed GQ multi-signature scheme and the above-mentioned multi-signature/threshold signature schemes in Table 5.8, in the aspects of security model, building blocks, key aggregation and signing rounds.

Here we consider each round of broadcasting or sending messages in a point-to-point fashion is one round. We observed that our proposed scheme are as secure as the series of ECDSA based schemes but keeps simplicity in the level of the number of signing rounds (4 rounds) close to the GQ and Schnorr based schemes. Note that Gagol et. al.'s scheme [41] achieves the identifiable abort only in the online signing phase, thus marked with (√) in the identifiable abort option in Table 5.8.

Also, in our proposed scheme, we achieve a stronger identifiable abort in a *timely manner*, which means the user can *instantly identify* (within the same communication round) the incorrect messages sent from the malicious parties (we denote this stronger property with *indentifiable abort** in Table 5.6) and avoid continuous useless computation.

TABLE 5.8: Benchmarks of trustless GQ multi-signature.

Sec. Level	# Party	Comp. IKeyGen	Comp. ISign	Comm. IKeyGen	Comm. ISign
112-bit security	2	10.908 s	3.139 s	1848 Bytes	2945 Bytes
	3	15.006 s	5.253 s	2771 Bytes	4417 Bytes
	4	19.947 s	7.663 s	3695 Bytes	5889 Bytes
	5	35.295 s	10.505 s	4619 Bytes	7361 Bytes
128-bit security	2	29.206 s	5.569 s	2466 Bytes	4003 Bytes
	3	36.594 s	9.298 s	3698 Bytes	6004 Bytes
	4	40.168 s	13.372 s	4931 Bytes	8005 Bytes
	5	47.825 s	17.991 s	6164 Bytes	10006 Bytes

5.7.5 Performance

We evaluate the running time and bandwidth of multi-party GQ without trusted setup. The running time is obtained from the median running time among 20 test samples each of which *sequentially executes* the computation of each signer (in fact the protocol can be executed in parallel but here we consider achieving a fair comparison). In a 5-user setting without considering the network constraint, each signer only needs around 2.1 and 3.6 seconds to sign a message in 112-bit and 128-bit security levels respectively. We computed the concrete Bytes needed for multi-party GQ in 112-bit and 128-bit asymmetric security levels, and gave the calculation formula (Notice that in the given formula λ means the length of Δ, instead of a security level 112 or 128). The details of computing the bandwidth are given in Appendix D. Both bandwidth and running time confirm that our trapdoorless GQ multi-signature is very practical in use. Our bandwidth is only about one-thirds of the bandwidth of joint signing in [42].

$$Comm.cost(\mathsf{IKeyGen}) = n \times \{10 \times \lceil \frac{\lambda - 1}{2} \rceil + 6 \times \eta(\lambda) + 5\} \quad (bits)$$

$$Comm.cost(\mathsf{ISign}) = n \times \{18 \times \lceil \frac{\lambda - 1}{2} \rceil + 4 \times \eta(\lambda) + 9\} \quad (bits)$$

Impacts from the number of users. Consider an N-party setting, since we assume the existence of broadcast channel, each party only computes their commitments and NIZK proofs once, and thus N computations in total are needed. On the receiver's side, however, each party should de-commit the commitments and verify the NIZK proofs received from all other parties, and thus $N(N-1)$ computations in total are needed. The accumulations of $\delta_i, v_i, J_i, T_i, t_i$ are also in $\mathcal{O}(N^2)$ complexity. Hence, the computational burden increases in a non-linear way when participants increase. Besides, as the

increasing of the size of Δ and v, the uncertainty of computing $\mathsf{NextPrime}^*$ and $\mathsf{NextPrime}$ is larger, leading to a noticeable variance of running time of IKeyGen. For example, in our 20 test samples of 5-user experiment, the longest running time is up to 70 seconds. On the other hand, the variance of the running time of ISign is trivial.

5.8 Optimization of the ZK Proof in Yi's Blind ECDSA

5.8.1 Zero-knowledge proof for well-formedness of Paillier ciphertext

A modified Paillier encryption scheme is proposed in [77], where the modulo N is set pqt (p, q, t are primes) and g is set $(1+N)^{pt}$, different from the original Paillier. By this modification, the message space becomes q instead of N and thus the message space can be set the same of the key space of an EC group of order q. The ciphertext is the same as a standard Paillier, which is $C = g^m r^N \mod N^2$. We review the ZK proof which ensures the correctness of the ciphertext encrypted from a modified paillier encryption scheme proposed in [77], where (N, g) is the public key, C is a ciphertext, (m, r) is the witness. The relation and the ZK protocol is described as follows:

$$R = \{(N, g, C) : (m, r) | C = g^m r^N \mod N^2\}$$

1. Prover randomly chooses $m' \in \mathbb{Z}_q$ and $r' \in \mathbb{Z}^*_{N^2}$ and computes $C' = g^{m'} r'^N \mod N^2$ and sends C' to the verifier;
2. Verifier randomly chooses $c \in \{0, 1\}$ and sends c to prover.
3. Prover computes $u = m' + cm \mod q, v = r^c r' \mod N^2$ and sends (u, v) to verifier.
4. Verifier outputs 1 if $C^c C' = g^u v^N \mod N^2$; outputs 0 otherwise.

Necessity of repeatition. This is a classic Σ protocol, including three phases of commit, challenge and response. To achieve a soundness error 2^ℓ, this ZK should be repeated by ℓ times. We demenstrate the necessity of such a repetition here: when proving the special soundness, the extractor can manipulate the random tape of the verifier. After a commitment of randomness r' by prover and a challenge c_1 by verifier, extractor rewinds the protocol to commitment phase again where the prover still uses the same randomness r but

the verifier uses another challenge c_2. Now we obtain two equations: $v_1 = r^{c_1}r'$ and $v_2 = r^{c_2}r'$. Then we have $r^{c_1-c_2} = \frac{v_1}{v_2} \mod N^2$. If $c_{i=1,2} \in \{0,1\}^k$ and k is an integer larger than 1, we have to solve the $(c_1-c_2)-th$ root of $\frac{v_1}{v_2} \mod N^2$. It is intractable to compute such a root without knowing the factorazation of N according to the DCRA assumption. Thus, $c_{i=1,2}$ has to be sampled from $\{0,1\}$ and ℓ repeatitions are required to achieve soundness error $2^{-\ell}$.

5.8.2 ZK waiving repeatition

We change the modified paillier in [77] a little bit by requiring that an $R = r^N \mod N^2$ should be published along with the ciphertext $C = g^m r^N \mod N$. In another word, we adjust the original ciphertext for a message m to a tuple (C, R) where $C = g^m r^N \mod N$ and $R = r^N \mod N^2$). The modified relation $\mathcal{R}^*$ and corresponding ZK as follows.

$$\mathcal{R}^* = \{(N, g, C, R) : (m, r) | C = g^m r^N, R = r^N \mod N^2\}$$

1. Prover randomly chooses $m' \in \mathbb{Z}_q$ and $r' \in \mathbb{Z}^*_{N^2}$ and computes $C' = g^{m'} r'^N \mod N^2, R' = r'^N \mod N^2$;
2. Verifier randomly chooses $c \in \{0,1\}^k$ and sends c to prover.
3. Prover computes $u = m' + cm \mod q, v = r^c r' \mod N^2$ and sends (u, v) to verifier.
4. Verifier outputs 1 if $C^c C' = g^u v^N \mod N^2$ and $R^c R' = v^N \mod N^2$; outputs 0 otherwise.

Theorem 5.8. *The ZK proof for for relation $\mathcal{R}^*$ with challenge c is sampled from $\{0,1\}^k$ where $1 < k < min\{|p|, |q|, |t|\}$ has special soundness with soundness error of $\frac{1}{2^k}$ when repeated for one round.*

Proof. By rewinding, we obtain

$$C^{c_1} C' = g^{u_1} {v_1}^N, C^{c_2} C' = g^{u_2} {v_2}^N \mod N^2$$
$$R^{c_1} R' = v_1^N, \quad R^{c_2} R' = v_2^N \mod N^2$$

After deviding

$$R^{c_1-c_2} = (\frac{v_1}{v_2})^N \mod N^2$$
$$C^{c_1-c_2} = g^{u_1-u_2 \mod q} (\frac{v_1}{v_2})^N \mod N^2$$

and accordingly

$$(CR^{-1})^{c_1-c_2} = g^{u_1-u_2} \mod q$$
$$CR^{-1} = g^{(u_1-u_2)(c_1-c_2)^{-1}} \mod q$$

Since $(c_1 - c_2) < min\{p, q, t\}$, then $gcd(N, c_1 - c_2) = 1$.

According to Bezout formula, there exists a unique pair of (α, β) where $0 \leq \alpha \leq c_1 - c_2 - 1$ and $0 \leq \beta \leq N - 1$ (W.L.O.G. assuming $c_2 \geq c_1$) which is easily computed from Enclidean algorithm s.t.:

$$\alpha N - \beta(c_1 - c_2) = gcd(N, c_1 - c_2) = 1.$$

Raise equation $R^{c_1-c_2} = (\frac{v_1}{v_2})^N \mod N^2$ to power β, we have:

$$R^{\beta(c_1-c_2)} = (\frac{v_1}{v_2})^{\beta N} \mod N^2$$
$$R^{\alpha N-1} = (\frac{v_1}{v_2})^{\beta N} \mod N^2$$
$$R = \{R^{\alpha}(\frac{v_2}{v_1})^{\beta}\}^N \mod N^2$$

Raise equation $C^{c_1-c_2} = g^{u_1-u_2}(\frac{v_1}{v_2})^N \mod N^2$ to power β, we have:

$$C^{\beta(c_1-c_2)} = g^{\beta(u_1-u_2)} \mod q (\frac{v_1}{v_2})^{\beta N} \mod N^2$$
$$C^{\alpha N-1} = g^{\beta(u_1-u_2)} \mod q (\frac{v_1}{v_2})^{\beta N} \mod N^2$$
$$C = C^{\alpha N} g^{-\beta(u_1-u_2)} \mod q (\frac{v_2}{v_1})^{\beta N} \mod N^2$$
$$C = (CR^{-1})^{\alpha N} g^{-\beta(u_1-u_2)} \mod q \{R^{\alpha}(\frac{v_2}{v_1})^{\beta}\}^N \mod N^2$$
$$C = g^{\alpha N(u_1-u_2)(c_1-c_2)^{-1}-\beta(u_1-u_2)} \mod q \{R^{\alpha}(\frac{v_2}{v_1})^{\beta}\}^N \mod N^2$$
$$C = g^{(u_1-u_2)(c_1-c_2)^{-1}(\alpha N-\beta(c_1-c_2))} \mod q \{R^{\alpha}(\frac{v_2}{v_1})^{\beta}\}^N \mod N^2$$
$$C = g^{(u_1-u_2)(c_1-c_2)^{-1}} \mod q \{R^{\alpha}(\frac{v_2}{v_1})^{\beta}\}^N \mod N^2$$

□

Now we extract the witnesses $r = R^{\alpha}(\frac{v_2}{v_1})^{\beta}$ and $m = (u_1-u_2)(c_1-c_2)^{-1} \mod q$. As such, with only a single round, the Zero-Knowledge (ZK) proof for Paillier ciphertext can achieve the requisite soundness. This is accomplished with an

additional requirement that the length of the challenge space is smaller than $min\{|p|, |q|, |t|\}$. This requirement is easy to fulfill, and it greatly enhances the efficiency of their blind ECDSA scheme, which is useful for anonymous Bitcoin transactions.

5.9 Conclusion

In this research, we start by formalizing the class group-based GQ signature. We then propose a trapdoorless GQ multi-signature scheme with an identifiable abort property. This scheme involves only four rounds of interaction during the signing phase and is secure in the dishonest majority model.

Our concise security proof eliminates the need for the simulator to detect a non semi-correct execution. We also include two compact one-round Non-Interactive Zero-Knowledge proofs (NIZKs), which remove the repetitions caused by the binary challenge.

We provide a thorough implementation and efficiency analysis, demonstrating that our scheme boasts impressive running times and exceptional bandwidth efficiency.

Chapter 6

Conclusion and Outlook

Trustless in cryptography means that there is no trusted third party for a cryptosystem, which satisfies the needs of a decentralized world like public blockchain. Schnorr and ECDSA are both trustless signature since they system parameters are simply cyclic groups without trapdoor which can be developed or chosen by the system developer. On the contrary RSA and GQ signatures are not trustless signatures since their system parameter contains trapdoor and hence cannot be developed by non-trusted party. ECDSA is widely used since 2009, the birth year of blockchain. Schnorr is a more efficient and secure signature scheme but it was not adopted at that time due to the unexpired patent protection. But research and exploration of upgrading ECDSA to Schnorr in blockchain never cease.

In the first work, we analyze the real-world scenario of address-based signature, which captures that the signature verification in blockchain and XIA is actually accomplished against a hash of the public key instead of a complete public key. This technique will bring good compatibility in XIA infrastructure and lower fees for blockchain transactions. For the first time, we formalize the notion of address-based signature and give two generic constructions. We additionally propose a novel address-based signature scheme which outperforms address-based Schnorr and ECDSA and other different instantiations of generic constructions. For the future work, it is interesting to see how to extend our proposed novel digital signature to exotic signature schemes like threshold signature, ring signature, adaptor signature, blind signature and etc., and see if there are any new properties it can bring along compared to ECDSA and Schnorr counterparts. For the limitation, since it is a quite new alternative digital scheme, it has a long distance to be further investigated by researchers for its security properties and feasibility to be included in any future blockchain systems.

In the second work, we focus on the trustless multiparty threshold ECDSA. Since a large amount of ZK proofs have to be used in threshold ECDSA, optimizing ZK proof becomes a promising breakthrough point to optimize threshold ECDSA. Then we optimize the existing ZK proofs for threshold ECDSA built on top of CL-encryption, a bandwidth-efficient additively homomorphic encryption scheme, and achieve a bandwidth-optimal UC-secure threshold ECDSA at a cost of larger computation power using our delicately designed ZK proofs. It gives one more choice when implementing the threshold ECDSA, given that the bandwidth is more important than the computational cost in blockchain. We leave optimizing the tedious pre-signing as future work. Additionally, except for the proposed CL variant of the UC threshold ECDSA, our ZK proofs can also derive two threshold ECDSA schemes based on scheme 1 and 2 from [78]. Those three schemes can all integrate with existing ECDSA supported blockchain to build up MPC wallet.

In the third work, we focus on GQ signature, which is an old and classical three-move signature scheme rather similar to Schnorr. It was almost abandoned to use due to that it is not trustless and it is of large storage burden. But we give some new explorations. We remake the GQ possible in a trustless world, namely, eliminate the trapdoor of its setup parameters, via the class group. Then we construct compact ZK proofs for its commitment and signature to achieve a multi-signature scheme with identifiable abort property, in which sense we assign the new GQ multi-signature the possibility to build up a digital wallet or an asset custody solution in a highly malicious environment. For future work, it is interesting to see how we convert the current class group based GQ multi-signature to a threshold version, make it support periodical key refreshment and adaptive corruption.

The latter two works are both related to class group and the major limitations are the low computational efficiency of class group cryptography. For the discriminant size, we adopt 1665 bit length in the third work as referenced from [47] which is an older standard, and adopt 1827 bit length in our second work as referenced in [8] which is consistent with most recent class group based ECDSA schemes in [22–24, 33]. However, choosing the bit size of the discriminant for ideal class group is still controversial. One good news is that class group representation can be further compressed from $\log_2(|\Delta|)$ bits to $\frac{3}{4}\log_2(|\Delta|)$ using the technique in [35]. But as claimed in [35], it is suggested to use a larger bit size than 1827 to provide better security in a 128-bit security level, which will be a burden for all class group based cryptosystems. Nevertheless, in a very recent work [14], BICYCL, a new implementation for cryptosystems based on class group, was proposed as the fastest implementation than any previous implementation of the CL encryption scheme. But it still adopts the 1827-bit for the class group discriminant size for 128-bit security. BICYCL achieves faster speed for CL encryption than Paillier encryption at any security level, which becomes the state-of-the-art linearly homomorphic encryption

solution for both bandwidth and computation aspects. It is thrilling news for all current class group based cryptosystems.

Bibliography

[1] Ashok Anand, Fahad Dogar, Dongsu Han, Boyan Li, Hyeontaek Lim, Michel Machado, Wenfei Wu, Aditya Akella, David G. Andersen, John W. Byers, Srinivasan Seshan, and Peter Steenkiste. Xia: An architecture for an evolvable and trustworthy internet. In *HotNets-X*, pages 2:1–2:6. ACM, 2011.

[2] David G. Andersen, Hari Balakrishnan, Nick Feamster, Teemu Koponen, Daekyeong Moon, and Scott Shenker. Accountable internet protocol (aip). In Victor Bahl, David Wetherall, Stefan Savage, and Ion Stoica, editors, *ACM SIGCOMM 2008*, pages 339–350. ACM, 2008.

[3] Elaine Barker, William Burr, Alicia Jones, Timothy Polk, Scott Rose, Miles Smid, and Quynh Dang. Recommendation for key management part 3: Application-specific key management guidance. *NIST special publication*, 800:57, 2009.

[4] Mihir Bellare, David Cash, and Rachel Miller. Cryptography secure against related-key attacks and tampering. In *ASIACRYPT 2011*, pages 486–503. Springer, 2011.

[5] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM conference on Computer and communications security - ACM CCS 2006*, pages 390–399, 2006.

[6] Mihir Bellare and Gregory Neven. Identity-based multi-signatures from rsa. In *Cryptographers' Track at the RSA Conference - CT-RSA 2007*, pages 145–162. Springer, 2007.

[7] Mihir Bellare and Adriana Palacio. Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *Annual International Cryptology Conference*, pages 162–177. Springer, 2002.

[8] Jean-François Biasse, Michael J Jacobson Jr, and Alan K Silvester. Security estimates for quadratic field based cryptosystems. In *Australasian Conference on Information Security and Privacy*, pages 233–247. Springer, 2010.

[9] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *International Workshop on Public Key Cryptography - PKC 2003*, pages 31–46. Springer, 2003.

[10] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004.

[11] Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to iops and stateless blockchains. In *Annual International Cryptology Conference - CRYPTO 2019*, pages 561–586. Springer, 2019.

[12] Dan Boneh and Matthew Franklin. Efficient generation of shared rsa keys. In *Annual international cryptology conference - CRYPTO '97*, pages 425–439. Springer, 1997.

[13] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.

[14] Cyril Bouvier, Guilhem Castagnos, Laurent Imbert, and Fabien Laguillaumie. I want to ride my bicycl: Bicycl implements cryptography in class groups. *Journal of Cryptology*, 36(3):17, 2023.

[15] Daniel R. L. Brown. Generic groups, collision resistance, and ECDSA. *Des. Codes Cryptogr.*, 35(1):119–152, 2005.

[16] Daniel R. L. Brown. On the provable security of ECDSA. In Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart, editors, *Advances in Elliptic Curve Cryptography*, London Mathematical Society Lecture Note Series, page 21–40. Cambridge University Press, 2005.

[17] Daniel R. L. Brown. Standards for efficient cryptography. sec 1: Elliptic curve cryptography, 2009.

[18] Johannes A Buchmann and Hugh C Williams. A key exchange system based on real quadratic fields extended abstract. In *Conference on the Theory and Application of Cryptology - CRYPTO '89*, pages 335–343. Springer, 1989.

[19] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of FOCS '01*, pages 136–145, Las Vegas, Nevada, USA, 8-11 October 2001. IEEE.

[20] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. Uc non-interactive, proactive, threshold ecdsa with identifiable aborts. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security - ACM CCS '20'*, pages 1769–1787, 2020.

[21] Ran Canetti, Nikolaos Makriyannis, and Udi Peled. UC non-interactive, proactive, threshold ECDSA. Cryptology ePrint Archive, Report 2020/492, 2020.

[22] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Two-party ecdsa from hash proof systems and efficient instantiations. In *Annual International Cryptology Conference - CRYPTO 2019*, pages 191–221. Springer, 2019.

[23] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold ec-dsa. In *IACR International Conference on Public-Key Cryptography - PKC 2020*, pages 266–296. Springer, 2020.

[24] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold ec-dsa revisited: Online/offline extensions, identifiable aborts proactive and adaptive security. *Theoretical Computer Science*, 939:78–104, 2023.

[25] Guilhem Castagnos and Fabien Laguillaumie. On the security of cryptosystems with quadratic decryption: The nicest cryptanalysis. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 260–277. Springer, 2009.

[26] Guilhem Castagnos and Fabien Laguillaumie. Linearly homomorphic encryption from DDH. In Kaisa Nyberg, editor, *CT-RSA 2015*, volume 9048 of *Lecture Notes in Computer Science*, pages 487–505. Springer, 2015.

[27] Cheng-Kang Chu and Wen-Guey Tzeng. Optimal resilient threshold gq signatures. *Information sciences*, 177(8):1834–1851, 2007.

[28] Ivan Damgård and Maciej Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In *International Conference on the Theory and Applications of Cryptographic Techniques - EUROCRYPT 2002*, pages 256–271. Springer, 2002.

[29] Ivan Damgård and Gert Læssøe Mikkelsen. Efficient, robust and constant-round distributed rsa key generation. In *Theory of Cryptography Conference*, pages 183–200. Springer, 2010.

[30] Christian Decker and Roger Wattenhofer. Bitcoin transaction malleability and mtgox. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *ESORICS 2014*, volume 8713 of *Lecture Notes in Computer Science*, pages 313–326. Springer, 2014.

[31] Olivier Delos and Jean-Jacques Quisquater. Efficient multi-signature schemes for cooperating entities. In *Workshop on Algebraic Coding*, pages 63–74. Springer, 1993.

[32] Olivier Delos and Jean-Jacques Quisquater. An identity-based signature scheme with bounded life-span. In *Annual International Cryptology Conference - CRYPTO '94*, pages 83–94. Springer, 1994.

[33] Yi Deng, Shunli Ma, Xinxuan Zhang, Hailong Wang, Xuyang Song, and Xiang Xie. Promise sigma-protocol: How to construct efficient threshold ecdsa from encryptions based on class groups. In *ASIACRYPT 2021*, pages 557–586. Springer, 2021.

[34] Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. Non-interactive and non-malleable commitment. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC '98*, pages 141–150, 1998.

[35] Samuel Dobson, Steven Galbraith, and Benjamin Smith. Trustless unknown-order groups. *arXiv preprint arXiv:2211.16128*, 2022.

[36] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. Threshold ecdsa from ecdsa assumptions: the multiparty case. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1051–1066. IEEE, 2019.

[37] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the fiat-shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *Lecture Notes in Computer Science*, pages 60–79. Springer, 2012.

[38] Manuel Fersch, Eike Kiltz, and Bertram Poettering. On the provable security of (EC)DSA signatures. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *CCS 2016*, pages 1651–1662. ACM, 2016.

[39] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and*

application of cryptographic techniques - CRYPTO '86, pages 186–194. Springer, 1986.

[40] Yair Frankel, Philip D MacKenzie, and Moti Yung. Robust efficient distributed rsa-key generation. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 663–672, 1998.

[41] Adam Gagol and Damian Straszak. Threshold ecdsa for decentralized asset custody. Technical report, Cryptology ePrint Archive, Report 2020/498, 2020., 2020.

[42] Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ecdsa with fast trustless setup. In *ACM Conference on Computer and Communications Security - ACM CCS 2018*, 2018.

[43] Rosario Gennaro and Steven Goldfeder. One round threshold ecdsa with identifiable abort. *IACR Cryptol. ePrint Arch.*, 2020:540, 2020.

[44] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold dss signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques - EUROCRYPT '96'*, pages 354–371. Springer, 1996.

[45] Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78, 1998.

[46] Louis Claude Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *Conference on the Theory and Application of Cryptography - CRYPTO '88*, pages 216–231. Springer, 1988.

[47] Safuat Hamdy and Bodo Möller. Security of cryptosystems based on class groups of imaginary quadratic orders. In *International Conference on the Theory and Application of Cryptology and Information Security - ASIACRYPT 2000*, pages 234–247. Springer, 2000.

[48] L-K Hua. *Introduction to number theory*. Springer Science & Business Media, 2012.

[49] Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In *Annual Cryptology Conference*, pages 369–386. Springer, 2014.

[50] I.S.I. Information technology—security techniques— digital signatures with appendix—part 2: Integer factorization based mechanisms. *ISO/IEC*, 14888-2(2008), 1999.

[51] K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, 71:1–8, 1983.

[52] Gene Itkis and Leonid Reyzin. Forward-secure signatures with optimal signing and verifying. In *Annual International Cryptology Conference - CRYPTO 2001*, pages 332–354. Springer, 2001.

[53] Michael J Jacobson. *Subexponential Class Group Computation in Quadratic Orders*. PhD thesis, Technische Universit at Darmstadt, Fachbereich Informatik, Darmstadt, Germany, 1999.

[54] Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016*, volume 9815 of *Lecture Notes in Computer Science*, pages 33–61. Springer, 2016.

[55] Yehuda Lindell and Ariel Nof. Fast secure multiparty ecdsa with practical distributed key generation and applications to cryptocurrency custody. In *ACM Conference on Computer and Communications Security - ACM CCS 2018*, 2018.

[56] Yehuda Lindell and Ariel Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In *CCS 2018*, pages 1837–1854, 2018.

[57] Li-Shan Liu, Cheng-Kang Chu, and Wen-Guey Tzeng. A threshold gq signature scheme. In *International Conference on Applied Cryptography and Network Security - ACNS 2003*, pages 137–150. Springer, 2003.

[58] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques - EUROCRYPT 2006*, pages 465–485. Springer, 2006.

[59] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. *Designs, Codes and Cryptography*, 87(9):2139–2164, 2019.

[60] Hiraku Morita, Jacob CN Schuldt, Takahiro Matsuda, Goichiro Hanaoka, and Tetsu Iwata. On the security of the schnorr signature scheme and dsa against related-key attacks. In *ICISC 2015*, pages 20–35. Springer, 2015.

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

[62] Jonas Nick, Tim Ruffing, and Yannick Seurin. Musig2: Simple two-round schnorr multi-signatures. Technical report, Cryptology ePrint Archive, Report 2020/1261, 2020. https://eprint. iacr. org . . . , 2020.

[63] Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. Musig-dn: Schnorr multi-signatures with verifiably deterministic nonces. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1717–1731, 2020.

[64] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, page 223–238. Springer-Verlag, 1999.

[65] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal K. Roy and Willi Meier, editors, *FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2004.

[66] Ahmad-Reza Sadeghi and Michael Steiner. Assumptions related to discrete logarithms: Why subtleties make a real difference. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 244–261. Springer, 2001.

[67] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.

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

[69] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[70] Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel P. Smart. Flaws in applying proof methodologies to signature schemes. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 93–110. Springer, 2002.

[71] Herman JJ te Riele et al. Factorization of a 512-bits rsa key using the number field sieve. *Announcment on the Number Theory List (NMBRTHRY@ listserv. nodak. edu)*, 1999.

[72] Pei-yih Ting, Dream-Ming Huang, and Xiao-Wei Huang. A proxy multisignature scheme with anonymous vetoable delegation. *International Journal of Computer and Information Engineering*, 3(5):1387–1392, 2009.

[73] A. Vanstone. Responses to NIST's proposa. *Communications of the ACM*, 35:50–52, 1992.

[74] Hong Wang, Zhen-Feng Zhang, and Deng-Guo Feng. Robust threshold guillou-quisquater signature scheme. *Wuhan University Journal of Natural Sciences*, 10(1):207–210, 2005.

[75] Pieter Wuille. bip-schnorr, 2018. https://github.com/sipa/bips/blob/bip-schnorr/bip-schnorr.mediawiki.

[76] Jun Yao and Gui-Hua Zeng. A distributed authentication algorithm based on gq signature for mobile ad hoc networks. *Journal of Shanghai Jiaotong University (Science)*, 11(3):346–350, 2006.

[77] Xun Yi and Kwok-Yan Lam. A new blind ecdsa scheme for bitcoin transaction anonymity. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security - AsiaCCS '19*, pages 613–620, 2019.

[78] Tsz Hon Yuen, Handong Cui, and Xiang Xie. Compact zero-knowledge proofs for threshold ecdsa with trustless setup. In *Public-Key Cryptography – PKC 2021*, pages 481–511. Springer, 2021.

[79] Tsz Hon Yuen and Siu-Ming Yiu. Strong known related-key attacks and the security of ecdsa. In *NSS 2019*, pages 130–145. Springer, 2019.

[80] Dae Hyun Yum and Pil Joong Lee. A distributed online certificate status protocol based on gq signature scheme. In *International Conference on Computational Science and Its Applications- ICCSA 2004*, pages 471–480. Springer, 2004.