VERIFIABLE TIMED COMMITMENTS EXPLORED: TIMED SIGNATURE SCHEMES AND AN APPLICATION TO SEALED-BID AUCTIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DUYGU ÖZDEN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
CRYPTOGRAPHY

JULY 2023

Approval of the thesis:

VERIFIABLE TIMED COMMITMENTS EXPLORED: TIMED SIGNATURE SCHEMES AND AN APPLICATION TO SEALED-BID AUCTIONS

submitted by **DUYGU ÖZDEN** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Cryptography Department, Middle East Technical University** by,

Prof. Dr. A. Sevtap Selçuk Kestel _____
Dean, Graduate School of **Applied Mathematics**

Assoc. Prof. Dr. Oğuz Yayla _____
Head of Department, **Cryptography**

Assoc. Prof. Dr. Oğuz Yayla _____
Supervisor, **Cryptography, METU**

Examining Committee Members:

Prof. Dr. Murat Cenk
Cryptography, METU _____

Assoc. Prof. Dr. Oğuz Yayla
Cryptography, METU _____

Prof. Dr. Ali Aydın Selçuk
Computer Engineering, TOBB ETU _____

Assoc. Prof. Dr. Fatih Sulak
Mathematics, Atılım University _____

Assist. Prof. Dr. Talha Arıkan
Mathematics, Hacettepe University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: DUYGU ÖZDEN

Signature :

ABSTRACT

VERIFIABLE TIMED COMMITMENTS EXPLORED: TIMED SIGNATURE SCHEMES AND AN APPLICATION TO SEALED-BID AUCTIONS

Özden, Duygu

Ph.D., Department of Cryptography

Supervisor : Assoc. Prof. Dr. Oğuz Yayla

July 2023, 73 pages

Timed commitments are cryptographic primitives that allow one party to commit to a value for a specific duration of time while providing proof of the committed value's existence and integrity. This thesis explores the concept of verifiable timed commitments and investigates their applications in timed signature schemes and sealed-bid auctions. We discuss the security requirements, construction techniques, and computational complexity of the proposed signature schemes. Furthermore, we examine the application of verifiable timed commitments in sealed-bid auctions, which are widely used in sensitive bidding environments. Sealed-bid auctions require bidders to submit their bids privately, without knowledge of other participants' bids, and simultaneously reveal them at a predetermined time. We explore how timed commitments can ensure bid confidentiality, integrity, and fairness in sealed-bid auctions. We discuss the protocols for commitment generation, commitment opening, and bid verification, along with the necessary security considerations. The research employs a multi-disciplinary approach, including theoretical analysis, algorithm design, and applications, to assess the security of the proposed schemes based on verifiable timed commitments.

Keywords: verifiable timed commitment, sealed-bid auction, timed signature, fairness

ÖZ

DOĞRULANABİLİR ZAMANLI TAAHHÜTLERİN İNCELENMESİ: ZAMANLI İMZA ŞEMALARI VE KAPALI TEKLİF MÜZAYEDELERİNE UYGULANMASI

Özden, Duygu
Doktora, Kriptografi Bölümü
Tez Yöneticisi : Doç. Dr. Oğuz Yayla

Temmuz 2023, 73 sayfa

Zamanlanmış taahhütler, bir tarafın bir değeri belirli bir süre için taahhüt etmesine ve o değerin varlığını ve bütünlüğünü kanıtlamasına olanak tanıyan protokollerdir. Bu tez, doğrulanabilir zamanlanmış taahhüt kavramını keşfeder ve zamanlanmış imza şemaları ve kapalı teklif müzayedeleri üzerindeki uygulamalarını araştırır. Çalışma, önerilen imza şemalarının güvenlik gereksinimlerini, yapım tekniklerini ve hesaplama karmaşıklığını ortaya koymaktadır. Ayrıca, hassas teklif verme ortamlarında yaygın olarak kullanılan kapalı teklif müzayedelerinde doğrulanabilir zamanlanmış taahhütlerin uygulamaları incelenmektedir. Kapalı teklif müzayedeleri, katılımcıların birbirlerinden habersiz olarak tekliflerini gizli bir şekilde sunmalarını ve aynı anda belirlenmiş bir zamanda ortaya çıkarmalarını gerektirir. Zamanlanmış taahhütlerin teklif gizliliği, bütünlüğü ve adilliği nasıl sağlayabileceği araştırılmıştır. Taahhüt oluşturma, taahhüt açma ve teklif doğrulama protokollerini ve gerekli güvenlik hususları da, bu tezde tartışılmaktadır. Araştırma, doğrulanabilir zamanlanmış taahhütler üzerine kurulu önerilen şemaların güvenliğini değerlendirmek için teorik analiz, algoritma tasarımı ve uygulamaları içeren çok disiplinli bir yaklaşım kullanmaktadır.

Anahtar Kelimeler: doğrulanabilir zamanlanmış taahhüt, kapalı teklif müzayedesi, zamanlanmış imza, adillik

To all the earthquake victims who lost their hope and to my dear grandmother...

ACKNOWLEDGMENTS

First of all, I would like to thank my esteemed supervisors, Prof. Dr. Murat Cenk and Assoc. Prof. Dr. Oğuz Yayla who worked with me throughout this long journey. Since I live abroad, I am grateful to them for working with me day and night, despite the time differences and all their busy schedules.

I would also like to thank the precious members of the thesis monitoring committee and the thesis jury, dear Prof. Dr. Ali Aydın Selçuk, Assoc. Prof. Dr. Pelin Angın, Assoc. Prof. Dr. Ali Doğanaksoy, Assoc. Prof. Dr. Fatih Sulak and Res. Asst. Talha Arıkan. They always enlightened my way with their questions and suggestions.

I am also grateful to my family, my beloved ones/close friends Mert, Nihal, Şahika, Buse, and my colleagues at Deloitte (especially those at the CIC) for their support during all these stressful times.

Finally, over the past 6 months, while working on my thesis, I faced two painful events. One was the loss of my dear grandmother, who had been a tremendous source of support and had prayed for the completion of this thesis. The other one was the earthquake disaster in which we lost thousands of people who had dreams like me. As I went along this path, I found the motivation to dedicate my thesis to them. Their memory and my purpose to dedicate my work to them inspired me. Thank you for being the light to me in those dark days when I lost you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

VTS	Verifiable Timed Signatures
BLS	Boneh–Lynn–Shacham
ECDSA	Elliptic Curve Digital Signature Algorithm
RSA	Rivest–Shamir–Adleman
VT-BLS	Verifiable Timed BLS Signature
VTC	Verifiable Timed Commitments
VTD	Verifiable Timed Dlog
MSP	Multi-signature Protocol
ASM	Accountable Subgroup Multi-signatures
TLP	Time Lock Puzzles
LHTLP	Linearly Homomorphic TLP
FHTLP	Fully Homomorphic TLP
ZKP	Zero-knowledge Proof
NIZK	Non-interactive ZKP
mASM	modified ASM
VAT-PS	Verifiable Accountable Timed Proxy Signatures
VAT-PMS	Verifiable Accountable Timed Proxy Multi-signatures
VT-MSP	Verifiable Timed Multi-Signature Protocol
VT-mASM	Verifiable Timed Accountable Subgroup Multi-signatures
PRAM	Parallel Random Access Machine
PPT	Probabilistic Polynomial Time
FPSB	First-price sealed-bid
crs	common reference string
PK	Public Key
apk	aggregated public key

CHAPTER 1

INTRODUCTION

In today's digital era, secure and trustworthy communication protocols are of utmost importance. Timed commitments play a vital role in ensuring integrity and fairness in various applications. This thesis delves into the exploration and analysis of verifiable timed commitments, specifically focusing on timed signature schemes and their application to sealed-bid auctions.

Verifiable timed commitments [28] provide a means to securely bind a commitment to a value while incorporating a temporal dimension. This temporal aspect introduces a time component, allowing for commitments to be time-stamped and validated within specified intervals. The utilization of timed commitments in cryptographic protocols ensures accountability, enhances transparency, and enables the detection of malicious behavior.

1.1 Background and Our Motivation

Thyagarajan *et al.* [29] have provided an efficient construction of verifiable timed commitments within well-known signature algorithms such as BLS [5], Schnorr [26], and ECDSA [15]. While exploring other timed signature schemes that hold practical relevance, we have identified proxy signatures and multi-signatures as particularly valuable due to their widespread use in various applications. Proxy signatures, initially proposed by Mambo *et al.* [21], enable the delegation of signing privileges while maintaining the security and integrity of the original signer's key. Subsequent research has focused on enhancing proxy signature schemes' efficiency, security, and

functionality. Notably, threshold proxy signature schemes [32] have emerged as significant improvements, introducing a threshold mechanism that requires the involvement of multiple proxies to generate a valid proxy signature. This approach strengthens security and mitigates the risk of a single point of failure. Various types of proxy signatures, including accountable proxy signatures [10], identity-based proxy (multi-)signatures [2, 25], anonymous proxy signatures [7]), certificate-based proxy signatures [18, 31], and homomorphic proxy signatures [19] have been proposed. In our research, we focus on making verifiable timed proxy signatures, allowing the delegation of signing authority to a proxy signer within a predefined time frame. Furthermore, multi-signature schemes are also important as they enable multiple parties to collectively and securely authorize transactions or actions, ensuring enhanced security, accountability, and decentralized decision-making. Although the accountable subgroup multi-signature (ASM) version of a multi-signature was initially introduced by Micali *et al.* [22], a recent and highly efficient ASM scheme introduced by Boneh *et al.* [4] has had a significant impact, especially in blockchain applications. This scheme allows users to generate their individual signatures first and subsequently select the subgroup for signing, facilitating flexibility and practical implementation. In our ASM-based proposals, we aim to use the advantages of ASM and design a timed version of it. All the schemes suggested in this research paper rely on pairings, which makes them ideal for low-bandwidth communication. This is because they can generate compact signature schemes, similar to the pairing-based (VT-BLS) [29] construction.

One other important motivation of this study is the application of a proposed signature scheme in a widely used application. Digital auctions have gained significant importance, serving as a widely used method for various purposes, including purchasing services or selling exclusive products. Given the broad range of applications, trust, verifiability, and fairness become indispensable attributes within auction mechanisms. Traditionally, timed commitments have been employed in auction setups, relying on a trusted third party. However, with the advent of blockchain technology, the focus has shifted toward decentralized auctions to eliminate the need for such trust requirements. A recent development in this area is the concept of a hybrid blockchain, which combines public and private blockchains [9]. Another notable example is "Strain" [3],

which utilizes blockchain infrastructure to ensure bid confidentiality, while Galal *et al.* [13] propose a construction emphasizing verifiability. Many proposals in this domain revolve around blockchain-based systems that leverage smart contracts, using programmable solutions to incorporate desired temporal features [29]. Malavolta *et al.* [20] proposed the use of homomorphic time-lock puzzles, similar to those in Verifiable Timed Commitments (VTC), for designing a sealed-bid auction. However, they suggested that for scalability, each user should place their bid in a single puzzle and submit it to the auctioneer, who would determine the winner through homomorphic calculations. This approach requires fully homomorphic time-lock puzzles (FHTLP) due to their complexity.

Overall, our research explores the integration of verifiable timed commitments with different signature protocols and uses the advantages derived from this integration in designing an auction.

1.2 Thesis Contribution

In our research, we introduce the concept of verifiable accountable timed proxy signatures, allowing the delegation of signing authority to a proxy signer within a predefined time frame. Similar to proxy signature schemes, the inclusion of verifiable timed commitments in accountable subgroup multi-signatures enables the creation of multi-signatures that require the participation of specific subgroup members within a designated time window. This feature ensures the accountability and verifiability of subgroup members' contributions, making the resulting multi-signature more robust and tamper-resistant. Considering these factors and the suitability for the design of proxy and multi-signatures, we propose a compact structure where we combine ASM within the concept of proxy multi-signatures and incorporate timed commitments. To the best of our knowledge, the suggested schemes are the first verifiable timed proxy signature and multi-signature proposals that take advantage of time lock puzzles. Additionally, we aim to adapt LHTLP to the auction schemes instead of FHTLP. We propose an auction design where the bid shares of users are distributed across multiple puzzles, instead of sharing the actual bid. This is also the first auction proposal that considers LHTLP in conjunction with verifiable timed commitments.

1.3 Organization

This thesis is structured into five sections. Section 1 provides an introduction to the topic, including the background, problem statement, objectives, and an overview of the structure. Section 2 presents the underlying cryptographic mechanisms and algorithms used in this work. Section 3 focuses on the integration of verifiable timed commitments into signature schemes, discussing their implications for generating secure and verifiable digital signatures. Section 4 explains our proposal for designing a fair sealed-bid auction using verifiable timed commitments. Finally, Section 5 concludes the study by summarizing the key findings, and outlining potential avenues for future research.

CHAPTER 2

PRELIMINARIES

This chapter of this thesis serves as a comprehensive introduction to the fundamental concepts and background information necessary for understanding verifiable timed commitments, and their usage in signature schemes and sealed-bid auctions. By explaining the algorithms and security requirements used during the research, it is aimed to deal with the proposed systems from different perspectives.

2.1 RSA Encryption

Key Generation: Let p, q be distinct prime numbers. Calculate $N = p \cdot q$ and $\phi(N) = (p-1) \cdot (q-1)$. Select an integer e, where $gcd(\phi(N), e) = 1; 1 < e < \phi(n)$. Find d such that $e \cdot d \mod \phi(N) = 1$. Parse public key$= (e, N)$ and private key$= (d, p, q)$.

Encryption: Let $M < N$ be a message to be encrypted. The encryption algorithm is $E(M) = M^e = c \mod N$, where c is the ciphertext.

Decryption: Let c be a ciphertext to be decrypted. The decryption algorithm is $D(c) = c^d = M \mod N$ where M is the plaintext.

2.2 RSA Signature

Key Generation: Let p, q be different prime numbers. Calculate N, ϕ, e, d, M as described in the RSA encryption.

RSA signature generation: Let $h = \text{Hash}(M)$. Compute the signature: $s = h^d \mod N$.

RSA signature verification: Calculate the hash of the message M as $h' = \text{Hash}(M)$. Check if $h' = h$. If it is equal, check if $h = s^e \mod N$. If it holds, the signature is verified.

2.3 Bilinear Pairing

A bilinear pairing is a function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ on groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ that satisfies bi-linearity and non-degeneracy:

Bilinearity: For all integers b and d, for all elements a in $\mathbb{G}_1$ and c in $\mathbb{G}_2$, $e(a^b, c^d) = e(a, c)^{bd}$.

Non-degeneracy: The pairing $e(g_1, g_2)$ is not equal to 1 for all generators g_1 and g_2 of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

Bilinear groups are fundamental to various cryptographic schemes, especially within certain types of digital signatures. The properties of bilinear groups enable the construction of efficient and secure cryptographic protocols based on mathematical operations and pairings between different groups.

2.4 BLS Signature

Given groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ and generators (g_1, g_2) of the group pair $(\mathbb{G}_1, \mathbb{G}_2)$, consider an efficient, non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. A function $H : \{0,1\}^* \to \mathbb{G}_1$ is defined to map any arbitrary binary string onto the group $\mathbb{G}_1$. Let $H_0, H_2 : \{0,1\}^* \to \mathbb{G}_1$ and $H_1 : \{0,1\}^* \to \mathbb{Z}_q$ be the hash functions which are collision resistant.

BLS signature [5] has three phases: Key Generation (KeyGen), Signature Generation (SigGen), and Verification.

1. Key Generation (KeyGen): Randomly select a secret key sk from the set of

integers modulo q, and use it to compute the corresponding public key $pk = g_2^{sk}$.

2. Signature Generation (SigGen): To generate a signature for a given message m, compute $\sigma = H(m)^{sk}$, where sk is the secret key.

3. Verification: Check if the following equation is satisfied:

$$e(H(m), pk) = e(\sigma, g_2) \tag{2.1}$$

2.5 Multi-signature Protocol (MSP)

Boneh *et al.* introduced the multi-signature protocol (MSP) in their work [4]. It is an efficient multi-signature scheme that builds on top of BLS and involves the steps given below:

Assume that the hash functions, groups, generators of the groups, and bilinear map are the same with BLS.

1. Key Generation: To generate a key pair in this scheme, a secret key sk_i is selected at random from $\mathbb{Z}_q$, and the corresponding public key pk_i is obtained by computing $g_2^{sk_i}$ for a user i.

2. Key Aggregation: Aggregated public key

$$apk = \prod_{i=1}^{n} pk_i^{a_i},$$

where

$$a_i = H_1(pk_i, \{pk_1, \ldots, pk_n\}).$$

3. Signature Generation: Calculate the multi-signature

$$\sigma_i = H_0(m)^{a_i sk_i}, \tag{2.2}$$

where m is the message to be signed.

4. Signature Aggregation: Multi-signature is

$$\sigma = \prod_{j=1}^{n} \sigma_i.$$

5. Verification: Check if

$$e(H_0(m), apk) = e(\sigma, g_2). \tag{2.3}$$

2.6 Accountable Subgroup Multi-signatures (ASM)

An accountable subgroup multi-signature (ASM) scheme is a type of multi-signature in which a message m can be signed by any subgroup S of a group G, and the signatories from the subgroup S are responsible for the signature. In multi-signatures, accountability pertains to the capacity to recognize and assign the actions of individual signers within the scheme. It guarantees that each signer can be held accountable for their unique signatures and actions within the group. ASM was defined first time in 2001 [22]. After the term construction with a generic protocol definition, one of the most important improvements in ASM schemes is the ASM scheme constructed by Boneh *et al.* [4]. The objective of developing this "short" ASM scheme was to reduce the size of the Bitcoin blockchain. In this algorithm, "short" means the signature size is $O(\lambda)$-bits, where λ is the security constant. This ASM scheme showed that it is very practical and applicable to blockchain cryptosystems. In addition, accountability is primarily ensured by the presence of a membership key unique to each user, corresponding to the specific group they belong to.

2.6.1 ASM by Boneh *et al.*

Boneh *et al.*'s ASM scheme [4] consists of 5 tuples: KeyGen, Group Setup, SignatureGen, Signature Aggregation (Signature Aggr), and Verification. These steps are explained below. PK is defined as the collection of public keys belonging to the members of group $\mathbb{G}$, denoted by $pk_1, \ldots, pk_n$.

- **KeyGen:** Every user $i \in \mathbb{G}$ gets a random secret key $sk_i \leftarrow \mathbb{Z}_q$ and calculates public key $pk_i \leftarrow g_2^{sk_i}$ where g_2 is the generator of the group $\mathbb{G}_2$.
- **Group Setup:** Every user performs group setup using one round interactive protocol.

- Users are responsible for computing the group's aggregated public key apk as: $apk = \prod_{i=1}^{n} pk_i^{a_i}$ where $a_i = H_1(pk_i, PK)$.
- Each user sends

$$\mu_{ji} \coloneqq H_2(apk, j)^{a_i sk_i} \tag{2.4}$$

to the j-th member for $j = 1, 2, \ldots, n$ and $j \neq i$.

- After receiving μ_{ij}, the user calculates $\mu_{ii} = H_2(apk, i)^{a_i sk_i}$.
- For a user i, the membership key is $mk_i = \prod_{j=1}^{n} \mu_{ij}$. The membership keys are specific to the group that contains the users and are used to create the signature of a user.

- **SignatureGen:** A signer calculates his/her individual signature as

$$s_i = H_0(apk, m)^{sk_i} \cdot mk_i$$

and delivers s_i to the combiner who is responsible for creating the multi-signature.

- **Signature Aggr:** The combiner first forms the set of signers $S \subseteq \mathbb{G}$. Then, she computes the aggregated subgroup multi-signature $\sigma = (s, pk)$ where $s = \prod_{i \in S} s_i$ and $pk = \prod_{i \in S} pk_i$.

- **Verification:** The signature $\sigma = (s, pk)$ can be verified by anyone in possession of par, apk, S, m, σ by checking if:

$$e(s, g_2) = e(H_0(apk, m), pk).e(\prod_{j \in S} H_2(apk, j), apk)$$

The ASM's membership key and individual signature generation stages contribute to achieving accountability and timed-based construction.

2.6.2 Security Requirements for Multi-signatures

As an expected security property, the unforgeability of multi-signatures is important because it enhances security by preventing unauthorized parties from creating valid signatures. It ensures that transactions require the consensus and cooperation of multiple parties, reducing the risk of fraud, enabling distributed trust, and providing accountability in digital transactions.

Definition 1 (Unforgeability)**.** *The unforgeability of a multi-signature scheme is defined by a three-stage game [4]:*

Setup: *A challenger runs KeyGen to generate a key pair* (pk', sk') *and sends* pk' *to adversary* Adv*.*

Signature Queries: Adv *issues chosen message queries and receives their signatures(signed with* sk'*) from the challenger.*

Output: Adv *outputs a forgery: list of public keys*$(pk_1, pk_2, pk_3, \ldots, pk_n)$*, a message from the message set(*$m \in M$*), and a multi-signature* σ*. The* Adv *wins the game if* $pk' \in PK$*,* Adv *creates no signing queries on* m *and*

$$\textit{Verification}(pk', pk_1, pk_2, pk_3, \ldots, pk_n, m, \sigma) = 1$$

Based on this game, Adv *is a* $(\tau, q_S, q_H, \epsilon)$*-forger for a multi-signature if it runs in time* τ*, creates* q_S *many signature queries, makes* q_H *many random oracle queries and becomes the winner of the game with probability at least* ϵ*. That multi-signature is unforgeable if no one like that exists.*

2.7 Proxy Signatures

A proxy signature allows one entity, known as the original signer/delegator, to delegate their signing authority to another entity, known as the proxy signer. The proxy signer can then generate signatures on behalf of the delegator. These are commonly used in such cases where the delegator wants to delegate signing capabilities temporarily or to perform signatures without direct involvement. In general, they work as follows:

- **Key Generation:**

  The main(original) signer and the proxy signer generate their key pair consisting of a private key and a corresponding public key.

- **Setup:**

The original signer and the proxy signer establish a trusted relationship, typically through a secure communication channel or a trusted intermediary. Then, the original signer authorizes the proxy signer to act on their behalf.

- **Delegation:**

  In the delegation step, the original signer creates a warrant w that contains relevant information about the delegation and signing process. This information generally contains the identity of the signers, the delegation time range, and the details of the message.

- **Proxy Signature Generation:**

  The proxy signer first checks if the delegation information is correct or not. Then, they create the signature.

- **Proxy Signature Verification:**

  A verifier receives the message, the proxy signature, and the public keys of the original and proxy signer and checks if the signature is valid or not.

2.7.1 Security Requirements

In the proxy signatures with the delegation by the warrant, several security requirements play a crucial role [17]. Verifiability ensures that the validity of a proxy signature can be confirmed by any party, including the warrantor or a third party. Strong unforgeability guarantees that generating a valid proxy signature without the proxy signer's private key is computationally infeasible, preventing unauthorized parties from creating fraudulent signatures. Strong undeniability establishes evidence or proof to prevent the proxy signer from denying their actions. Strong identifiability enables the unique identification of the proxy signer. Finally, the prevention of misuse ensures that the delegated authority is used only within the defined limits and restrictions specified by the warrant, preventing any unauthorized or improper use of the signing rights.

2.8 Zero Knowledge Proof (ZKP)

ZKP enables the demonstration of a statement's truthfulness without revealing anything other than the statement's own validity. The non-interactive version of ZKP [8] allows proof of the statement by the prover without any interaction with the verifier. NIZK [8] contains 3-tuples: ZKSetup, ZKProve and ZKVerify:

- **ZKSetup phase**(1^λ)**:** Note that 1^λ is a security constant .

  -The output will be the "common reference string" *(crs)*.

- **ZKProve phase**(crs, x, w)**:** The presence of a "witness" (w) is crucial in proving the truthfulness of a statement (x).

  -The statement's validity with the witness outputs the proof π.

- **ZKVerify phase**(crs, x, π)**:** The verification will show if the statement and the corresponding proof are true.

  -The output will be "yes/verified" or "no'/not verified", depending on whether the requirements for the verification of the protocol are met.

2.9 Timed Cryptography

Timed-release cryptography aims to achieve the idea that the message can be sent to the future and opened only after some predefined time T. As the first construction of timed cryptography, the concept of "timed commitments" was introduced for the first time in 2000 [6]. A timed commitment scheme for a message or any value provides the sender with a commitment to the value. After some time T, the sender can prove this commitment or if the sender refuses this statement, the receiver can retrieve the committed message within time T by forcing it open. Therefore, a time commitment basically contains 3 phases:

- **Commitment generation:** To commit on a message.
- **Open:** Sender can open the commitment to show the message.

- **Forced Open:** If the sender chooses not to disclose the commitment, the receiver can use this algorithm to prove the commitment and unveil the committed message within a duration of T.

Timed cryptography contains several types of applications such as time-lock puzzle generation, timed release of signatures as well as timed commitments.

2.9.1 Timed Signatures

Timed signatures are a type of time commitment in which the sender can commit to a signature of a message/document. More formally, a timed signature contains five tuples:

1. Setup phase: Any message signer generates a key pair $(publickey, privatekey)$ using a $KeyGen$ algorithm.

2. A signature generation algorithm: A valid signature contains 3 tuples, namely S, C, and Sig where C is a timed commitment when committing to a string S and Sig can be verified using $publickey$ as a valid signature on a message M.

3. Commit phase: The signer of a message chooses a random private string S and creates a timed commitment on S named C.

4. Open phase: The sender (signer) opens the committed string S. The receiver (verifier) gets a valid signature tuple namely $\{S, C, Sig\}$.

5. Forced Open phase: Use forced open protocol to retrieve the committed value S by the related timed commitment phase.

One crucial requirement for timed signatures is verifiability, which is proof that the receiver gets the correct message with a valid signature. An efficient version of the term “Verifiable Timed Signature" was defined recently in an article [29] by Thyagarajan *et al.* as an application of timed signature.

2.9.2 Time Lock Puzzles (TLP) and Advantages of Homomorphism:

TLPs [24] let a person encrypt a message to the future. According to the verifiable timed signature (VTS) and verifiable timed commitment (VTC) proposed by [29], usage of TLP provides efficiency to the current applications, and the linearly homomorphic time lock puzzle (LHTLP) version makes them even more efficient where ForceOpen algorithm needs to solve only one packed puzzle instead of many. We use the same notation of LHTLP given in [29] while constructing timed lock puzzles. In addition, we use the range proofs for linearly homomorphic time lock puzzles. These range proofs enable the verification of properties like the value being within a certain range or meeting certain arithmetic conditions. By using linearly homomorphic time lock puzzles, it becomes possible to perform computations on locked values. This allows the prover to create proof for a locked value without revealing itself. The verifier can then verify the range proof without knowing the actual value. In this thesis, we follow the range proofs provided by [29]. In our schemes, range proofs are used in timed structures where the signer needs to verify a value or the signature created by the signer needs to be verified. Those proofs are zero-knowledge proofs (denoted as ZKProve).

LHTLP [20] over the ring $(\mathbb{Z}_N, +)$ contains a tuple of 4 algorithms: PuzzleSetup, PuzzleGeneration, PuzzleSolve, and PuzzleEval and is defined as follows:

- **PuzzleSetup phase($1^\lambda, T$):** Note that 1^λ is a security constant and T is the agreed time.

  -With prime numbers p' and q', sample $p = 2p' + 1$ and $q = 2q' + 1$ and set $N \coloneqq pq$.
  -Sample a uniform $\tilde{g} \xleftarrow{\$} \mathbb{Z}_N^*$ and set $g \coloneqq -\tilde{g}^2 \mod N$.
  -Calculate and set $h \coloneqq g^{2^T}$ that can be optimized by reducing $2^T \mod \phi(N)/2$.
  -In the end, the output will be public parameters $pp \coloneqq (T, N, g, h)$.

- **PuzzleGeneration phase(pp, s):** Note that s is the solution from the solution set S. The solution is the value to be locked in a puzzle that is intended to be solved after time T.

  -Set $pp \coloneqq (T, N, g, h)$.

-Sample an $r \xleftarrow{\$} \{1, \ldots, N^2\}$.

-Calculate $u \coloneqq g^r \mod N$ and $v \coloneqq h^{r.N}.(1+N)^s \mod N^2$.

-In the end, the output will be the puzzle $Z \coloneqq (u, v)$.

- **PuzzleSolve phase**(pp, Z)**:**

  -Set $pp \coloneqq (T, N, g, h)$.

  -Set the puzzle $Z \coloneqq (u, v)$.

  -Calculate $w \coloneqq u^{2^T} \mod N$ by Repeated Squaring Method.

  -In the end, the output will be the solution $s \coloneqq \frac{v/(w)^N (mod N^2)-1}{N}$.

- **PuzzleEval phase**($pp, Z_1, \ldots, Z_N$)**:**

  -Set $pp \coloneqq (T, N, g, h)$.

  -Set every $Z_i \coloneqq (u_i, v_i) \in \mathbb{J}_N \times Z^*_{N^2}$.

  -Calculate $\tilde{u} \coloneqq \prod_{i=1}^n u_i \mod N$ and $\tilde{v} \coloneqq \prod_{i=1}^n v_i \mod N^2$.

  -In the end, the output will be the puzzle $Z' = (\tilde{u}, \tilde{v})$.

  Note that, PuzzleSetup, PuzzleGeneration, and PuzzleEval phases are the probabilistic algorithms whereas the PuzzleSolve phase is a deterministic algorithm.

Here, it is important to explain NIZK for LHTLP [29]:

1. **ZKSetup phase:** Let N be an RSA modulus, pp are defined in LHTLP, intervals for the statements are L and B with $B < L$ and let k is the statistical security constant. Let the input here is the time lock puzzles $Z_1, \ldots, Z_l$.

2. **ZKProve phase:** Let $wit \coloneqq ((x_1, r_1), \ldots, (x_l, r_l))$ denote the witness, where $x_i \in [-B, B]$ for all i. If $Z_i \leftarrow HTLP.PuzzleGeneration(pp, x_i; r_i)$ for any i, then the ZKProve phase performs the following steps:

   (a) Let $y_1, \ldots, y_k \leftarrow [-L/4, L/4]$, $r_1^{'}, \ldots, r_k^{'}$ be the values of the corresponding ring.

   (b) For $i = 1, \ldots, k$ calculate $D_i \leftarrow HTLP.PGen(pp, y_i; r_i^{'})$.

   (c) Calculate $(t_1, \ldots, t_k) \leftarrow H(Z_1, \ldots, Z_l, D_1, \ldots, D_k), \forall t_i \in \{0,1\}^l$.

   (d) For $i = 1, \ldots, k$, calculate $v_i \leftarrow y_i + \sum_{j=1}^l t_{i,j}.x_j$ and $w_i \leftarrow r_i^{'} + \sum_{j=1}^l t_{i,j}.r_j$.

(e) Set $\pi \leftarrow (D_i, v_i, w_i)_{i\in[k]}$, provide π.

3. **ZKVerify phase:**

   (a) Calculate $(t_1, \ldots, t_k) \leftarrow H(Z_1, \ldots, Z_l, D_1, \ldots, D_k)$.

   (b) For $i = 1, \ldots, k$ calculate if $v_i \in [-L/2, L/2]$, then calculate $F_i \leftarrow D_i.\prod_{j=1}^{l} Z_j^{t_{i,j}}$ and calculate if $F_i = HTLP.PuzzleGeneration(pp, v_i; w_i)$.

   (c) If all the conditions are satisfied, output 1, else output 0.

2.9.3 Verifiable Timed Signatures (VTS)

Thyagarajan *et al.* recently proposed [29] efficient versions of VTS for BLS [5], Schnorr [26] and ECDSA [15]. VTS versions of these algorithms do not require any modification of signature algorithms, instead, they are chosen as committed signatures in a commitment scheme.

VTS has 4 phases, namely Commit, Vrfy, Open, and ForceOp:

- Commit : (C,π) $\leftarrow$ Commit(σ,T)
- Vrfy : 0 /1 $\leftarrow$ Vrfy(pk, m, C, σ)
- Open : (σ, r) $\leftarrow$ Open(C)
- ForceOp : $\sigma \leftarrow$ ForceOp(C)

We will include the VTS for the BLS (VT-BLS) because it is the only one that is pairing-based, and the pairing-based ones will be the focus of this study.

2.9.3.1 Verifiable Timed BLS Signature (VT-BLS):

In order to show how VTS phases are used for BLS, the following VT-BLS [29] algorithm steps are explained. Note that, n will be used as a security parameter, threshold $t \coloneqq n/2+1$ and $|\sigma| = \lambda$ will be the bit-length of σ, $H' : \{0,1\}^* \rightarrow I \subset [n]$ with $|I| = t - 1$ is a random oracle. As a pairing-based efficient timed signature,

VT-BLS uses threshold secret sharing to increase performance. Specifically, it uses a cut-and-choose protocol to use the threshold value.

- **Setup phase:** Run ZKSetup(1^λ) to generate crs_{range}. Generate public parameters

$$pp \leftarrow \text{LHTLP.PuzzleSetup}(1^\lambda, T)$$

and output $crs \coloneqq (crs_{range}, pp)$.

- **Commit and prove phase:** For input (crs, wit), follow the steps.

-$wit \coloneqq \sigma$, $crs \coloneqq (crs_{range}, pp)$, pk is the public key generated in the BLS, m is the message.

-For all $i \in [t-1]$, sample $\alpha_i \leftarrow \mathbb{Z}_q$ and fix $\sigma_i = H(m)^{\alpha_i}$ and $h_i \coloneqq g_2^{\alpha_i}$.

-For all $i \in \{t, \ldots, n\}$:

$$\sigma_i = \left(\frac{\sigma}{\prod_{j \in [t-1]} \sigma_j^{l_j(0)}} \right)^{l_i(0)^{-1}} \tag{2.5}$$

and

$$h_i = \left(\frac{pk}{\prod_{j \in [t-1]} h_j^{l_j(0)}} \right)^{l_i(0)^{-1}} \tag{2.6}$$

Here l_i is the i-th Lagrange Polynomial basis.

-For $i \in [n]$, generate puzzles and proofs:

$$r_i \leftarrow \{0,1\}^\lambda, Z_i \leftarrow \text{LHTLP.PuzzleGeneration}(pp, \sigma_i; r_i)$$

and

$$\pi_{range,i} \leftarrow \text{ZKProve}(crs_{range}, (Z_i, 0, 2^\lambda, T), (\sigma_i, r_i))$$

-Calculate

$$I \leftarrow H'(pk, (h_1, Z_1, \pi_{range,1}), \ldots, (h_n, Z_n, \pi_{range,n})).$$

-Results are the commitment $C \coloneqq (Z_1, \ldots, Z_n, T)$ and corresponding range proof which is

$$\pi \coloneqq (h_i, \pi_{range,i_{i \in [n]}}, I, \{\sigma_i, r_i\}_{i \in I}).$$

- **Verification phase:** (crs, pk, m, C, π) is the input values and the Vrfy algorithm works as follows:

  -Parse $C := (Z_1, \ldots, Z_n, T)$, $\pi := (h_i, \pi_{range,i_{i\in[n]}}, I, \{\sigma_i, r_i\}_{i\in I})$ and $crs := (crs_{range}, pp)$.

  -If any of the below conditions are correct, the Vrfy algorithm outputs 0:

  1. There is $j \notin I$ satisfying

  $$\prod_{i\in I} h_i^{l_i(0)}.h_j^{l_j(0)} \neq pk.$$

  2. There is $i \in [n]$ satisfying

  $$\text{ZKVerify}(crs_{range}, (Z_i, 0, 2^\lambda, T), \pi_{range,i}) \neq 1.$$

  3. There is $i \in I$ satisfying

  $$Z_i \neq \text{LHTLP.PuzzleGeneration}(pp, \sigma_i; r_i)$$

  or

  $$e(g_2, \sigma_i) \neq e(h_i, H(m)).$$

  4. $I \neq H'(pk, (h_1, Z_1, \pi_{range,1}), \ldots, (h_n, Z_n, \pi_{range,n}))$.

- **Open phase:** The result of the open phase is opening the commitment and receiving $(\sigma, \{r_i\}_{i\in[n]})$. The committer is expected to open at least the puzzles for the challenge set I chosen by the verifier.

- **ForceOp (forced open) phase:** This phase takes input $C := (Z_1, \ldots, Z_n, T)$ and:

  -Performs $\sigma_i \leftarrow \text{LHTLP.PuzzleSolve}(pp, Z_i)$ for $i \in [n]$ and receives all the signature shares. It should be observed that, given the committer has revealed $t-1$ puzzles, the ForceOp step will only involve solving $(n-t+1)$ puzzles.

  -Output $\sigma := \prod_{j\in[t]}(\sigma_j)^{l_j(0)}$ by considering the first t signatures shares are valid.

2.9.4 Verifiable Timed Commitments (VTC)

Verifiable timed commitment (VTC) or verifiable timed dlog (VTD) [28] is used to generate a timed commitment for a secret value $x \in \mathbb{Z}_q^*$ satisfying $h = g^x$ where h is a publicly known value and g is a generator of G, which is a group of order q. This structure of VTC is similar to VTS in terms of steps followed and algorithms used.

- **Setup phase:** Run ZKSetup(1^λ) to generate crs_{range}, generate public parameters

  $$pp \leftarrow \text{LHTLP.PuzzleSetup}(1^\lambda, T)$$

  and output

  $$crs := (crs_{range}, pp).$$

- **Commit and prove phase:** For a given (crs, wit), follow the steps:

  -$wit := x$, $crs := (crs_{range}, pp)$, $h := g^x$.

  -$\forall i \in [t-1]$, sample $x_i \leftarrow Z_q$ and fix $h_i = g^{x_i}$.

  -For all $i \in t, \ldots, n$ compute

  $$x_i = \left(x - \sum_{j \in [t]} x_j . l_j 0 \right) . l_i(0)^{-1} \tag{2.7}$$

  and

  $$h_i = \left(\frac{h}{\prod_{j \in [t]} h_j^{l_j(0)}} \right)^{l_i(0)^{-1}} \tag{2.8}$$

  Here, l_i is the i-th Lagrange polynomial basis.

  -For $i \in [n]$:

  $$r_i \leftarrow \{0,1\}^\lambda, Z_i \leftarrow \text{LHTLP.PuzzleGeneration}(pp, x_i; r_i)$$

  and

  $$\pi_{range,i} \leftarrow \text{ZKProve}(crs_{range}, (Z_i, a, b, T), (x_i, r_i)).$$

  -Calculate

  $$I \leftarrow H'(pk, (h_1, Z_1, \pi_{range,1}), \ldots, (h_n, Z_n, \pi_{range,n})).$$

-Output the commitment $C := (Z_1, \ldots, Z_n, T)$ and corresponding range proof

$$\pi := (\{h_i, \pi_{range,i}\}_{i\in[n]}, I, \{x_i, r_i\}_{i\in I}).$$

-Final output is (h, C, π).

- **Verification phase:** Given (crs, h, C, π), the Vrfy works as follows:

  -Let $C \coloneqq (Z_1, \ldots, Z_n, T)$, $crs := (crs_{range}, pp)$ and

  $$\pi := (\{h_i, \pi_{range,i}\}_{i\in[n]}, I, \{x_i, r_i\}_{i\in I}).$$

  -If any of the below conditions are correct, the Vrfy algorithm outputs 0. Therefore, it is expected that these conditions are wrong.

  1. There is $j \notin I$ satisfying

     $$\prod_{i\in I} h_i^{l_i(0)}.h_j^{l_j(0)} \neq h.$$

  2. There is $i \in [n]$ satisfying

     $$\text{ZKVerify}(crs_{range}, (Z_i, a, b, T), \pi_{range,i}) \neq 1.$$

  3. There is $i \in I$ satisfying $Z_i \neq$ LHTLP.PuzzleGeneration$(pp, x_i; r_i)$ or $h_i = g^{x_i}$.

  4. $I \neq H'(pk, (h_1, Z_1, \pi_{range,1}), \ldots, (h_n, Z_n, \pi_{range,n}))$.

- **Open phase:** The result of the open phase is opening the commitment and receiving $(x, \{r_i\}_{i\in[n]})$. The committer is expected to open at least the puzzles for the challenge set I chosen by the verifier.

- **ForceOp (forced open) phase:** This phase takes $C := (Z_1, \ldots, Z_n, T)$ and:

  -Calculates $x_i \leftarrow$ LHTLP.PuzzleSolve(pp, Z_i) for $i \in [n]$ to retrieve all key shares. It should be observed that, given the committer has revealed $t-1$ puzzles, the ForceOp step will only involve solving $(n-t+1)$ puzzles.

  -Output $x \coloneqq \sum_{j\in[t]}(x_j).l_j(0)$ by considering the first t signature shares are the expected ones.

2.9.5 Security Requirements for VTS and VTC

There are two main security requirements for VTS-VTC schemes: soundness and privacy. Soundness promises to the user that the ForceOp algorithm will reveal the committed value after T time, under the given commitment C. All Parallel Random Access Machine (PRAM) algorithms [29] with run-time less than T can reveal the committed value using commitment and proof with only a negligible probability. The formal definitions for VTS and VTC are as follows.

Definition 2 (Soundness/Simulation-soundness of VTS)**.** *A VTS scheme is sound if $\forall A$ which are the probabilistic polynomial time adversaries and $\forall \lambda \in \mathbb{N}$, there is a negligible function $negl(\lambda)$ that satisfies the following statement:*

$$\Pr\left[\begin{array}{c} b_1 = 1 \wedge b_2 = 0 \end{array} : \begin{array}{c} (pk, m, C, \pi, T) \leftarrow A(1^\lambda) \\ (\sigma, r) \leftarrow ForceOp(C) \\ b_1 := Verification - forVTS(pk, m, C, \pi) \\ b_2 := Verification - forSignature(pk, m, \sigma) \end{array}\right] \leq negl(\lambda)$$

.

The concept of simulation-soundness indicates that a prover cannot easily persuade a verifier of a false statement, even if the prover has the ability to generate many simulated proofs for statements they desire.

Definition 3 (Privacy of VTS)**.** *A VTS scheme can be considered as private if there exists a PPT simulator S, $negl$ as defined in Definition 2, and a polynomial $\tilde{T}$ satisfying $\forall$ polynomials $T > \tilde{T}$, the PRAM algorithms A whose running time is at most t which is smaller than T, $\forall m \in \{0,1\}^*$ and $\forall \lambda \in \mathbb{N}$, the following statement is satisfied:*

$$\left|\Pr\left[A(pk, m, C, \pi) = 1 \ : \begin{array}{r} (pk, sk) \leftarrow KeyGeneration(1^\lambda) \\ \sigma \leftarrow SignatureGeneration(sk, m) \\ (C, \pi) \leftarrow Commit(\sigma, T) \end{array}\right] - \Pr\left[A(pk, m, C, \pi) = 1 \ : \begin{array}{c} (pk, sk) \leftarrow KeyGeneration(1^\lambda) \\ (C, \pi, m) \leftarrow S(pk, T) \end{array}\right]\right|$$

$\leq negl(\lambda)$.

Definition 4 (Soundness/Simulation-soundness of VTC)**.** *A VTC scheme is sound if $\forall A$ which are the probabilistic polynomial time adversaries and $\forall \lambda \in \mathbb{N}$, there is a*

negligible function $negl(\lambda)$ that satisfies the following statement:

$$\Pr\left[\begin{array}{c} b_1 = 1 \wedge b_2 = 0 \end{array} : \begin{array}{c} (h, C, \pi, T) \leftarrow A(1^\lambda) \\ x \leftarrow ForceOp(C) \\ b_1 := Verification - forVTC(h, C, \pi) \\ b_2 := (h = g^x) \end{array}\right] \leq negl(\lambda)$$

.

Definition 5 (Privacy of VTC)**.** *A VTC scheme can be considered as private if there exists a Probabilistic Polynomial Time(PPT) simulator S, $negl$ as defined in Definition 2, and a polynomial $\tilde{T}$ satisfying $\forall$ polynomials $T > \tilde{T}$, the PRAM algorithms A whose running time is at most t which is smaller than T, all $\lambda \in \mathbb{N}$ and $\forall \lambda \in \mathbb{N}$, the following statement is satisfied:*

$$\Pr\left[\begin{array}{c} b' = b \end{array} : \begin{array}{c} x \leftarrow \mathbb{Z}_q^*, h := g^x, b \leftarrow \{0, 1\} \\ b = 0 \Rightarrow (C, \pi) \leftarrow Commit(x, T) \\ b \neq 0 \Rightarrow (C, \pi) \leftarrow S(h, T) \\ b' \leftarrow A(h, C, \pi) \end{array}\right] \leq negl(\lambda)$$

.

2.10 Auction Schemes

Auctions play a crucial role in various domains, facilitating transactions by allowing participants to competitively bid for goods, services, or assets. However, in an increasingly digitized world, the privacy of bids of the participants and digital identities has become a significant concern [14, 23]. Timed commitments have emerged as a promising approach for privacy preservation in auction protocols [27].

Auctions are mechanisms in which participants compete by submitting bids to acquire a particular item. There are typically two categories of auctions: open auctions and sealed-bid auctions. Open types of auctions are English or Dutch-style auctions which are bidding by increasing the bid and decreasing the bid respectively. On the other hand, sealed-bid auctions are either first-price or second-price auctions, which means the winner pays either the highest price or the second-highest price, respectively. In this study, we focus on first-price sealed-bid auctions (FPSB). Further details and explanations can be found in the comprehensive survey [1].

2.10.1 Privacy and Security Considerations in Auctions

Preserving privacy and security in auctions involves safeguarding the identities of the participants, bids, and other confidential information from unauthorized disclosure. Cryptographic protocols and security mechanisms are commonly employed to achieve this goal while enabling participants to engage in the bidding process.

In an ideal auction protocol, there are several key points [1] that one can consider even though satisfying all of them is a challenging task. They are given as follows:

- **Non-repudiation:** Preventing bidders from denying their actions, ensuring that they cannot later deny truly submitted bids.
- **Privacy:** Keeping the losing bids unrevealed.
- **Verifiability:** Enabling the ability to verify and validate the correctness and accuracy of the auction outcome.
- **Secrecy:** Employing measures to protect the identity of the participants.
- **Robustness:** When dealing with parties who are willing to cheat, it is essential to have a counter strategy in place that effectively prevents their dishonest actions.
- **Fairness:** Enhancing public perception and trust in the auction process. Participants are more likely to engage and submit competitive bids if they have confidence that the process is fair and unbiased. This, in turn, can lead to increased participation and more favorable outcomes for both the auction organizer and participants.
- **Anonymity:** Maintaining the privacy of the bidder-bid relationship, ensuring that no indications or traces are left that can link a bidder to their bid.

CHAPTER 3

VERIFIABLE TIMED COMMITMENT APPLICATIONS WITHIN SIGNATURE SCHEMES

Considering the usage and the efficiency of VT-BLS, VT-ECDSA, and VT-Schnorr structures defined for VTS, which other signing algorithms can be used with them is discussed in this study. Adding accountability to the chosen VTS has been one of our main motivations. In this context, we chose accountable subgroup multi-signatures due to their default accountability feature and accountable proxy signatures that have many applications today. In this section, we explained our proposals for those signature schemes blended with the VTC concept. While creating these designs, we slightly modified the existing algorithm structures at some points and made them suitable for the timed signature/commitment structures.

3.1 Verifiable Accountable Timed Proxy Signatures (VAT-PS)

In this section, we introduce the “Verifiable Accountable Timed Proxy Signatures (VAT-PS)" which presents three practical construction scenarios for pairing-based proxy signature schemes. Our scheme focuses on incorporating verifiability, accountability, and timed cryptography to define a proxy signature providing all of these properties in a single scheme. Unlike the traditional proxy signatures, our design brings a novel approach by using timed commitments during the delegation process or signing process which allows a design of a proxy signature involving only the original signer and the proxy signer. That means, there is no need to have a trusted or semi-trusted third party in such a proxy scheme. Its consideration of verifiability and accountabil-

ity makes our proposal a desired digital signature design because it provides better transparency and traceability. Our proposed scenarios use the delegation by warrant type of proxy design. The warrant, generated by the original signer, plays a crucial role in establishing the accountability and trustworthiness of the delegation process in our scheme. The warrant consists of the membership keys of the users and the information about the message to be signed. Moreover, one can observe that our VAT-PS constructions can be seen as a two-person multi-signature scheme in a sense, consisting of the original signer and the proxy signer. In such schemes, both the membership key and the individual signature calculation part of the ASM structure proposed by Boneh *et al.* [4] are quite suitable. However, instead of using the individual signature generation of the ASM scheme directly, we slightly modify it to be compatible with the timed commitments proposed in [29]. The proposed schemes are given under the following three scenarios.

3.1.1 Scenario 1-VAT-PS by timed delegation

In the first scenario, we aim to allow the original signer to delegate their signing rights to a proxy signature with a specific condition. The delegation is timed, meaning that after a specific time T, the proxy signature gains the authority to sign on behalf of the original signer. This scenario is a pretty suitable use case in order to apply timed signatures. To achieve this, it is intended that the original signer signs the warrant with the timed signature and sends it to the proxy signer. In this way, the authority can only be used in the future. Figure 3.1 shows the high-level construction of the timed delegation of the signature and the steps are explained below.

1. **Key Generation:** The original signer and the proxy signer generate their key pair, (sk_i, pk_i) with the same generation procedure as the BLS scheme. For simplicity, we call the original signer user 1 and the proxy signer as user 2. Therefore, their key pairs are (sk_1, pk_1) and (sk_2, pk_2), respectively.

2. **Setup:** The setup process uses the ASM group setup parameters and functions (H_0, H_1 and H_2 are defined such that H_0 and H_2 map binary strings to $\mathbb{G}_1$ and H_1 maps binary strings to $\mathbb{Z}_q$.) with an additional function $H_3 : \mathbb{G}_1 \rightarrow \mathbb{Z}_q$.

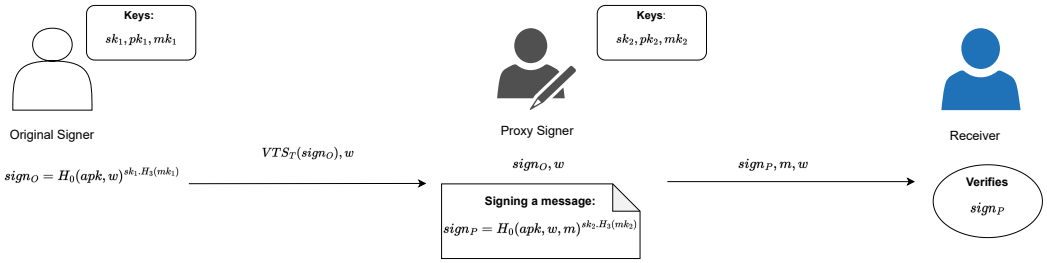

Figure 3.1: VAT-PS by Timed Delegation

Therefore, considering one original signer and one proxy signer in the process, the group setup algorithm works as follows. Both users compute the aggregated public key

$$apk \leftarrow \prod_{i=1}^{2} pk_i^{H_1(pk_i, \{pk_1, pk_2\})}.$$

Let $H_1(pk_i, \{pk_1, pk_2\})$ be parsed as a_i. Calculating the membership keys is similar to the ASM, both signers have

$$mk_i \leftarrow \prod_{j=1}^{2} \mu_{ij}.$$

3. **(Timed) Delegation:** As a part of the delegation process, the original signer creates a warrant value of w. Later, the original signer computes $sign_O \leftarrow H_0(apk, w)^{sk_1 \cdot H_3(mk_1)}$. This part is the slight modification of ASM signature generation in order to be compatible with the VT-BLS structure and the same modification will be used for the other timed proxy constructions as well. Let k be the value of $sk_1 \cdot H_3(mk_1)$ in $\mathbb{Z}_q$. Then, k can be considered as the new secret key of the original signer. In this case, $g_2^k = pk'$ is the new public key of the original signer. Since the $sign_O$ is a BLS signature of apk, w with the new modified keys of the original signer, it is compatible with the VT-BLS structure.

   (a) **Setup phase:** Run ZKSetup(1^λ) to generate crs_{range}. Generate public parameters

   $$pp \leftarrow \text{LHTLP.PuzzleSetup}(1^\lambda, T)$$

   and output $crs := (crs_{\text{range}}, pp)$.

   (b) **Commit and prove phase:** For input (crs, wit), follow the steps.

   -$wit := sign_O$, $crs := (crs_{\text{range}}, pp)$, pk' is the public key generated as in the BLS scheme, m is apk, w which is used as a message.

-$\forall i \in [t-1]$, take $\alpha_i \leftarrow \mathbb{Z}_q$ and fix $\sigma_i = H(m)^{\alpha_i}$ and $h_i := g_2^{\alpha_i}$.

-$\forall i \in \{t, \ldots, n\}$ calculate:

$$\sigma_i = \left(\frac{\sigma}{\prod_{j \in [t-1]} \sigma_j^{l_j(0)}} \right)^{l_i(0)^{-1}} \tag{3.1}$$

and

$$h_i = \left(\frac{pk^{'}}{\prod_{j \in [t-1]} h_j^{l_j(0)}} \right)^{l_i(0)^{-1}}, \tag{3.2}$$

-For $i \in [n]$, generate puzzles and proofs

$$r_i \leftarrow \{0,1\}^{\lambda}, Z_i \leftarrow \text{LHTLP.PuzzleGeneration}(pp, \sigma_i; r_i)$$

and

$$\pi_{\text{range},i} \leftarrow \text{ZKProve}(crs_{\text{range}}, (Z_i, 0, 2^{\lambda}, T), (\sigma_i, r_i)),$$

respectively.

-Calculate

$$I \leftarrow H'(pk^{'}, (h_1, Z_1, \pi_{\text{range},1}), \ldots, (h_n, Z_n, \pi_{\text{range},n})).$$

-Provide the commitment $C \coloneqq (Z_1, \ldots, Z_n, T)$ and corresponding range proof which is

$$\pi \coloneqq (h_i, \pi_{\text{range},i i \in [n]}, I, \{\sigma_i, r_i\}_{i \in I}).$$

Therefore, the original signer can compute the VT-BLS of $sign_O$ to send the signature to the proxy signer as a verifiable timed signature on the warrant. In the end, we can consider w as a valid warrant value whose signature can only be opened after some pre-defined time T and can be verified.

4. **Proxy Signature Generation:**

Note that, since the warrant is sent in a verifiable manner, the signature of the warrant and the committed shares of its signature can be verified during the verification phase before opening the commitments.

(a) **Verification phase:** $(crs, pk^{'}, m, C, \pi)$ is the input values and the Vrfy algorithm works as follows:

-Define $C \coloneqq (Z_1, \ldots, Z_n, T)$, $\pi \coloneqq (h_i, \pi_{\text{range},i_{i\in[n]}}, I, \{\sigma_i, r_i\}_{i\in I})$ and $crs \coloneqq (crs_{\text{range}}, pp)$.

-If any of the conditions listed below are met, the Vrfy algorithm will produce an output of 0:

i. There is $j \notin I$ satisfying

$$\prod_{i\in I} h_i^{l_i(0)}.h_j^{l_j(0)} \neq pk^{'}.$$

ii. There is $i \in [n]$ satisfying

$$\text{ZKVerify}(crs_{\text{range}}, (Z_i, 0, 2^{\lambda}, T), \pi_{\text{range},i}) \neq 1.$$

iii. There is $i \in I$ satisfying

$$Z_i \neq \text{LHTLP.PuzzleGeneration}(pp, \sigma_i; r_i)$$

or

$$e(g_2, \sigma_i) \neq e(h_i, H(m)).$$

iv. $I \neq H'(pk^{'}, (h_1, Z_1, \pi_{\text{range},1}), \ldots, (h_n, Z_n, \pi_{\text{range},n}))$.

After time T passes, the proxy signer is able to force open the VTS and the proxy key $(sign_O, w)$ can be obtained. There is also another option that the original signer chooses to provide a mechanism that opens all the committed values (Open phase) after time T automatically. This is completely optional and depends on the original signer's choice.

i. **Open phase:** The result of the open phase is opening the commitment and receiving $(sign_O, \{r_i\}_{i\in[n]})$. In this case, the committer is expected to open at least the puzzles for the challenge set I chosen by the verifier.

ii. **ForceOp phase:** This phase takes input $C := (Z_1, \ldots, Z_n, T)$ and:
-Performs $\sigma_i \leftarrow \text{LHTLP.PuzzleSolve}(pp, Z_i)$ for $i \in [n]$ and receives all the signature shares. It should be observed that, given the committer has revealed $t-1$ puzzles, the ForceOp step will only involve solving $(n-t+1)$ puzzles. In [29], the possibility of solving one

puzzle instead of $n-t+1$ puzzles is explained in detail with the help of LHTLP.

-Output $sign_O \coloneqq \prod_{j\in[t]}(\sigma_j)^{l_j(0)}$ by considering the first t signatures shares are valid.

After all the verification of the timed signature is completed, the proxy signer is ready to sign the messages.

Let m be a message to be signed by the proxy signature on behalf of the original signer. The proxy signer first calculates $g_2^{sk_2 \cdot H_3(mk_2)}$, names it as p and makes it public. Then, the proxy signer signs m by computing $sign_P \leftarrow H_0(apk, w, m)^{sk_2 \cdot H_3(mk_2)}$.

(b) **Proxy Signature Verification:**

Given the values $(sign_P, apk, w, m, p)$, the verifier accepts the proxy signature $sign_P$ if the equation $e(sign_P, g_2) = e(H_0(apk, w, m), p)$ holds.

3.1.2 Scenario 2-VAT-PS by time-bounded delegation

In the second scenario, we aim to allow the original signer to delegate their signing rights to a proxy signature for a strictly limited duration in a verifiable and accountable way. Upon expiration of the temporary authorization, the warrant is added to a public revocation list and should no longer be used. This method can be thought of as similar to the concept of certificate transparency in digital certificates. Since the temporary authorization will be public when it is added to the revocation list, it is assumed that the information it contains is not confidential. In addition, since the warrant does not have to contain the duration period for the signing rights, we prevent fraudulent activities over the warrant or the need for a trusted person to check if the warrant contains the intended time limit. The steps are explained as follows:

1. **Key Generation:** Same as Scenario 1.
2. **Setup:** Same as Scenario 1.
3. **(Time-bounded) Delegation:** As a part of the time-bounded delegation process, the original signer generates a warrant value of w. Later, the original

signer calculates $p = g_2^w$ to make it compatible with the VTC and makes it public. In the delegation process, the original signer computes the VTC of the w, publishes it publicly, and sends the original w to the proxy signer from a secure channel as a secret value, which is similar to Scenario 1 but using VTC protocol instead of VTS. Note that, we do not consider how to share the warrant by making it confidential, but we assume that it will be kept secret between the original signer and the proxy signer until time T passes. After the committed time T passed, the warrant value is written in a public revocation list. In this way, the proxy signer will only be eligible to sign messages on behalf of the original signer before the revocation.

4. **Proxy Signature Generation:**

   Since the warrant is published in a verifiable way, the proxy signer can easily verify the committed value with the verification step of VTC. Consider the message m to be signed by the proxy signature, acting on behalf of the original signer. The proxy signer first calculates $p := g_2^{sk_2 \cdot H_3(mk_2)}$, and makes it public. Then, the proxy signer signs m by computing $sign_P \leftarrow H_0(apk, w, m)^{sk_2 \cdot H_3(mk_2)}$.

5. **Proxy Signature Verification:**

   Given the values $(sign_P, apk, w, m, p)$, the verifier accepts the proxy signature $sign_P$ if w is not in the revocation list and the equation $e(sign_P, g_2) = e(H_0(apk, w, m), p)$ holds.

Figure 3.2 shows the high-level construction of the time-bounded (i.e. temporal) delegation of the signature.

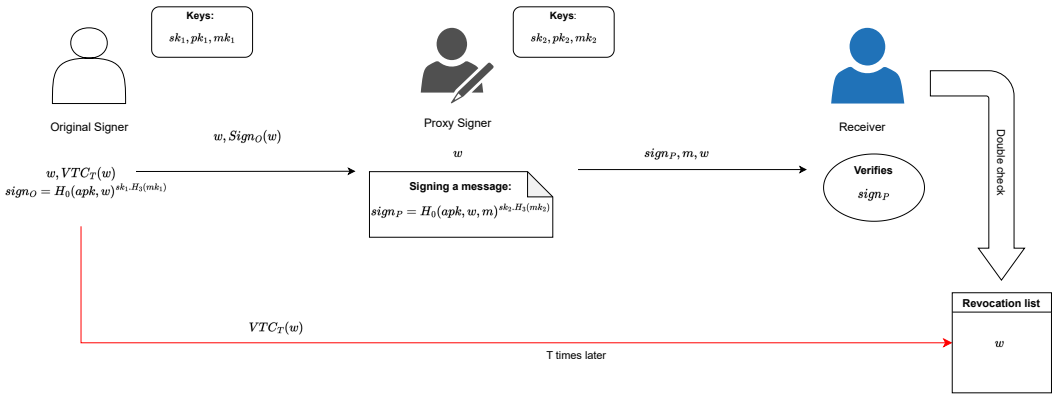

Figure 3.2: VAT-PS by Time-bounded Delegation

3.1.3 Scenario 3-VAT-PS as a timed signature

In the third scenario, we aim to allow the proxy signer to sign documents on behalf of the original signer but also endow them with the ability to apply timed signatures. As a delegation rule, the original signer authorizes the proxy signer, but preconditions that the signature of the message can only be opened after time T. This expectation can be found in the warrant. The steps are explained as follows:

1. **Key Generation:** Same as Scenario 1.

2. **Setup:** Same as Scenario 1.

3. **Delegation:** As a part of the delegation process, the original signer first produces a warrant value of w. Then, the original signer computes $sign_O \leftarrow H_0(apk, w)^{sk_1 \cdot H_3(mk_1)}$. Let $k := sk_1 \cdot H_3(mk_1) \mod q$. Then, k can be considered as the new secret key of the original signer. In this case, $r = g_2^k$ is the new public key of the original signer. The proxy keys to be used during the proxy signing process are the signature of the warrant and the warrant itself.

4. **(Timed) Proxy Signature Generation:**

   The proxy signer needs to first verify that $e(sign_O, g_2) = e(H_0(apk, w), r)$ holds. The proxy signer then calculates $p := g_2^{sk_2 \cdot H_3(mk_2)}$, and makes it public. Then, the proxy signer signs m by computing $sign_P \leftarrow H_0(apk, w, m)^{sk_2 \cdot H_3(mk_2)}$. By using VT-BLS structure for the $sign_P$ under the new secret key $sk_2 \cdot H_3(mk_2)$ and the new public key p, the proxy signer calculates the timed version of its signature by using the VT-BLS process, calls it VAT-PS, and sends it to the receiver.

5. **Proxy Signature Verification:**

   After time T of the VAT-PS passes, the signature can be obtained by ForceOp. Then, given the values $(sign_P, apk, w, m, p)$, the verifier accepts the proxy signature $sign_P$ if the equation $e(sign_P, g_2) = e(H_0(apk, w, m), p)$ holds. Also, since the commitments are designed as verifiable, the whole signature scheme and its committed shares are verifiable.

Figure 3.3 shows the high-level construction of the timed proxy signature construction.

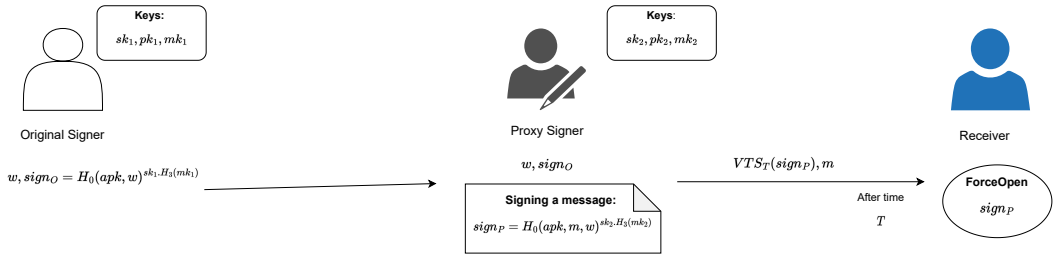

Figure 3.3: VAT-PS as a Timed Signature

3.1.4 Security Analysis of VAT-PS Schemes

We provide the security requirements for our proposals, used for proxy signature schemes and verifiable timed commitments/signatures.

3.1.4.1 VTS/VTC Security:

The security analysis of VTS/VTC is explained here:
Scenario 1:

- **Theorem 1 (Soundness).** It can be inferred that if LHTLP is a time-lock puzzle with perfect correctness, then Scenario 1 outlined in Figure 3.1 meets the soundness requirements set forth in Definition 2, assuming the random oracle model.
- **Theorem 2 (Privacy).** It is asserted that if LHTLP is a secure time-lock puzzle, then Scenario 1 outlined in Figure 3.1 meets the privacy requirements outlined in Definition 3 within the random oracle model.

The proofs can be found in Appendix A.1 and A.2, respectively.

Scenario 2:

- **Theorem 3 (Soundness).** It can be inferred that if LHTLP is a time-lock puzzle with perfect correctness, then Scenario 2 outlined in Figure 3.2 meets the soundness requirements set forth in Definition 4, assuming the random oracle.
- **Theorem 4 (Privacy).** It is asserted that if LHTLP is a secure TLP, then Scenario 2 outlined in Figure 3.2 meets the privacy requirements outlined in Definition 5 within the random oracle model.

The proofs can be found in Appendix A.3 and A.4, respectively.

Scenario 3:

- **Theorem 5 (Soundness).** It can be inferred that if LHTLP is a time-lock puzzle with perfect correctness, then Scenario 3 outlined in Figure 3.3 meets the soundness requirements set forth in Definition 2, assuming the random oracle model.
- **Theorem 6 (Privacy).** It is asserted that if LHTLP is a secure TLP, then Scenario 3 outlined in Figure 3.3 meets the privacy requirements outlined in Definition 3 within the random oracle.

These are very similar to ones given in Appendix A.1 and A.2, respectively.

The theorems play a crucial role in verifying the security of the timed commitments that form the foundation of the proposed schemes. Alternatively, when assessing the proposed schemes based on broader considerations, we can make the following considerations:

3.1.4.2 Verifiability:

Verifiability is one of the most important requirements while designing digital signature schemes. Observe that, since we use verifiable timed commitments in each proposal, the committed values are already verifiable by default verifiability property

of the VTS/VTC. We prove the verifiability of the proxy signature itself as follows:

Scenario 1: Given $(sign_P, apk, w, m, p)$, $sign_P$ should hold the equation $e(sign_P, g_2) = e(H_0(apk, w, m), p)$.

We can show its correctness as follows:

$$\begin{aligned} e(H_0(apk, w, m), p) &= e(H_0(apk, w, m), g_2^{sk_2 \cdot H_3(mk_2)}) \\ &= e(H_0(apk, w, m)^{sk_2 \cdot H_3(mk_2)}, g_2) \\ &= (sign_P, g_2). \end{aligned}$$

Note that, since Scenario 2 and 3 use a similar signing algorithm, their verification can be shown in a similar way.

3.1.4.3 Strong Unforgeability:

In all of the proposed scenarios, we use a group setup process that is used to create membership keys for the original signer and proxy signer. Since the private key of the original signer is used when generating the proxy signer's membership key, and the private key of the proxy signer is used in the signing process, no one other than the proxy signer can generate this signature. Even the original signer cannot create the proxy signature since it requires the private key of the proxy signer.

3.1.4.4 Strong Undeniability:

In all of the proposed scenarios, the warrant w contains the membership keys and determines the involvement of the original and proxy signers. Also, since the aggregated public key apk, which is produced by the public keys of both users, is used to verify the signature, our schemes provide the undeniability property.

3.1.4.5 Strong Identifiability:

All verification processes need apk and the warrant values. Also, the warrant contains the membership key of the proxy signer which provides identifiability.

3.1.4.6 Prevention of Misuse:

Since the warrant contains the information about the message to be signed and the responsibility of the proxy signer, the misuse of the proxy signer is prevented in our schemes. The warrant is used in the verification process, therefore anyone can check if the information is valid or not.

3.1.5 Computational Complexity of VAT-PS Schemes

In this section, we calculate the computational complexity of the proposed VAT-PS algorithms and two recently proposed pairing-based proxy signature schemes [19,31]. Considering that there exists no proxy signature scheme constructed using verifiable timed commitments with the time lock puzzles, the complexity of the commitment creation process counts as an extra cost. The comparison of those algorithms shows that our proposed schemes bring more complexity in terms of computation because of the timed commitment calculation complexity. However, our schemes provide additional transparency and traceability which make them more suitable for some use cases in decentralized infrastructures such as blockchain. In addition, VT-BLS [29] offers the advantages of homomorphism while creating time lock puzzles. This opportunity lets the receiver of the puzzles solve one single puzzle instead of more during the ForceOp phase. Therefore, we believe that our proposal is quite suitable for specific environments. The following notations are used in Table 3.1:

- We calculate the complexity of the delegation, delegation verification, signature generation, and verification steps to compare our results with [19, 31]. Note that, we decided to separate the delegation verification step since it brings additional cost when the delegation values themselves need to be checked.

- We also mention the revocation mechanism used in the proposed schemes and the articles to highlight where the delegation and signing period is given.
- T_H is the number of hash queries for H_0, H_1, H_2, H_3.
- T_{SM} is the time complexity for scalar multiplication.
- T_{MM} is the time complexity for modular multiplication.
- T_{pair} is the time complexity for pairing operation for e.
- T_{Exp} is the time complexity for modular exponentiation.
- T_I is the time complexity for the modular inverse.
- T_{VTS_C} is the time complexity for the VT-BLS commit and prove phase.
- T_{VTS_V} is the time complexity for the VT-BLS verification phase.
- T_{VTC_C} is the time complexity for the VTC commit and prove phase.
- T_{VTC_V} is the time complexity for the VTC verification phase.

3.2 Verifiable Timed Multi-signatures

In this section, we propose timed versions of multi-signature schemes within alternative scenarios. For simplicity, MSP is chosen as the starting point to construct a timed version of a multi-signature. However, timed versions of ASM schemes are the main focus of this work. At first, we present a variant of MSP that requires users to create VTS of their own signatures. Then, we will construct a modified version of the ASM scheme, namely mASM, by slightly modifying Boneh *et al.*'s ASM scheme, which is useful for constructing VTC on membership keys.

3.2.1 Verifiable Timed Multi-signature Protocol (VT-MSP-v1)

The concept revolves around creating a multi-signature on a message m by using VTS. To construct such a scheme, users create their own signatures by MSP, see (2.2)

Table 3.1: Computational complexity of the proposed VAT-PS schemes and recent proposals for pairing-based proxy signature schemes

Proxy Signature Schemes	[19]	[31]	Our Scenario 1	Our Scenario 2	Our Scenario 3
Delegation	$1T_H$	$1T_{SM} + 1T_H$	$1T_{SM} + 2T_H + 1T_{VTS_C}$	$1T_{Exp} + 1T_{VTC_C}$	$1T_{SM} + 2T_H + 1T_{Exp}$
Delegation Verification	$1T_H$	$2T_{Exp} + 1T_H$	$1T_{VTS_V}$	$1T_{VTC_V}$	$2T_{Exp} + 1T_H$
Proxy Signature Generation	$3T_{Exp} + 1T_{MM} + 1T_H + N.T_{SM}$	$1T_{SM} + 1T_I$	$1T_{SM} + 2T_H + 1T_{Exp}$	$1T_{SM} + 2T_H + 1T_{Exp}$	$2T_{SM} + 2T_{Exp} + 3T_H + 1T_{VTS_C}$
Proxy Signature Verification	$5T_{Exp} + 2T_H + N.T_{SM}$	$2T_{Exp} + 1T_{SM} + 1T_H$	$2T_{Exp} + 1T_H$	$2T_{Exp} + 1T_H$	$2T_{Exp} + T_{VTS_V}$
Revocation	By the warrant	By the warrant	By the VTS	By the VTC	By the VTS

and put them in a VTS protocol to make sure that the combiner can only receive the valid signatures after some predefined time T. Note that, since the signature generation for individual users is similar to BLS, VT-MSP-v1 is very similar to VT-BLS except that the initial $[t-1]$ signature parts are assigned as follows:

-$\forall i \in [t-1]$, $\alpha_i \xleftarrow{\$} \mathbb{Z}_q$ and fix $\sigma_i = H(m)^{a_i \cdot \alpha_i}$ and $h_i := g_2^{\alpha_i}$ where $a_i = H_1(pk_i, \{pk_1, \ldots, pk_n\})$. This selection does not change the calculation of the rest of the shares, therefore the rest of the shares are defined as (A.2) and (3.6).

Each user except one can define his/her own time $t \leq T$ to lock the signatures. However, if the requirement is to produce a multi-signature after some predefined time T, at least one user should lock his/her signature with time T. Once the time T has passed, the combiner can obtain all the individual signatures and calculate the multi-signature. In other words, the combiner calculates

$$\sigma = \prod_{j=1}^{n} \sigma_i.$$

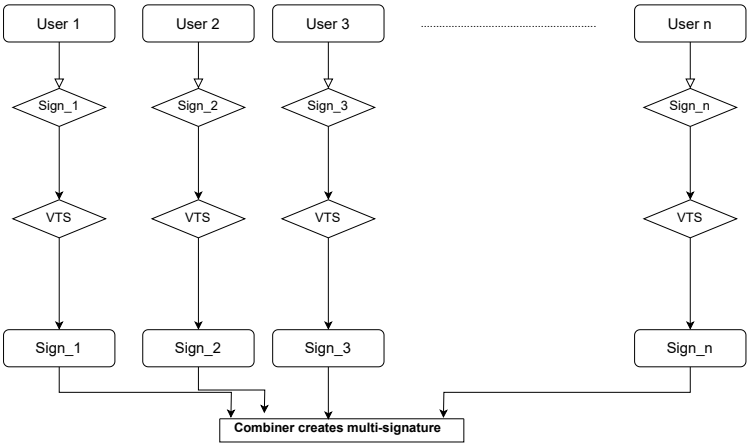

Figure 3.4: VT-MSP-v1

If anyone wants to verify the created multi-signature, they can use the below verification:

$$e(H_0(m), apk) = e(\sigma, g_2). \tag{3.3}$$

In addition, the individual signatures generated can be checked by the VTS verification phase. Thus, it is possible to verify both the individual signatures and the multi-signature created by the combiner using the same public keys. Figure 3.4 shows the high-level construction of VT-MSP-v1.

3.2.1.1 Security of VT-MSP-v1:

The security of VT-MSP-v1 depends on the LHTLP and the unforgeability of the multi-signature as given in the theorems below.

- **Theorem 7 (Soundness).** It can be inferred that if LHTLP guarantees perfect correctness, then VT-MSP-v1 outlined in Figure 3.4 meets the soundness requirements set forth in Definition 2.
- **Theorem 8 (Privacy).** It is asserted that if LHTLP is a secure time-lock puzzle, then VT-MSP-v1 outlined in Figure 3.4 meets the privacy requirements outlined in Definition 3.

The proofs of Theorems 7 and 8 are similar to the ones in Appendix A.1, A.2.

3.2.2 Verifiable Timed Multi-signature Protocol - VT-MSP-v2

Version 2 of VT-MSP is similar to version 1. However, in this version, we would like to use 1 VTS instead of n by making the combiner responsible for the timed signature. Let us consider a payment signed by a company's board of directors that is requested to reach the recipient after time T passes. In such cases, sending the produced multi-signature with the help of timed commitments is beneficial in terms of both verifiability and efficiency.

In this scheme, first, users create their individual signatures using the MSP signature generation scheme. After that, the combiner creates multi-signature by calculating

$$\sigma = \prod_{j=1}^{n} \sigma_i = \prod_{j=1}^{n} H_0(m)^{a_j sk_j} = H_0(m)^{a_1 sk_1 + \ldots + a_n sk_n}. \tag{3.4}$$

Since σ needs to be created as a timed signature, the combiner takes

$$a_1 sk_1 + \ldots + a_n sk_n$$

and calls it sec. After that, the combiner creates VTS of the multi-signature as follows:

- **Setup phase:** Run ZKSetup(1^λ) to generate crs_{range}. Generate public parameters

  $$pp \leftarrow \text{LHTLP.PuzzleSetup}(1^\lambda, T)$$

  and output $crs \coloneqq (crs_{range}, pp)$.

- **Commit:** For input (crs, wit), follow the steps.

  -$wit \coloneqq \sigma$, $crs \coloneqq (crs_{range}, pp)$, pk is the public key value g_2^{sec} created and published by the combiner, m is the message.

  -$\forall i \in [t-1]$, $sec_i \xleftarrow{\$} \mathbb{Z}_q$ and fix $\sigma_i = H(m)^{sec_i}$ and $h_i := g_2^{sec_i}$.

  -$\forall i \in \{t, \ldots, n\}$ compute:

  $$\sigma_i = \left(\frac{\sigma}{\prod_{j \in [t-1]} \sigma_j^{l_j(0)}} \right)^{l_i(0)^{-1}} \tag{3.5}$$

and

$$h_i = \left(\frac{pk}{\prod_{j \in [t-1]} h_j^{l_j(0)}} \right)^{l_i(0)^{-1}} . \tag{3.6}$$

-$\forall i \in [n]$, calculate puzzles and corresponding proofs:

$$r_i \leftarrow \{0,1\}^\lambda, Z_i \leftarrow \text{LHTLP.PuzzleGeneration}(pp, \sigma_i; r_i)$$

and

$$\pi_{range,i} \leftarrow \text{ZKProve}(crs_{range}, (Z_i, 0, 2^\lambda, T), (\sigma_i, r_i)).$$

-Calculate

$$I \leftarrow H'(pk, (h_1, Z_1, \pi_{range,1}), \ldots, (h_n, Z_n, \pi_{range,n})).$$

-Publish the commitment and corresponding range proof which is

$$\pi := (h_i, \pi_{range,i_{i \in [n]}}, I, \{\sigma_i, r_i\}_{i \in I}).$$

- **Vrfy:** (crs, pk, m, C, π) is the input values and the Vrfy algorithm works as follows:

  -Assign $C := (Z_1, \ldots, Z_n, T)$, $\pi := (h_i, \pi_{range,i_{i \in [n]}}, I, \{\sigma_i, r_i\}_{i \in I})$ and $crs := (crs_{range}, pp)$.

  -If any of the below conditions are correct, the Vrfy algorithm outputs 0:

  1. There is $j \notin I$ satisfying

  $$\prod_{i \in I} h_i^{l_i(0)} . h_j^{l_j(0)} \neq pk.$$

  2. There is $i \in [n]$ satisfying

  $$\text{ZKVerify}(crs_{range}, (Z_i, 0, 2^\lambda, T), \pi_{range,i}) \neq 1.$$

  3. There exists $i \in I$ satisfying

  $$Z_i \neq \text{LHTLP.PuzzleGeneration}(pp, \sigma_i; r_i)$$

  or

  $$e(g_2, \sigma_i) \neq e(h_i, H(m)).$$

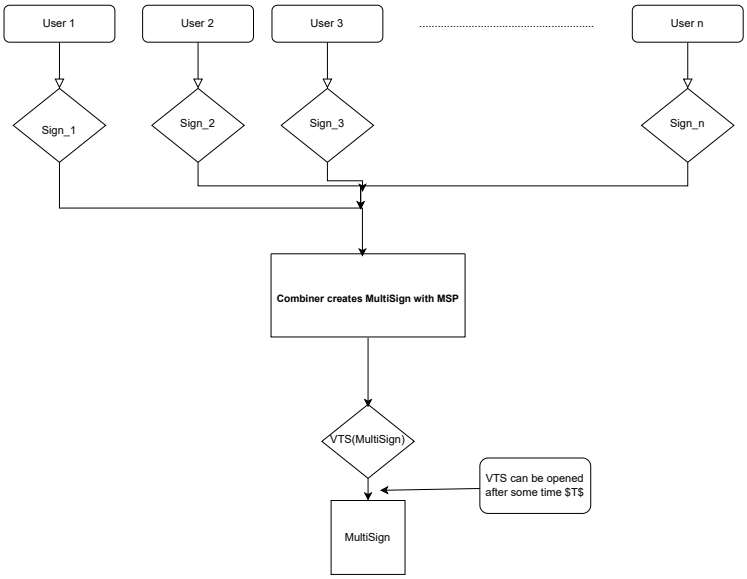

Figure 3.5: VT-MSP-v2

4. $I \neq H'(pk, (h_1, Z_1, \pi_{range,1}), \ldots, (h_n, Z_n, \pi_{range,n}))$.

- **Open:** The combiner is expected to open at least the puzzles for the challenge set I chosen by the verifier. Otherwise, the commitments can be opened within the force open phase similar to VT-BLS.

As it is seen, the combiner can create a timed multi-signature when it is authorized both to create a multi-signature and to time this signature. What should be noted here is that it can be easily verified that the integrity of the whole process is preserved and that the signature has not been changed, even though it seems that the combiner has been given too much authority. Figure 3.5 shows our high-level construction of MSP with VTS as a Version 2.

3.2.2.1 Security of VT-MSP-v2:

The security of VT-MSP-v2 depends on the LHTLP and the unforgeability of the multi-signature as given in the theorems below.

- **Theorem 9 (Soundness).** Assuming the random oracle model, if LHTLP is a time-lock puzzle with perfect correctness, then the VT-MSP-v2 scheme depicted in Figure 3.5 satisfies the soundness criteria outlined in Definition 2.

- **Theorem 10 (Privacy).** If LHTLP is a secure time-lock puzzle, then VT-MSP-v2 outlined in Figure 3.5 meets the privacy requirements outlined in Definition 3.

The proofs of Theorems 9 and 10 are similar to the ones in Appendices A.1, A.2 respectively.

3.2.3 Verifiable Timed Accountable Subgroup Multi-signatures

3.2.3.1 Modified Accountable Subgroup Multi-signatures:

Boneh *et al.* showed that their ASM construction is efficient and applicable in blockchain cryptosystems. In this section, two modified versions of ASM are introduced. The modified versions of the ASM scheme, namely mASM-v1 and mASM-v2, are similar to the ASM scheme until individual signature generation.

mASM-v1:

Individual signatures in the mASM-v1 scheme are calculated as follows:

- SignatureGen: Let H_3 be defined as an additional function that maps elements from $\mathbb{G}_1$ to $\mathbb{Z}_q$. A signer i calculates his own signature as:

$$s_i = H_0(apk, m)^{sk_i \cdot H_3(mk_i)} \tag{3.7}$$

  and delivers s_i and $pk_i^{H_3(mk_i)}$ to the combiner. Note that, the aim of this modification is creating a similar structure with VT-BLS to make it easily adaptable to the timed version.

- Signature Aggregation: The individual signatures let the combiner construct the set of signers $S \subseteq G$. After that, the combiner calculates the ASM $\sigma = (s, pk)$ where $s = \prod_{i \in S} s_i$ and $pk = \prod_{i \in S} pk_i^{H_3(mk_i)}$.

- Verification: With having (par, apk, S, m, σ), anyone can check if:

$$e(H_0(apk, m), pk) = e(s, g_2)$$

- **Theorem 11 (Unforgeability).** mASM-v1 is an unforgeable multi-signature scheme under the computational co-Diffie-Hellman problem in the random-oracle model.

The proof Theorem 11 is similar to the proof of MSP [4].

mASM-v2:

Another modified version of ASM is introduced with authentication of the subgroup. In this case, we want the subgroup to be known from the beginning by creating subgroup-specific membership keys instead of membership keys. To make the scheme simpler, individual signatures can be calculated as follows:

- SignatureGen: $i \in S$ calculates his own signature as:

$$s_i = H_0(apk, m)^{sk_i \cdot H_3(smk_i)}, \tag{3.8}$$

  where smk_i is defined as the subgroup-specific membership key and equal to $\prod_{j \in S} \mu_{ij}$, where μ_{ij} is defined in (2.4). Also, H_3 is a function as defined in mASM-v1. Then, she sends s_i and $pk_i^{H_3(smk_i)}$ to the combiner.

- Signature Aggregation: By utilizing the individual signatures, the combiner can compute the subgroup multi-signature $\sigma = (s, pk)$, where $s = \prod_{i \in S} s_i$ and $pk = \prod_{i \in S} pk_i^{H_3(smk_i)}$.

- Verification: With having $par, apk, pk, S, m, \sigma$, anyone can check if

$$e(H_0(apk, m), pk) = e(s, g_2).$$

- **Theorem 12 (Unforgeability).** mASM-v2 is an unforgeable multi-signature scheme under the computational co-Diffie-Hellman problem in the random-oracle model.

The proof Theorem 12 is similar to the proof of MSP [4].

3.2.3.2 VTC with mASM-v1 (VT-mASM-v1):

Although both mASM modifications proposed are suitable to be timed, we explained the timed mASM-v1 here as an idea, due to the similarities of the schemes. VTC usage in mASM-v1 allows a user to send her membership key to the combiner so that the combiner can only receive individual membership keys after some predefined time T. In this way, membership keys will not be public at the beginning of the protocol but a combiner will be able to use them to construct a multi-signature on behalf of a group of n people. The following steps show how a user sends her mk with VTC. For simplicity, assume that a user has membership key mk and membership key shares are defined as mk_i.

- **Setup phase:** Same as VTC Setup phase.
- **Commit:** For input (crs, wit):

  -Parse $wit := sk \cdot H_3(mk)$, $crs := (crs_{range}, pp)$, $h := g^{sk \cdot H_3(mk)}$.

  -$\forall i \in [t-1]$, $sk'_i \xleftarrow{\$} Z_q$ of the form $sk \cdot H_3(mk)$ and set $h'_i = g^{sk'_i}$.

  -$\forall i \in t, \ldots, n$, compute

$$sk'_i = \left(sk \cdot H_3(mk) - \sum_{j \in [t]} sk'_j . l_j 0 \right) . l_i(0)^{-1}$$

  and

$$h'_i = \left(\frac{h}{\prod_{j \in [t]} h'^{l_j(0)}_j} \right)^{l_i(0)^{-1}},$$

  where l_i is the i-th Lagrange polynomial basis.

  -For $i \in [n]$, calculate puzzles and corresponding range proofs:

$$r_i \leftarrow \{0,1\}^{\lambda}, Z_i \leftarrow \text{LHTLP.PuzzleGeneration}(pp, sk'_i; r_i)$$

  and

$$\pi_{range,i} \leftarrow \text{ZKProve}(crs_{range}, (Z_i, a, b, T), (sk'_i, r_i)).$$

- Set

$$I \leftarrow H'(pk, (h'_1, Z_1, \pi_{range,1}), \ldots, (h'_n, Z_n, \pi_{range,n})).$$

-Output the commitment $C := (Z_1, \ldots, Z_n, T)$ and corresponding range proof which is

$$\pi := (\{h'_i, \pi_{range,i}\}_{i\in[n]}, I, \{sk'_i, r_i\}_{i\in I}).$$

-Final output is (h, C, π).

- **Vrfy:** By using (crs, h, C, π), the Vrfy algorithm works as follows:

  -Let $C \coloneqq (Z_1, \ldots, Z_n, T)$,
  $crs \coloneqq (crs_{range}, pp)$ and $\pi \coloneqq (\{h'_i, \pi_{range,i}\}_{i\in[n]}, I, \{sk'_i, r_i\}_{i\in I})$.

  -If any of the below conditions are correct, the Vrfy algorithm outputs 0. Therefore, the expectation is that these conditions are wrong.

  1. There is $j \notin I$ satisfying

$$\prod_{i\in I} h_i'^{l_i(0)}.h_j'^{l_j(0)} \neq h.$$

  2. There is $i \in [n]$ satisfying

$$\text{ZKVerify}(crs_{range}, (Z_i, a, b, T), \pi_{range,i}) \neq 1.$$

  3. There is $i \in I$ satisfying $Z_i \neq$ LHTLP.PuzzleGeneration$(pp, sk'_i; r_i)$ or $h_i = g^{sk'_i}$.

  4. $I \neq H'(pk, (h'_1, Z_1, \pi_{range,1}), \ldots, (h'_n, Z_n, \pi_{range,n}))$.

- **Open:** The result of the open phase is opening the commitment and receiving $(sk \cdot H_3(mk), \{r_i\}_{i\in[n]}))$. The combiner is expected to open at least the puzzles for the challenge set I chosen by the verifier.

- **ForceOp:** This phase takes $C := (Z_1, \ldots, Z_n, T)$ and then performs the following steps.

  -Calculates $sk'_i \leftarrow$ LHTLP.PuzzleSolve(pp, Z_i) for $i \in [n]$ to retrieve all membership key shares. It should be observed that, given the committer has revealed $t-1$ puzzles, the ForceOp step will only involve solving $(n-t+1)$ puzzles.

  -Publish $sk \cdot H_3(mk) \coloneqq \sum_{j\in[t]}(sk'_j).l_j(0)$ by considering the initial t shares are correct.

In this scenario, a combiner cannot change the membership keys as they are received within a verifiable commitment and are related to the user's secret keys. Also, a combiner cannot create an invalid multi-signature on behalf of a group of n people, as verification of the multi-signature requires aggregated public keys of users which can be verified by any person. Note that, since the combiner needs to choose subgroup S to create mASM, she does not have to calculate all individual signatures of users. She can only calculate the signatures of a subgroup which decreases computational load. Also, since multi-signature is defined as the multiplication of individual signatures in the subgroup, the combiner can first calculate the sum of membership keys of the chosen subgroup and use it to calculate multi-signature. This method can be considered a delegated multi-signature scheme. VT-mASM-v1 can also be reconstructed if the subgroup is known from the beginning of the protocol. Figure 3.6 shows the high-level construction of VT-mASM-v1.

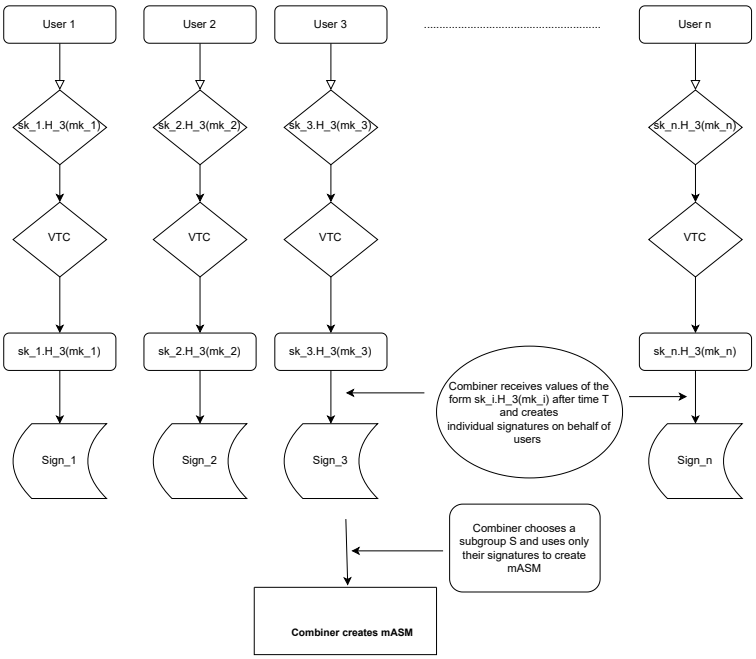

Figure 3.6: VT-mASM-v1

3.2.3.3 Security of VT-mASM-v1:

- **Theorem 13 (Soundness).** If LHTLP is a time-lock puzzle with perfect correctness, then VT-mASM-v1 illustrated in Figure 3.6 meets the soundness re-

quirements set forth in Definition 4, assuming the random oracle model.

- **Theorem 14 (Privacy).** If LHTLP is a secure time-lock puzzle, then VT-mASM-v1 outlined in Figure 3.6 meets the privacy requirements outlined in Definition 5.

The proofs of Theorems 13 and 14 are similar to those given in Appendices A.3, A.4, respectively.

3.2.4 Verifiable Accountable Timed Proxy Multi-signatures (VAT-PMS):

Proxy multi-signature is a cryptographic mechanism that allows a proxy signer to sign messages or transactions on behalf of multiple original signers. Instead of each original signer individually signing a message, the proxy entity holds the necessary private keys and generates a collective signature. In other words, it is similar to proxy signatures, but instead of one original signer, we have more than one original signer. It is particularly useful in scenarios where multiple signers need to delegate their signing power to a trusted intermediary.

A timed version of proxy multi-signatures can be constructed similarly to the proxy signatures. As an example, we construct a timed version of a proxy multi-signature scheme using mASM-v1 suggested above as a multi-signature within Scenario 3 of the proxy signature.

3.2.4.1 VAT-PMS as a timed signature:

In this scheme, we aim to allow the proxy signer to sign documents on behalf of a group of original signers but also endow them with the ability to apply timed signatures. Before the delegation, a warrant is assumed to be created and it needs to be signed by a group of original signers. In this one-time multi-signature creation of the warrant, we choose mASM-v1. While creating mASM over the warrant, wlog, we assume that original signer 1 is the combiner of the mASM who interacts with the proxy signer. As a delegation rule, the original signers authorize the proxy signer, but

preconditions that the signature of the message can only be opened after time T. This expectation can be found in the warrant. The steps are explained as follows:

1. **Key Generation:** Let $o_1, \ldots, o_n$ be the list of n-original signers, and let P be the $n+1$-th signer, who is a proxy signer. The original signers generate their key pair (sk_i, pk_i) and the proxy signer generates his key pair sk_{n+1}, pk_{n+1} with the same key generation procedure as the BLS scheme. For simplicity, we delegate "the combiner rights" to o_1. Therefore, the key pairs of o_1 and P are (sk_1, pk_1) and (sk_{n+1}, pk_{n+1}), respectively.

2. **Setup:** The setup process uses the ASM group setup parameters and functions (H_0, H_1 and H_2 are defined such that H_0 and H_2 map binary strings to $\mathbb{G}_1$ and H_1 maps binary strings to $\mathbb{Z}_q$.) with an additional function defined as $H_3 : \mathbb{G}_1 \rightarrow \mathbb{Z}_q$. Therefore, considering n original signers and one proxy signer in the process, the group setup algorithm works as follows. All users compute the aggregated public key

$$apk \leftarrow \prod_{i=1}^{n+1} pk_i^{H_1(pk_i, \{pk_1, \ldots, pk_{n+1}\})}.$$

   Let $H_1(pk_i, \{pk_1, \ldots, pk_{n+1}\})$ be parsed as a_i. Calculating the membership keys is similar to the ASM, both the original signers and the proxy signer have

$$mk_i \leftarrow \prod_{j=1}^{n+1} \mu_{ij}.$$

3. **Delegation:** As a part of the delegation process, the original signers first produce and agree on a common warrant value of w. After that, n original signers create their own signature $\sigma_i = H_0(apk, w)^{sk_i \cdot H_3(mk_i)}$ and $pk_i^{H_3(mk_i)}$. Then, sends them to the combiner, which is o_1 in our case. o_1 first forms the set of signers $S \subseteq \mathbb{G}$. Then, she computes the aggregated subgroup multi-signature $\sigma = (mASM_O, pk)$ where $mASM_O = \prod_{i \in S} \sigma_i$ and $pk = \prod_{i \in S} pk_i^{H_3(mk_i)}$. Here, $mASM_O$ is the signature of the warrant and it is sent by the o_1 to the proxy signer.

4. **(Verifiable Accountable Timed) Proxy Multi-Signature Creation:**

   The proxy signer needs to first verify that the warrant and its signature are indeed valid, i.e. $e(mASM_O, g_2) = e(H_0(apk, w), pk)$ holds. The proxy signer

then calculates $p := g_2^{sk_{n+1} \cdot H_3(mk_{n+1})}$, and makes it public. Then, the proxy signer signs m by computing

$$sign_P \leftarrow H_0(apk, w, m)^{sk_{n+1} \cdot H_3(mk_{n+1})}.$$

By using VT-BLS structure for the $sign_P$ under the new secret key $sk_{n+1} \cdot H_3(mk_{n+1})$ and the new public key(p) as $g_2^{sk_{n+1} \cdot H_3(mk_{n+1})}$, the proxy signer calculates the timed version of its signature by using VT-BLS process and calls it VAT-PMS and sends it to the receiver. It can be noticed here that the signing process is accountable, delegated to the proxy signer by the original signers, and the proxy multi-signature is created.

5. **(Verifiable Accountable Timed) Proxy Multi-Signature Verification:**

   After time T of the VAT-PMS passed, the signature can be obtained by the open or force open. Then, given the values $(sign_P, apk, w, m, p)$, the verifier accepts the proxy signature $sign_P$ if the equation $e(sign_P, g_2) = e(H_0(apk, w, m), p)$ holds. Also, since the commitments are designed as verifiable, the whole signature scheme and its committed shares are verifiable.

Figure 3.7 shows the high-level construction of the timed proxy signature construction.

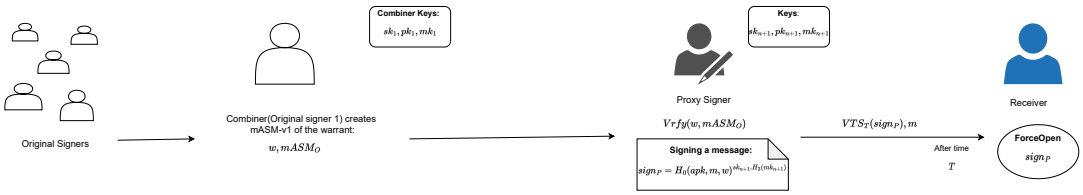

Figure 3.7: VAT-PMS

As seen in VAT-PMS, the mASM creation process is delegated to one of the original signers. Here, it is aimed to make a more efficient design, and the structure in Scenario 3 is tried to be preserved by determining the person in charge to hand the delegation process to the proxy signer. It is possible to construct different schemes where mASM is created by a different combiner (someone not from the original signer group) or other multi-signature protocols can be used except mASM.

3.2.4.2 Security of VAT-PMS:

- **Theorem 15 (Soundness).** If LHTLP is a time-lock puzzle with perfect correctness, then VT-mASM-v1 illustrated in Figure 3.7 meets the soundness requirements set forth in Definition 2, assuming the random oracle model.
- **Theorem 16 (Privacy).** If LHTLP is a secure time-lock puzzle, then VT-mASM-v1 outlined in Figure 3.7 meets the privacy requirements outlined in Definition 3.

The proofs of Theorems 15 and 16 are similar to the ones in Appendices A.1, A.2, respectively.

3.2.5 Performance Evaluation:

In this section, we calculate the time complexity of the VT-BLS algorithm [29] and our proposed VT-based multi-signature schemes. The complexity of VT-BLS was calculated with our notation to understand how much more calculation would be needed to create a verifiable timed multi-signature using the method in a pairing-based verifiable timed signature algorithm. This comparison makes sense since all of our proposed VT-based schemes are also pairing-based and considering pairing is an expensive operation compared to other digital signature schemes (Schnorr, ECDSA, etc.) decreasing the pairing amount is an important improvement. Some operations are assumed to have relatively low computational complexity and are therefore ignored.

Note that, n is the number of users to join multi-signature and s is the subgroup size for ASM. T_H is the total hash queries for H_0, H_1, H_2, H_3. T_G is the time complexity for group set-up queries. T_S is the time complexity for signing queries. T_{pair} is the time complexity for pairing operation for e. T_{Exp_1} is the time complexity for exponentiation in G_1. T_{Exp_2} is the time complexity for exponentiation in G_2. T_{MExp_1} is the time complexity for multi-exponentiation in G_1. T_{MExp_2} is the time complexity for multi-exponentiation in G_2. T_{VTS_C} is the time complexity for VTS Commit and Prove phase. T_{VTS_V} is the time complexity for the VTS Verification phase. T_{VTC_C} is

Table 3.2: Computational complexity of VT-BLS and proposed VT-based multi-signature schemes

VT-BLS [29]	$T_H.T_{Exp_1} + T_{pair} + T_{VTS_C} + T_{VTS_V}$
VT-MSP-v1	$T_H.T_{Exp_1} + T_S.(T_{MExp_2} + T_{Exp_1}) + T_{MExp_2} + T_{pair} + n.(T_{VTS_C} + T_{VTS_V})$
VT-MSP-v2	$T_H.T_{Exp_1} + T_S.(T_{MExp_2} + T_{Exp_1}) + T_{MExp_2} + T_{pair} + (T_{VTS_C} + T_{VTS_V})$
VT-mASM-v1	$T_H.max(nT_{MExp_2} + 2T_{Exp_1}) + T_G.(n-1)T_{Exp_1} + T_S.(T_{MExp_2} + T_{Exp_1}) + T_{pair} + 3T_{MExp_1} + n.(T_{VTC_C} + T_{VTC_V})$
VAT-PMS	$T_H.max(nT_{MExp_2} + 3T_{Exp_1}) + T_G.(n-1)T_{Exp_1} + T_S.(T_{MExp_2} + T_{Exp_1}) + T_{pair} + 3T_{MExp_1} + (T_{VTS_C} + T_{VTS_V})$

the time complexity for VTC Commit and Prove phase. T_{VTC_V} is the time complexity for the VTC Verification phase.

Table 3.2 illustrates the extent to which our constructions introduce additional costs to covert VT-based signatures into a VT-based multi-signature. Considering the amount of puzzle generation, the computationally feasible proposed multi-signature scheme is obviously VT-MSP-v2 since it only uses 1 VTS. VT-MSP-v2 only brings

$$T_S.(T_{MExp_2} + T_{Exp_1}) + T_{MExp_2} + T_{pair}$$

as an additional cost to convert VT-signature (VT-BLS) into VT-multi-signature. VT-MSP-v1 and VT-mASM algorithms are computationally expensive, but they would also be useful in such scenarios where each user wants her signature to be opened after some time t.

CHAPTER 4

VERIFIABLE TIMED COMMITMENTS FOR FAIR SEALED BID AUCTIONS

Ensuring fairness and transparency in sealed bid auctions is crucial for maintaining trust and encouraging participation. This chapter explores the potential of verifiable timed commitments as a solution to address these concerns. By utilizing cryptographic techniques and time-stamping mechanisms, verifiable timed commitments offer a secure and tamper-proof method for participants to submit their bids while keeping the bid values concealed until a predefined deadline. This approach ensures bid privacy and facilitates a fair and accountable auction process. The ability to verify the integrity and timing of commitments empowers both auction organizers and participants to ensure that the auction is conducted in a trustworthy manner.

4.1 Proposed Auction Scheme

In this case, we offer a two-stage auction design. The first part includes the signing of the contract, which includes the rules created by the seller including punishment. The contract is signed by a multi-signature protocol in order to ensure the accountability of all participants, consisting of auctioneers and bidders. The second part is the integration of verifiable timed commitments as a novel approach to enhance the trust, fairness, and efficiency of first-price sealed bid auctions. By incorporating verifiable timed commitments into the auction process, participants will be able to independently verify the validity and integrity of commitments made by other bidders, ensuring a fair and trustworthy auction environment. The approach is conducting

evaluations to assess the trust, verifiability, and practicality of the proposed mechanism.

4.1.1 Stage 1: Signing the contract of the auction

It is assumed that a contract is a document that includes material penal provisions in case of non-compliance and is prepared in accordance with the law upon the seller's request. In this stage, there are mainly three parties: the seller, the auctioneer, and the bidders. Figure 4.1 shows the high-level construction of producing a multi-signature over a contract that contains a set of rules for the auction. Using similar notations in the preliminaries, the process is explained as follows:

1. Assume that the scheme has a trusted infrastructure in terms of the creation of the required parameters.

2. Seller holds an auction contract, namely M, and sends it to the auctioneer. Assume that M is smaller than N where N is the RSA modulus.

3. Auctioneer receives M, calculates RSA keys (e_0, d_0) from the RSA key generation algorithm by considering $N > M$, and signs the contract. $Sign(A) = h^{d_0} \mod N$, where $h = \text{Hash}(M)$. Then, sends the signature to the n-bidders.

4. Each bidder i receives his/her key pair from BLS key generation algorithm $(sk_i, pk_i = g_2^{sk_i})$. A combiner (any bidder) aggregates the public keys and individual signatures are generated as

$$\sigma_i = H_0(Sign(A))^{a_i \cdot sk_i}.$$

5. Seller receives the individual signatures and composes the multi-signature by MSP protocol

$$\sigma = \prod_{j=1}^{n} \sigma_j.$$

6. Verification can be tested by anyone who has $(Sign(A), apk, H_0, \sigma)$ during the auction process by $e(\sigma, g_2^{-1}) \cdot e(H_0(Sign(A), apk)) = 1_{\mathbb{G}_T}$.

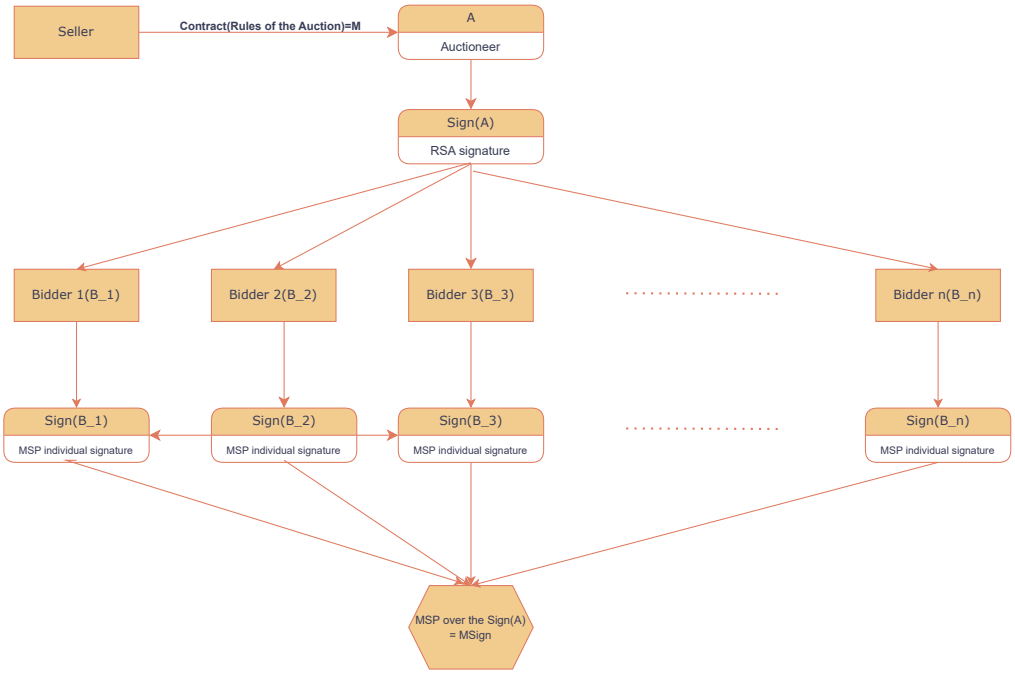

Figure 4.1: Signing the contract

The aim of Stage 1 is to mainly satisfy the accountability of the participants on the auction contract by creating a multi-signature in order to be verified efficiently by someone knowing (par, apk, s_0, σ). The way the multi-signature was created is inspired by a very recent article by Kara *et al.* in [16] both considering efficiency and making the auctioneer dependent on bidders and bidders on auctioneers. The reason we use both RSA signature and MSP together here is to take advantage of both algorithms. More precisely, the use of RSA parameters by the auctioneer allows the same infrastructure to be used both for signing the contract by the auctioneer and for encrypting bids. On the other hand, MSP, in terms of enabling public key aggregation, allows a bidder group to jointly sign a contract even if they do not trust each other.

4.1.2 Stage 2: Bidding phase with VTC

In this stage, there are mainly two parties: the auctioneer, and the bidders. Figure 4.2 shows the high-level construction of producing a VTC over an encrypted bid. Using similar notations in the preliminaries, the process is explained as follows:

1. Each bidder i chooses a bid amount B_i. Assume that $B_i < N$. This is made

possible by the seller setting the maximum amount as well as the minimum amount from the very beginning.

2. Each bidder i uses the RSA public key of the auctioneer (i.e. e_0) to encrypt the bid values

$$E(B_i) = B_i^{e_0} = c_i \mod N.$$

3. Each bidder i produces his/her VTC (commitment and proofs) by considering encrypted bid c as a value to be committed. In other words, each of them performs the following steps:

Setup phase: Same as VTC setup parameters.

Commit and prove phase: For a given (crs, wit), follow the steps:

-$wit \coloneqq c$, $crs \coloneqq (crs_{range}, pp)$, $h \coloneqq g^c$.

-For all $i \in \{1, 2, \ldots, t-1\}$, sample $c_i \leftarrow Z_q$ and fix $h_i = g^{c_i}$.

-For all $i \in \{t, \ldots, n\}$ compute

$$c_i = \left(c - \sum_{j \in [t]} c_j \cdot l_j(0) \right) \cdot l_i(0)^{-1}$$

and

$$h_i = \left(\frac{h}{\prod_{j \in [t]} h_j^{l_j(0)}} \right)^{l_i(0)^{-1}},$$

-For $i \in [n]$, produce puzzles of shares and related proofs as

$$r_i \leftarrow \{0,1\}^\lambda, Z_i \leftarrow \text{LHTLP.PuzzleGeneration}(pp, c_i; r_i)$$

and

$$\pi_{range,i} \leftarrow \text{ZKProve}(crs_{range}, (Z_i, a, b, T), (c_i, r_i)),$$

where $[a, b]$ is the range of the length of the committed value c.

-Calculate

$$I \leftarrow H'(pk, (h_1, Z_1, \pi_{range,1}), \ldots, (h_n, Z_n, \pi_{range,n})),$$

where pk is the public key of the corresponding bidder.

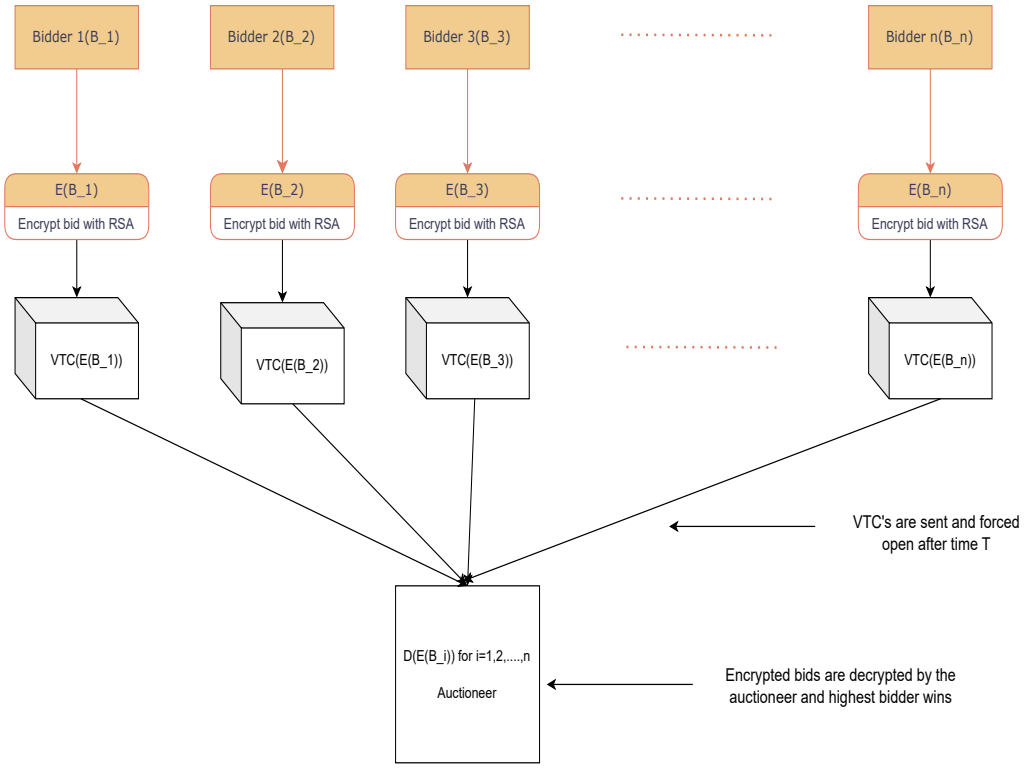

Figure 4.2: Bidding Phase with VTC

-Output the commitment $C := (Z_1, \ldots, Z_n, T)$ and corresponding range proof

$$\pi := (\{h_i, \pi_{range,i}\}_{i\in[n]}, I, \{c_i, r_i\}_{i\in I}).$$

-Final output is (h, C, π).

4. Auctioneer: calculates all bids after time T with the help of the ForceOp algorithm. Then, he compares them to find the maximum bid. He announces the winner to the whole group.

The aim of Stage 2 is to mainly satisfy the verifiability and confidentiality of the bids and to make sure that all the bids are opened after some predefined time T in order to construct a fair auction. Assuming that placing encrypted bids in TLP in order to implement the VTC protocol is mandated by the auction contract, we guarantee that each bidder will keep their bid equally confidential.

Note that there are important points that are valid for both stages and need to be underlined. The first of them is that the message to be signed in the first stage and the bids to be encrypted in the second stage must be less than the selected RSA modulus

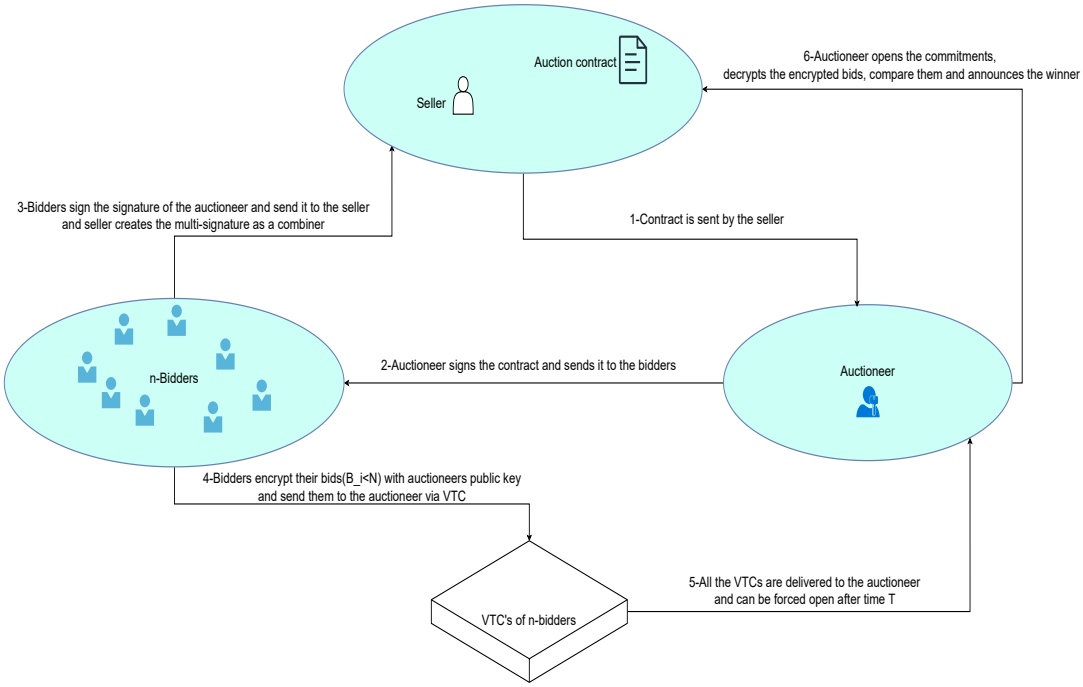

Figure 4.3: Proposed Auction Scheme

N. When holding an auction in this way, this modulus value must be predetermined by a trusted party. In the proposed auction mechanism, we accept that the seller made this choice and that the seller is reliable since he/she is the person who creates the rules in a legal framework and will provide the product. Last, but not least, the primary task of VTCs is to maintain fairness by securely storing bids for time T. Figure 4.3 illustrates the high-level overview of the full proposed auction scheme.

4.2 Security Consideration

In this section, the targeted security analysis for the first and second parts of our hybrid scheme will be discussed. In the first part, the aim is to make the auctioneer and bidders responsible to the seller by producing a multi-signature on an auction contract. Within a multi-signature context, the natural expectation is unforgeability.

In the second part of our hybrid scheme, the aim is to provide soundness and privacy as in [28] of the timed commitments which are used to lock bid shares of the corresponding bids. Other general privacy and security considerations of the auction protocols are defined in Section 2.10.1.

4.2.1 Security analysis

Theorem 17 (Soundness of the VTCs): The conclusion that can be drawn from Stage 2 is that if each LHTLP generated by every bidder adheres to the criteria of being a time-lock puzzle with perfect correctness, then the VTCs produced by the bidders satisfy the soundness requirements within the context of the random oracle model.

Theorem 18 (Privacy of the VTCs): It is asserted in Stage 2 that if each LHTLP is a secure time-lock puzzle, then VTCs produced by the bidders meet the privacy requirements outlined in Definition 5 within the random oracle model.

Theorem 19 (Unforgeability of the multi-signature): Stage 1 of the proposed scheme produces an unforgeable multi-signature scheme under the computational co-Diffie-Hellman problem in the random-oracle model.

The proofs are similar to the proofs given in Appendix.

These theorems are important while proving the security of the cryptographic mechanisms underlying the proposed auction scheme. On the other hand, if we evaluate the proposed scheme in terms of the general auction considerations given in Section 2.10.1, we can consider as follows:

- **Non-repudiation:** Multi-signature scheme is created to provide non-repudiation.
- **Privacy:** All the bids are encrypted by the auctioneer's public key therefore only the auctioneer can open all the bid values. Within the signed contract in the first stage, it is expected to be stated that the auctioneer does not reveal the losing bids. In that case, since the auctioneer can see all the bids after time T, we can say that our proposed scheme satisfies partial privacy.
- **Verifiability:** All the bids are shared by a verifiable timed commitment which can be checked by any participant.
- **Secrecy:** In our scheme, we did not focus on protecting the personal identities of bidders, but we think this can be easily achieved by assigning each VTC

group to a different user identifier.

- **Robustness:** In the first stage, we try to maintain the integrity of the auctioneer and bidders by signing the contract with a joint multi-signature. In the second stage, we aim to detect any abuse and apply the rules in the contract in case of any abuse, with verifiable commitments.

- **Fairness:** Let's say that after the auctioneer calculates the highest bidding amount, he/she announces that amount by increasing it in favor of a pre-arranged bidder, declaring that the pre-arranged bidder is the winner. Even if he/she does this, the seller or any participant can encrypt the announced value with the auctioneer's public key and publicly solve the VTC in question, compare the output value with the encrypted value, and notice the change. In this way, a reliable and fair system is established for both bidders and auctioneers.

- **Anonymity:** In our scheme, we did not focus on protecting anonymity.

CHAPTER 5

CONCLUSION

This chapter summarizes the findings and contributions of the thesis on verifiable timed commitments and suggests future research directions to improve the efficiency and scalability of the proposed timed signature schemes.

The thesis primarily focused on novel applications of verifiable timed commitment schemes within different scenarios. In Chapter 3, our target was to design a timed signature scheme by taking into account the efficient implementations of verifiable timed commitments. In Section 3.1, we wanted to address the time-constraint requirements of proxy signatures, focusing on the delegation or signing stages. We considered different scenarios such as granting limited-time authority during the delegation process, enabling the authority to become usable after a specific time, and allowing the proxy signature to be opened only after a certain period has passed. Furthermore, the evaluation process of these scenarios in terms of satisfying various security requirements of proxy signatures was described. It was demonstrated that all three proposed schemes fulfill the security properties expected in proxy signatures. A comparison was made between the proposed usage scenarios and existing proxy signature schemes in terms of computational complexity. Given that there is currently no proxy signature scheme that utilizes verifiable timed commitments with time lock puzzles, the computational complexity of the commitment creation process can be seen as an additional cost. When comparing these algorithms, it becomes evident that our proposed schemes introduce more computational complexity due to the cost of timed commitment calculations. However, our schemes offer added transparency and traceability, making them well-suited for specific use cases in decentralized infras-

tructures like blockchain.

In Section 3.2, we focused on transforming multi-signatures into timed multi-signatures. There were two significant approaches to achieve this. The first approach involved incorporating the individual signatures of signers into a timed commitment structure. The second approach involved adapting the overall multi-signature structure created by the combiner into a timed commitment framework. In this regard, we prioritized the transformation of efficient algorithms proposed for blockchain structures like MSP and ASM [4] into timed versions. When considering the effective use cases of these algorithms, it was observed that they are well-suited for the timed signature structure and they present a novel approach to generating proxy multi-signatures. By following this approach, in Section 3.2.4, we presented a compact scheme, named a verifiable accountable timed proxy multi-signature scheme(VAT-PMS) that encompasses the proposed timed versions of signatures and multi-signatures. In Table 3.2, we demonstrated the additional computational overhead required for pairing-based timed signature structures to be transformed into multi-signatures, particularly timed multi-signatures. During this process, we showed that among our proposed timed multi-signature solutions, VT-MSP-v2 exhibits the highest computational efficiency.

On the other hand, in Chapter 4, we focused on designing an end-to-end auction system using verifiable timed commitments. We constructed this framework by combining two stages. The first stage involves the establishment of rules and the mutual signing of a contract by bidders and the auctioneer using a multi-signature. The second stage focuses on the bidding process, emphasizing the verifiability of bids. These stages form a hybrid framework that incorporates both aspects. To ensure the fulfillment of the expected conditions while designing a secure auction, we utilized verifiable timed commitments in this process. Security proofs of all proposed algorithms and structures are given in the appendix.

As future work, it is important to conduct additional research on improving the efficiency of the schemes. Reducing the computational costs would greatly enhance their practical viability. In addition, assessing the scalability of the proposed approach to conduct large-scale auctions with many participants and the effect of timed commitments would be an interesting study.

REFERENCES

[1] R. Alvarez and M. Nojoumian, Comprehensive survey on privacy-preserving protocols for sealed-bid auctions, Computers & Security, 88, p. 101502, 2020.

[2] M. R. Asaar, M. Salmasizadeh, and W. Susilo, A short id-based proxy signature scheme, International Journal of Communication Systems, 29(5), pp. 859–873, 2016.

[3] E.-O. Blass and F. Kerschbaum, Strain: A secure auction for blockchains, in *Computer Security: 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I 23*, pp. 87–110, Springer, 2018.

[4] D. Boneh, M. Drijvers, and G. Neven, Compact multi-signatures for smaller blockchains, in *Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part II*, pp. 435–464, Springer, 2018.

[5] D. Boneh, B. Lynn, and H. Shacham, Short signatures from the weil pairing, in *Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7*, pp. 514–532, Springer, 2001.

[6] D. Boneh and M. Naor, Timed commitments, in *Annual international cryptology conference*, pp. 236–254, Springer, 2000.

[7] J.-S. Chou, A novel anonymous proxy signature scheme, Advances in Multimedia, 2012, pp. 13–13, 2012.

[8] A. De Santis, S. Micali, and G. Persiano, Non-interactive zero-knowledge proof systems, in *Advances in Cryptology—CRYPTO'87: Proceedings 7*, pp. 52–72, Springer, 1988.

[9] H. Desai, M. Kantarcioglu, and L. Kagal, A hybrid blockchain architecture for privacy-enabled and accountable auctions, in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 34–43, IEEE, 2019.

[10] H. Du and J. Wang, An anonymous but accountable proxy multi-signature scheme., J. Softw., 8(8), pp. 1867–1874, 2013.

[11] A. Fiat and A. Shamir, How to prove yourself: Practical solutions to identification and signature problems, in *Advances in Cryptology—CRYPTO'86: Proceedings 6*, pp. 186–194, Springer, 1987.

[12] M. Fischlin and A. Mittelbach, An overview of the hybrid argument, Cryptology ePrint Archive, 2021.

[13] H. S. Galal and A. M. Youssef, Verifiable sealed-bid auction on the ethereum blockchain, in *Financial Cryptography and Data Security: FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers 22*, pp. 265–278, Springer, 2019.

[14] W. Gao, W. Yu, F. Liang, W. G. Hatcher, and C. Lu, Privacy-preserving auction for big data trading using homomorphic encryption, IEEE Transactions on Network Science and Engineering, 7(2), pp. 776–791, 2018.

[15] D. Johnson, A. Menezes, and S. Vanstone, The elliptic curve digital signature algorithm (ecdsa), International journal of information security, 1, pp. 36–63, 2001.

[16] M. Kara, A. Laouid, and M. Hammoudeh, An efficient multi-signature scheme for blockchain, Cryptology ePrint Archive, 2023.

[17] B. Lee, H. Kim, and K. Kim, Strong proxy signature and its applications, in *Proceedings of SCIS*, volume 2001, pp. 603–608, 2001.

[18] J. Li, L. Xu, and Y. Zhang, Provably secure certificate-based proxy signature schemes., J. Comput., 4(6), pp. 444–452, 2009.

[19] Q. Lin, J. Li, Z. Huang, W. Chen, and J. Shen, A short linearly homomorphic proxy signature scheme, IEEE Access, 6, pp. 12966–12972, 2018.

[20] G. Malavolta and S. A. K. Thyagarajan, Homomorphic time-lock puzzles and applications, in *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part I*, pp. 620–649, Springer, 2019.

[21] M. Mambo, K. Usuda, and E. Okamoto, Proxy signatures: Delegation of the power to sign messages, IEICE transactions on fundamentals of electronics, communications and computer sciences, 79(9), pp. 1338–1354, 1996.

[22] S. Micali, K. Ohta, and L. Reyzin, Accountable-subgroup multisignatures, in *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pp. 245–254, 2001.

[23] M. Naor, B. Pinkas, and R. Sumner, Privacy preserving auctions and mechanism design, in *Proceedings of the 1st ACM Conference on Electronic Commerce*, pp. 129–139, 1999.

[24] R. L. Rivest, A. Shamir, and D. A. Wagner, Time-lock puzzles and timed-release crypto, 1996.

[25] R. A. Sahu and S. Padhye, Identity-based multi-proxy multi-signature scheme provably secure in random oracle model, Transactions on Emerging Telecommunications Technologies, 26(4), pp. 547–558, 2015.

[26] C.-P. Schnorr, Efficient identification and signatures for smart cards, in *Advances in Cryptology—CRYPTO'89 Proceedings 9*, pp. 239–252, Springer, 1990.

[27] J. Sun and H. Ma, Privacy-preserving verifiable incentive mechanism for online crowdsourcing markets, in *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–8, IEEE, 2014.

[28] S. A. Thyagarajan, G. Malavolta, and P. Moreno-Sanchez, Universal atomic swaps: Secure exchange of coins across all blockchains, in *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 1299–1316, IEEE, 2022.

[29] S. A. K. Thyagarajan, A. Bhat, G. Malavolta, N. Döttling, A. Kate, and D. Schröder, Verifiable timed signatures made practical, in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1733–1750, 2020.

[30] D. Unruh, Random oracles and auxiliary input, in *Advances in Cryptology-CRYPTO 2007: 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007. Proceedings 27*, pp. 205–223, Springer, 2007.

[31] G. K. Verma and B. Singh, Short certificate-based proxy signature scheme from pairings, Transactions on Emerging Telecommunications Technologies, 28(12), p. e3214, 2017.

[32] K. Zhang, Threshold proxy signature schemes, in *Information Security: First International Workshop, ISW'97 Tatsunokuchi, Ishikawa, Japan September 17–19, 1997 Proceedings 1*, pp. 282–290, Springer, 1998.

APPENDIX A

PROOFS OF THE THEOREMS

In this section, proofs of the theorems used in the above-mentioned and proposed diagrams are given.

A.1 Proof of Theorem 1

Proof. In this context, we are focusing on the interactive version of our protocol, and it is worth noting that the soundness of the non-interactive protocol can be inferred from [11]. Let A be an attacker who breaks the sound by producing the commitment $C = (Z_1, \ldots, Z_n)$ such that $\forall Z_i \notin I$, satisfies LHTLP.PuzzleSolve$(pp, Z_i) = \tilde{\sigma}_i$ where $e(g_2, \tilde{\sigma}_i) \neq e(h_i, H(w))$.

Let us suppose the opposite is true. In that case, it would be possible to recover a legitimate signature on the message w by interpolating $\tilde{\sigma}_i$ with $\sigma_{ii \in I}$, which meets the aforementioned inequality condition. Also, let Z_is be well-formed. In this case, with given Z_i values, through the process of solving the puzzles and confirming which signature shares meet the inequality relation, we can obtain a polynomial-time solution for some set $I^{'}$. When $I^{'} = I$, the verifier can accept the statement provided by the prover, indicating that the prover accurately guessed a randomly chosen n-bit string with $n/2$-many 0's. This situation could happen with a probability $\frac{((n/2)!)^2}{n!}$. In the non-interactive type of our protocol, this statement remains true in the face of any number of simulated proofs, so long as the NIZK has the property of simulation-soundness. Therefore, starting with a simulation-sound NIZK makes our scheme simulation-sound as well. □

In addition, the probabilistic condition in definition 2 tells us that although we can verify the verification steps in VTS, there is a negligible possibility of obtaining a signature that cannot be verified. This definition matches the proof given above.

Note that, this proof is similar to the proof of Theorem 5, 7, 10, and 16 as they are also constructed with VTS.

A.2 Proof of Theorem 2

Proof. Considering all VTS operations are identical for any time value t, it is enough to prove only one of them. Assume that A is an adversary of depth bounded by T^{ϵ}, where $0 \leq \epsilon < 1$ and T is the predefined time for the puzzles. We use the hybrid argument method [12] to construct a series of hybrids that is similar to VT-BLS privacy proof [29].

- Hybrid $\mathcal{H}_0$: Same as the original execution.
- Hybrid $\mathcal{H}_1$: In this case, the random oracle is created by using lazy sampling, which distinguishes it from $\mathcal{H}_0$ [30]. Moreover, a set I^*, comprising $t-1$ elements, is selected beforehand and is used in the cut-and-choose step.
- Hybrid $\mathcal{H}_2$: We sample a simulated crs_{range}. As it is used in the Zero-Knowledge proof setup (in section 2.8), changing crs_{range} is indistinguishable.
- Hybrid $\mathcal{H}_3 \ldots \mathcal{H}_{3+n}$: $\forall i \in [n]$, the proof $\pi_{range,i}$ is calculated by the simulator provided by the NIZK proof in the hybrid $\mathcal{H}_{3+i}$. As it is used in the Zero-Knowledge setup, the distance between all the hybrids is negligibly small.
- Hybrid $\mathcal{H}_{3+n+1} \ldots \mathcal{H}_{3+2n-t+1}$: $\forall i \in [n-(t-1)]$, the puzzle of the i-th value of the complement of the set I^* is calculated by

  $$\text{LHTLP.PuzzleGeneration}(pp, 0^{\lambda}; r_i).$$

  In other words, we use 0^{λ} for all the σ_i values within the puzzle generation algorithm. The A is depth-bounded and therefore indistinguishability is based on the security of the puzzle.

- Hybrid $\mathcal{H}_{3+2n-t+2}$: The prover computes cut-and-choose protocol and corresponding puzzles by choosing the first $t-1$ share $i \in I^*$. For all $i \in I^*$, it samples a uniform $k_i \leftarrow \mathbb{Z}_q$ and sets $pk_i^{'} = g^{k_i}$ and computes the corresponding puzzles as in the VTS algorithm. For all $i \notin I^*$, it computes $pk_i^{'}$ as:

$$pk_i^{'} = \left(\frac{pk^{'}}{\prod_{j \in I^*} h_j^{l_j(0)}} \right)^{l_i(0)^{-1}} \tag{A.1}$$

The rest is not changed. We also know that, for all $i \notin I^*$,

$$\prod_{j \in I^*} h_j^{l_j(0)} . h_i^{l_i(0)} = pk^{'}.$$

Here, the simulator S is the same as the last hybrid. There is no information processed about the witness who sees the correct value of σ. Below, the parts that have changed until the final hybrid are highlighted. As can be seen, no information regarding σ was used:

1. **Setup phase:** Run ZKSetup(1^λ) to generate $\mathbf{crs}_{\text{range}}$. Generate public parameters

$$pp \leftarrow \text{LHTLP.PuzzleSetup}(1^\lambda, T)$$

and output $crs \coloneqq (crs_{\text{range}}, pp)$.

2. **Commit and prove phase:** For input (crs, wit), follow the steps below:

-$crs \coloneqq (crs_{range}, pp)$, $pk^{'}$ is the public key generated as in the BLS signature scheme, and (apk, w) is used as the message.

-For all** $i \in I^*$**, sample** $\alpha_i \leftarrow \mathbb{Z}_q$ **and fix** $\sigma_i = H(m)^{\alpha_i}$ **and** $h_i \coloneqq g_2^{\alpha_i}$**.

-For all $i \in \{t, \ldots, n\}$ compute:

$$\sigma_i = \left(\frac{\sigma}{\prod_{j \in [t-1]} \sigma_j^{l_j(0)}} \right)^{l_i(0)^{-1}} \tag{A.2}$$

and

$\mathbf{h_i} = \left(\frac{\mathbf{pk}^{'}}{\prod_{\mathbf{j} \in \mathbf{I}^*} \mathbf{h_j^{l_j(0)}}} \right)^{\mathbf{l_i(0)^{-1}}}$

-For $i \in [n]$, generate puzzles and proofs

$r_i \leftarrow \{0,1\}^{\lambda}$, $\mathbf{Z_i} \leftarrow$ LHTLP.PuzzleGeneration$(\mathbf{pp}, \mathbf{0}^{\lambda}; \mathbf{r_i})$ and

$\pi_{\mathbf{range,i}}$ **is computed via the simulator.**

Random oracle is (lazy)sampled, I^* ***is also pre-sampled. The output of the random oracle is deliberately configured to be*** I^* ***for the cut-and-choose instance.***

-Calculate $I^* \leftarrow H'^*(pk', (h_1, Z_1, \pi_{\textbf{range},1}), \ldots, (h_n, Z_n, \pi_{\textbf{range},n}))$.
-Produce the commitment $C \coloneqq (Z_1, \ldots, Z_n, T)$ and corresponding range proof which is

$$\pi \coloneqq (h_i, \pi_{\text{range},i_{i\in[n]}}, I^*, \{\sigma_i, r_i\}_{i\in I^*}).$$

Therefore, the algorithm is private against the adversary A.

□

Note that, this proof is similar to the proof of Theorem 6, 8, 11, and 17 as they are also constructed with VTS.

A.3 Proof of Theorem 3

Proof. In this context, we are focusing on the interactive version of our protocol, and it is worth noting that the soundness of the non-interactive protocol can be inferred from [11]. Let A be an attacker who breaks the sound by producing the commitment $C = (Z_1, \ldots, Z_n)$ such that $\forall Z_i \notin I$, satisfies LHTLP.PuzzleSolve$(pp, Z_i) = \tilde{w_i}$ where $h_i \neq g^{\tilde{w_i}}$.

Let us suppose the opposite is true. In that case, it would be possible to recover a legitimate warrant value by interpolating $\tilde{w_i}$ with $\{w_i\}_{i\in I}$, which meets the aforementioned inequality condition. Also, let Z_is be well-formed. In this case, with given Z_i values, through the process of solving the puzzles and confirming which shares meet the inequality relation, we can obtain a polynomial-time solution for the set I'. When $I' = I$, the verifier can accept the statement provided by the prover, indicating that the

prover accurately guessed a randomly chosen n-bit string with $n/2$-many 0's. This situation could happen with a probability $\frac{((n/2)!)^2}{n!}$. In the non-interactive variant of our protocol, the above statement remains true in the face of any number of simulated proofs, so long as the NIZK has the property of simulation-soundness. Therefore, starting with a simulation-sound NIZK makes our scheme simulation-sound as well, similar to Theorem 1. □

In addition, the probabilistic condition in Definition 4 tells us that although we can verify the verification steps in VTC, there is a negligible possibility of obtaining a warrant that cannot be verified. This definition matches the proof given above.

Note that, this proof is similar to the proof of Theorem 13 and 19 as they are also constructed with VTC.

A.4 Proof of Theorem 4

Proof. Considering all VTC operations are identical for any time value t, it is enough to prove only one of them. Assume that A is an adversary of depth bounded by T^{ϵ}, where $0 \leq \epsilon < 1$ and T is the predefined time for the puzzles. We construct a series of hybrids to be used in the simulator S.

- Hybrid $\mathcal{H}_0$: Same as the original execution.
- Hybrid $\mathcal{H}_1$: In this case, the random oracle is created by using lazy sampling, which distinguishes it from $\mathcal{H}_0$ [30]. Moreover, a set I^*, comprising $t-1$ elements, is selected beforehand and is used in the cut-and-choose step.
- Hybrid $\mathcal{H}_2$: We sample a simulated crs_{range}. As it is used in the Zero-Knowledge proof setup (in section 2.8), changing crs_{range} is indistinguishable.
- Hybrid $\mathcal{H}_3 \ldots \mathcal{H}_{3+n}$: $\forall i \in [n]$, the proof $\pi_{range,i}$ is calculated by the simulator in the hybrid $\mathcal{H}_{3+i}$. As it is used in the Zero-Knowledge setup, the distance between all the hybrids is negligibly small.

- Hybrid $\mathcal{H}_{3+n+1} \ldots \mathcal{H}_{3+2n-t+1} : \forall i \in [n-(t-1)]$, the puzzle of the i-th value of the complement of the set I^* is calculated by

$$\text{LHTLP.PuzzleGeneration}(pp, 0^\lambda; r_i).$$

  In other words, we use 0^λ for all the w_i values within the puzzle generation algorithm. The A is depth-bounded and therefore indistinguishability is based on the security of the puzzle.

- Hybrid $\mathcal{H}_{3+2n-t+2}$: The prover computes cut-and-choose protocol and corresponding puzzles by choosing the first $t-1$ share $i \in I^*$. For all $i \in I^*$, it samples a uniform $w_i \leftarrow \mathbb{Z}_q$ and sets $p_i = g_2^{w_i}$ and computes the corresponding puzzles as in the VTC algorithm. For all $i \notin I^*$, it computes p_i as:

$$p_i = \left(\frac{p}{\prod_{j \in I^*} p_j^{l_j(0)}} \right)^{l_i(0)^{-1}} \tag{A.3}$$

  The rest is not changed. We also know that, for all $i \notin I^*$,

$$\prod_{j \in I^*} p_j^{l_j(0)} . p_i^{l_i(0)} = p.$$

Here, the simulator S is the same as the last hybrid. There is no information processed about the witness who sees the correct value of the warrant w. Therefore, the algorithm is private against the adversary A.

□

Note that, this proof is similar to the proof of Theorem 14 and 20 as they are also constructed with VTC.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Özden, Duygu
Nationality: Turkish

PUBLICATIONS

Conference

Duygu Özden, Oğuz Yayla. Verifiable Timed Commitments for Fair Sealed-bid Auctions, *International Conference on Cryptography, Informatics and Cybersecurity*, Bogor, Indonesia, 2023

Duygu Özden, Oğuz Yayla. Verifiable Timed Accountable Subgroup Multi-signatures, 2023. *Submitted*

Journal

Duygu Özden, Oğuz Yayla. Verifiable Timed Proxy Signatures and Multi-signatures. July 2023. *Submitted*