# Non-mallable Zero-knowledge Primitives

*Xuhan Zhang*


4th Year Project Report
Computer Science and Mathematics
School of Informatics
University of Edinburgh

2024

# Abstract

This dissertation introduces a significant advancement in the field of non-malleable zero-knowledge (NMZK) proofs by addressing a longstanding challenge in cryptographic research. Prior to this work, Kim et al. [30] proposed the first non-black-box NMZK protocol, $\Pi_{KLP}$, capable of practical instantiation in 2022. By using the new technique, their work was only based on symmetric primitives. However, the feasibility of achieving NMZK strictly through the black-box use of underlying primitives remained an unresolved question, regardless of the constraints on the number of rounds and the actual efficiency.

This dissertation presents a novel development in this area by discussing the first black-box NMZK protocol, $\Pi_{NM}$, proposed by Dr. Michele Ciampi et al. [12]. Remarkably, this protocol achieves optimality in its use of primitives, requiring only one-way functions, and it is asymptotically optimal regarding its round complexity, merely requiring a constant number of rounds (specifically, only 9 rounds needed). Moreover, $\Pi_{NM}$ can be efficiently instantiated in Minicrypt, providing strong foundations for practical application.

This project practically implements $\Pi_{NM}$, translating theoretical constructions into functional realities. The dissertation first discusses the structure of $\Pi_{NM}$ and any relevant background information and then delves into the specific implementation decisions made during the protocol's development. It provides practical validations for the theoretical work of $\Pi_{NM}$ through comprehensive benchmark testing, it is demonstrated that $\Pi_{NM}$ significantly outperforms $\Pi_{KLP}$ both in terms of communication complexity and execution time.

# Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

# Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Xuhan Zhang*)

# Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor, Dr. Michele Ciampi, for the continuous support of my undergraduate thesis, and for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this dissertation.

I also want to thank my second marker, Dr. Bjorn Ross, for his insightful comments and encouragement during our meeting at the end of the first semester, it helps a lot in writing the final dissertation.

I would like to thank my teammate, Ethan Li, who helped me a lot with the programming language used, Rust, which was brand new to me before this project. Ethan's expertise in Rust and knowledge-sharing were very important in our success.

Finally, a special thanks to my family. Words cannot express how grateful I am to my mother, for all of the sacrifices she has made and all the mental support she provided all the time.

Thank you!

# Table of Contents


**1 Introduction** **1**
- 1.1 Related Work . . . 3

**2 Preliminary** **5**
- 2.1 Commitment Scheme . . . 5
- 2.2 Extractable Commitment . . . 6
- 2.3 Non-malleable Commitment . . . 7
- 2.4 Arguments/Proofs . . . 8
- 2.5 Σ-Protocols . . . 10
- 2.6 Non-malleable Interactive Arguments/Proofs . . . 11

**3 Overview of the Protocol $\Pi_{NM}$** **12**
- 3.1 The Naive Protocol and Limitation . . . 12
- 3.2 Achieving Non-malleability with $\Pi^{3R}_{BGRRV}$ . . . 13
- 3.3 Making $\Pi^{3R}_{BGRRV}$ Extractable . . . 14
- 3.4 Final Protocol $\Pi_{NM}$ . . . 15

**4 Overview and Implementation Details of Sub-protocols** **16**
- 4.1 Extractable Commitment $\Pi_{3Ext}$ . . . 16
- 4.2 Σ-protocol . . . 19
  - 4.2.1 Schnorr's Protocol . . . 19
  - 4.2.2 ZKBoo++ . . . 20

**5 Results and Discussion** **23**
- 5.1 Results Comparison . . . 23
- 5.2 Efficiency Analysis . . . 25

**6 Conclusion and Future Work** **27**

**A Details of Benchmark Tests** **32**

**B Optimisation on ZKBoo++ over ZKBoo** **34**

# Chapter 1

# Introduction

Zero-knowledge proof, introduced by Goldwasser, Micali and Rackoff [20], is a fascinating and important concept in the field of cryptography, that offers a powerful method for ensuring privacy and security in digital communications. At their core, zero-knowledge proofs allow a prover to convince a verifier of the validity of a statement without disclosing any specific details about the statement itself. This paradoxical-sounding technique employs mathematical algorithms to achieve a balance between transparency and confidentiality. By allowing the prover to demonstrate knowledge of a secret (like a password or a cryptographic key) without actually disclosing the secret itself, zero-knowledge proofs enable a new level of security in digital transactions, safeguarding sensitive information from exposure while still providing authenticity. This innovative approach has wide-ranging applications, from blockchain technologies [17, 40] to secure voting systems [36].

Non-Malleable Zero Knowledge (NMZK), introduced by Dolev, Dwork, and Naor [15], enhances the concept of standard zero-knowledge by ensuring that proofs cannot be tampered with or transformed into valid proofs for other (potentially false) statements, safeguarding the privacy and integrity of information exchanged between parties. Although NMZK is primarily designed to defend against Man-in-the-Middle (MIM) attacks, where an adversary intercepts and possibly alters communications between two unaware parties, it also serves as a foundational building block for more complex secure computation protocols. For instance, the concept of NMZK has been pivotal in developing protocols resistant to concurrent attacks, as demonstrated in various studies [8, 9, 10].

When it comes to the use of underlying cryptographic primitives, there are two distinct approaches: the black-box and the non-black-box. The black-box approach focuses solely on the input/output behavior of the primitives, in contrast to the non-black-box approach, which utilises the code that computes the primitives [29]. In this project, we focus on the black-box methodology due to both theoretical interest and practical considerations. The independence from the complexity of the underlying primitives makes the black-box approach more suitable for practical applications. Additionally, from the practical point of view, efficiency concerns also favour the black-box approach. Non-black-box constructions tend to suffer from reduced efficiency due to the use

of the general NP reductions to prove statements in zero-knowledge, affecting both computational and communication complexities [38]. Despite its practical advantages, the black-box approach does encounter limitations when compared to non-black-box constructions for numerous cryptographic tasks. Often, translating non-black-box constructions into practical black-box implementations is not straightforward and requires extra resources to employ either specific or additional general assumptions [38]. This challenge can be found in various studies [26, 27, 28] that utilize primitives in a black-box manner. To bridge the gap between black-box and non-black-box constructions, these protocols typically need extra rounds of interaction, making them less efficient than their non-black-box counterparts.

Despite the feasibility of achieving zero-knowledge argument systems in as few as four rounds using a black-box approach [29], a significant challenge remains in achieving non-malleability. Interestingly, the landscape changes when focusing solely on non-malleable commitment schemes. Non-malleable commitments that employ a black-box use of one-way functions (OWFs) can be established in a constant number of rounds [23]. Moreover, with the black-box use of one-to-one one-way functions, this can be reduced to just 4 rounds [13]. Yet, the construction of non-malleable zero-knowledge (NMZK) proofs from non-malleable commitments using a black-box approach remains surprisingly unknown [30]. This observation raises the following questions [12]:

*Is it possible to construct NMZK argument systems relying solely on the black-box use of cryptographic primitives? Can these systems be designed to operate in a constant number of rounds? For efficiency purposes, can we limit ourselves to using only one-way functions? If so, can the efficiency surpass the state-of-the-art non-black-box construction?*

The protocol proposed by Dr. Michele et al., as outlined in an unpublished paper [12], provides positive answers to all the questions above. It is the first NMZK argument system based on the black-box use of underlying primitives. Furthermore, it only needs the OWFs and can be instantiated in a small constant number of rounds.

The primary goal of this project is to thoroughly understand Dr. Michele et al.'s protocol and to implement it in practice. Following a successful implementation, we also conducted benchmark testing against the recent non-black-box NMZK argument system based on symmetric assumptions proposed by Kim et al. [30]. This system will serve as the baseline for our comparisons. The findings from our evaluation indicate that Dr. Michele et al.'s protocol significantly outperforms the protocol proposed by Kim et al. in terms of concrete efficiency and communication complexity when living in the Minicrypt [25], where we only assume the existence of OWFs but not public-key cryptography.

This is a two-man project (me and Ethan) with the work split as follows:

Each of us has conducted the benchmark tests using our own devices, therefore the results may differ slightly.

Table 1.1: Contributions to the Project

| **Ethan** | **Xuhan (Me)** | **Together** |
|---|---|---|
| Naor's commitment | Extractable commitment | Merging work (Ethan did the main part) |
| Weak Non-malleable commitment | Schnorr's Protocol | ZKBoo++ |
| Benchmark test | Benchmark test | |

## 1.1 Related Work

Considering the general-purpose zero-knowledge proofs that work for all NP languages, there are mainly three methods to achieve non-malleability. The most common approach involves using non-malleable commitment schemes in the proof construction. The process begins with the prover sending a non-malleable commitment of the witness, followed by a standard zero-knowledge proof to demonstrate that the committed value is indeed a valid witness [13, 22]. This approach is also the method used in our protocol construction.

Also, note that state-of-the-art non-malleable commitment requires a Decisional Diffie-Hellman (DDH) assumption [7]. Conversely, theoretical considerations suggest that "symmetric assumptions", such as the existence of one-way functions should be enough. The preference is therefore to avoid assuming any public-key assumptions such as DDH since it would impact the efficiency of the protocol.

An alternative approach for achieving non-malleability, which does not rely on non-malleable commitments, was initially introduced by Barak [2] and further refined by Pass et al. [37] under improved assumptions. These works were fundamentally based on Barak's non-black-box simulation [3] and they all utilised the technique called *universal argument* [4], which involves a Merkle tree commitment to a Probabilistically Checkable Proof (PCP). Parts of the proofs are then opened later in the protocol. Unfortunately, Ben-Sasson et al. [6] showed that the underlying PCP proof in the universal argument could reach prohibitively large sizes even for moderate parameters.

The third approach depends on the DDH assumption and efficiently transforms any public-coin honest-verifier zero-knowledge argument into a concurrent non-malleable zero-knowledge proof [5, 35]. Although this method employs non-malleable commitments, it avoids general-purpose proofs over them by using the concepts from the "simulatable commitment", introduced by Micciancio et al. [32]. This approach requires not only efficient non-malleable commitments but also compatible and efficient simulatable commitments, both are only available under DDH assumptions. However, we are more interested in using only symmetric assumptions due to efficiency concerns.

Furthermore, we focus on analysing Kim et al.'s work [30] which is the first efficient non-black-box constant-round NMZK argument system based on symmetric assumptions, specifically, requiring collision-resistant hash functions. It is also the baseline for our comparison. Hereafter, we will denote Kim et al.'s protocol as $\Pi_{KLP}$.

$\Pi_{KLP}$ has introduced a novel technique called instance-based non-malleable commitment (IB-NMC), which can be considered a more efficient variant of standard non-

malleable commitments. This technique modifies the non-malleable commitment of [7], denoted as $\Pi_{BGRRV}$, which consists of a three-round commit phase and a subsequent proof phase for consistency verification. The key adaptation in $\Pi_{KLP}$ involves changing the proof phase of $\Pi_{BGRRV}$ to an adapted version of OR-composition [14] of two instances of the Ligero [1], which is a general-purpose zero-knowledge protocol. This adaptation incurs additional computational and communication costs due to the variant of the OR-composition required. This variant is necessary because Ligero is not a $\Sigma$-protocol while OR-composition typically only applies to $\Sigma$-protocols. Moreover, to achieve non-malleability, their method needs to repeatedly apply the OR-composition, which leads to multiple executions of Ligero and thus results in a total of over 28 rounds for completion. This extensive repetition significantly impacts both the round complexity and computational efficiency of $\Pi_{KLP}$. A detailed explanation of $\Pi_{KLP}$ is available in [30].

# Chapter 2

# Preliminary

In this dissertation, we will use the same notation established in [12], wherever feasible. The security parameter is represented by $\lambda$. The notation [n] refers to the set $\{1, \ldots, n\}$. The term "PPT" is shorthand for probabilistic polynomial time. Vectors are indicated in boldface, and the inner product of two vectors is expressed as $\langle \cdot, \cdot \rangle$.

A function $f : N \to N$ is said to be negligible, denoted as negl($\lambda$), if for all constant $c$, there exists $N$ such that $\forall \lambda > N : f(\lambda) < \lambda^{-c}$.

Two interactive machines are introduced, A and B, and describe an interactive protocol between them as (A, B). The notation $\langle A(a), B(b) \rangle (x)$ specifies the interaction between A and B given a common input x, with $a$ as the private input for A and $b$ as the private input for B. The transcript resulting from this interaction is denoted by $\tau$, which is generated by $\langle A(a), B(b) \rangle (x)$.

## 2.1 Commitment Scheme

A commitment scheme is a cryptographic protocol that allows a party, called committer C, to commit to a chosen value (or chosen statement) while keeping it hidden from others, with the ability to reveal the committed value to receive R, later. The commitment scheme is designed such that once a value is committed, it cannot be changed.

Formally, a commitment scheme $\Pi_{com} = (C, R)$ is a two-phase protocol. In the first phase, called the *commit phase*, C chooses a message m that they wish to commit to along with a random value $r_c$ called a nonce and sends $\pi_1 = C(m, r_c)$ to R. After receiving $\pi_1$, R responds with a random challenge $r_r$. Overall, let $\tau = \langle C(m, r_c), R(r_r) \rangle$ denote the commitment transcript with committer input $m$, committer randomness $r_c$ and receiver randomness $r_r$. In the second phase, called the *open phase*, C opens the commitment send earlier based on randomness $r_r$ and sends opening information to R. R accepts the value if and only if values matched against the committer value received in the opening phase. Let $Dec(\tau, m, r_c)$ denote the algorithm that takes three inputs: a commitment transcript $\tau$, a message $m$ from the committer, and randomness $r_c$. It outputs 1 or 0 to indicate whether the decommitment is accepted or not, respectively. [12]

A commitment scheme is said to be delayed input when the message intended for commitment is only needed in the final round of the commit phase by the committer, C. A typical commitment scheme must satisfy two properties: hiding and binding. Hiding ensures that the commitment conceals the actual message from anyone other than C while binding prevents C from changing the message after the commitment has been made. Here, we present the classic definitions from [19].

**Definition 1 (Statistical Binding)** *A commitment scheme $(C,R)$ is said to be statistically binding if for every $C^*$ there exists a negligible function $\upsilon$ such that $C^*$ succeeds in the following game with probability at most $\upsilon(\lambda)$:*

- *On input the security parameter $\lambda$, $C^*$ interacts with $R$ in the commit stage obtaining commitment $\tau$.*
- *$C^*$ outputs pairs $(m_0,r_0)$ and $(m_1,r_1)$.*
- *$C^*$ succeeds if $Dec(\tau,m_0,r_0) = Dec(\tau,m_1,r_1) = 1$ and $m_0 \neq m_1$.*

**Definition 2 (Computationally Hiding)** *A commitment scheme $(C,R)$ is said to be computationally hiding if for every PPT $R^*$, and every two messages $m_0,m_1$, the view of $R^*$ after a commitment phase where C committed to $m_0$ is computationally indistinguishable from the view of $R^*$ after participating a commitment phase where C committed to $m_1$.*

The term *well-formed* is used to describe a transcript, $\tau$, from the commitment phase, for which a pair $(m,r_c)$ exists, such that $Dec(\tau,m,r_c) = 1$.

## 2.2 Extractable Commitment

Intuitively, an extractable commitment scheme is a cryptographic construction in which, given a commitment, there exists an extraction algorithm (typically possessed by a trusted entity or simulator) that can derive or "extract" the committed value without the need for the usual decommitment process, conditioned on the commitment being well-formed. In particular, if the commitment is maliciously generated, then the extractor is expected to output $\perp$, where $\perp$ represents an undefined or null value. Conversely, if the commitment is created honestly, then the extractor should successfully retrieve and output the correct value committed. More formally, we have the following definition [12, 38]:

**Definition 3 (Extractable Commitment)** *Let $(C,R)$ denote a statistically binding, computationally hiding commitment scheme. We define $(C,R)$ as an extractable commitment scheme if there exists an expected PPT oracle algorithm Ext (the extractor), such that for any PPT committer $C^*$ the following holds. Let $\tau = \langle C^*, R(r_r)\rangle$ denote a (potentially maliciously generated) transcript of the interaction between $C^*$ and R. The extractor $Ext^{C^*}(\tau,r_r)$, with oracle access to $C^*$, outputs m such that, over the randomness of Ext and of sampling $\tau$,*

$$\Pr[(\exists \tilde{m} \neq m, \tilde{r}_c) : Dec(\tau,\tilde{m},\tilde{r}_c) = 1] \leq negl(\lambda)$$

We can extend the definition of extractable commitment above to multiple transcripts as follows [12]:

**Definition 4 (k-Extractable Commitment)** *An extractable commitment satisfying Definition 3 is said to be k-extractable if there exists a PPT extractor algorithm Ext such that, given a set of k well-formed transcripts sharing the same first round committer message of the commitment phase, the extractor successfully extracts the value committed in the first transcript, except with negligible probability.*

## 2.3 Non-malleable Commitment

A non-malleable commitment is a cryptographic primitive that ensures that once a party has committed to a value, another adversarial party cannot generate another commitment that is related in some meaningful way to the original commitment, even if they see the original commitment. Non-malleable commitments are a stronger form of commitment than standard commitment schemes. At first glance, it might seem that non-malleability is unachievable, especially when considering that a MIM attacker could merely replicate messages from one protocol session to another. Therefore, non-malleable security doesn't necessarily prevent all forms of copying by a MIM. Instead, its primary objective is to guard against a MIM who attempts to modify messages substantively.

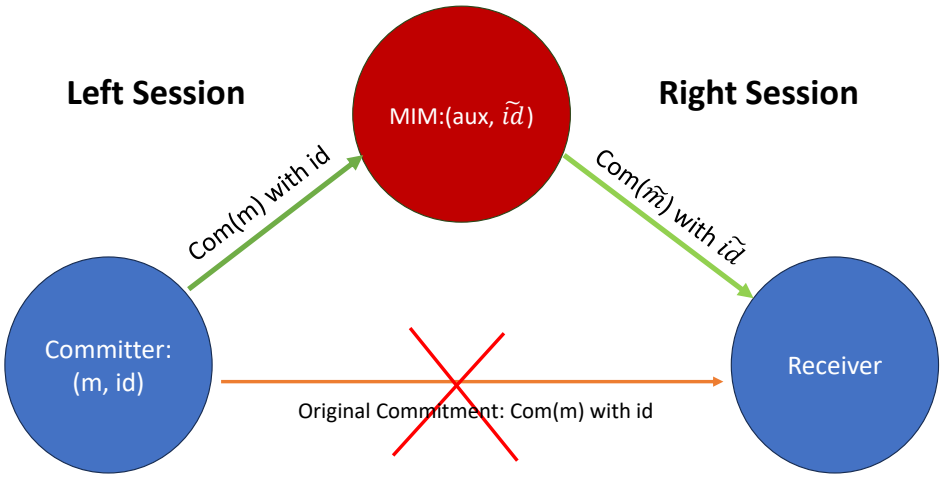


Figure 2.1: Simple Illustration of MIM attacks. The randomness is omitted for simplicity. We consider the attack to be successful if the commitment in the right session is accepted and $\tilde{m}$ is related to $m$ while $id \neq \tilde{id}$.

We use the tag-based definition as described in [7, 21, 31]. This framework considers the scenarios involving a man-in-the-middle (MIM) adversary engaged in two concurrent sessions: in the left session, MIM interacts as a receiver with an honest committer C, and in the right session, MIM plays the role of a committer with an honest receiver R.

To distinguish between the entities in these sessions, we use a 'tilde' notation for those in the right session. For instance, if C commits a value $m$, then MIM commits a value $\tilde{m}$ in the right session. The committer has an identity (or 'tag') $id \in \{0,1\}^\lambda$ of their choice. At the beginning of the commitment phase, C receives $m$ as local input, while MIM receives an auxiliary input $aux$.

To evaluate the non-malleability of the commitment scheme, two types of executions are considered: the real execution, where MIM interacts with both C and R, and the simulated execution, where a simulator, SIM, takes the place of MIM in interacting with R. In the real execution, MIM's objective is to commit to a value $\tilde{m}$ in the right session that is somehow related to $m$, committed by C in the left session, using a distinct identity $\tilde{id}$. The attack is considered to be a failure if MIM uses the same identity in both interactions (i.e. MIM uses the same identity as the honest committer) or if the commitment is otherwise invalid, or undefined, in which case the committed value is set to $\perp$.

Let the random variable $MIM_{(C,R)}((m), aux)$ represent the pair (view, $\tilde{m}$), consisting of MIM's view and the values committed by MIM during the real execution. Conversely, in the simulated execution, SIM interacts with R, aiming to replicate the role of MIM. The random variable $SIM_{(C,R)}(1^\lambda, aux)$ describes the pair (view, $\tilde{m}$), reflecting the view of SIM and the value of the commitments in the simulated execution. As in the real execution, if SIM commits using an identity identical to one from the left session or the commitment in the right session is invalid, the committed value is set to $\perp$.

The effectiveness of the non-malleable commitment scheme is measured by comparing the outcomes of these executions, specifically looking at the values committed by MIM or SIM and their respective views during the interaction. The scheme aims to ensure that, regardless of MIM's strategies, it cannot produce a related commitment $\tilde{m}$ using the identity $\tilde{id}$ in a way that breaks the scheme's security. We report the formal definitions from [22].

**Definition 5 (Non-malleable Commitment)** *A commitment scheme is non-malleable with respect to commitment if, for every PPT man-in-the-middle (MIM) adversary, there exists a PPT simulator SIM such that the following ensembles are computationally indistinguishable for all $m \in \{0,1\}^\lambda$:*

$$\{MIM_{(C,R)}(m, aux)\}_{aux \in \{0,1\}^*} \quad \textit{and} \quad \{SIM_{(C,R)}(1^\lambda, aux)\}_{aux \in \{0,1\}^*}$$

## 2.4 Arguments/Proofs

**Definition 6 (Interactive Argument/Proof System)** *A pair of PPT interactive algorithms $\Pi = (P,V)$ constitutes a proof (respectively, argument) system for an NP-language $L$ that is associated with the relation $Rel_L$ if the following conditions hold:*

- ***Completeness:*** *For every $x \in L$ and $w$ such that $Rel_L(x,w) = 1$, it holds that $V$ accepts the proof with probability* 1.

- ***Soundness:*** *For every algorithm $P^*$ (respectively, PPT algorithm $P^*$) there exists a negligible function negl such that for every $x \notin L$ and every auxiliary input aux:*

$$\Pr\left[Out_V\langle P^*(aux), V\rangle(x) = 1\right] \leq negl(|x|).$$

In the interactive proof system $\Pi = (P,V)$, defined as "public coin", the verifier $V$ generates random challenges by tossing a predetermined number of coins in every round of interaction with the prover $P$. These challenges are then sent to $P$. The communication between $P$ and $V$ produces a series of messages, denoted as $\pi^1, \pi^2, ..., \pi^\ell$, where $\ell$ is a constant which represents the total number of rounds in the protocol. The execution of this interaction for a specific instance $x$ with $P$'s witness $w$ results in a transcript $\tau$, which is considered accepted if the outcome of $V$'s verification process, represented by $Out_V(\langle P(w), V\rangle(x))$, equals 1. The following discussion focuses on proof systems where the interaction has exactly three rounds ($\ell = 3$).

**Definition 7 (SHVZK [12])** *Let $L$ be an NP-language with the associated relation $Rel_L$. A 3-round proof (respectively, argument) system $\Pi = (P,V)$, as defined in Definition 6, is special honest-verifier zero knowledge (SHVZK) if there exists a probabilistic PPT algorithm Sim such that, for any $x \in L$, security parameter $\lambda$, and any challenge $\pi_2$, works as follows:*

$$(\pi^1, \pi^3) \leftarrow Sim(1^\lambda, x, \pi^2).$$

*Furthermore, the distribution of the output of Sim is (computationally) indistinguishable from the distribution of a transcript obtained when $V$ sends $\pi_2$ as a challenge and $P$ runs on common input $x$ and any $w$ such that $Rel_L(x,w) = 1$.*

We also recall the definition of canonical extractability on proofs:

**Definition 8 (Proof of Knowledge with Canonical Extractability [12])** *A 3-round proof system $\Pi = (P,V)$ for an NP-language $L$ that is associated with the relation $Rel_L$ as defined in Definition 6, is a proof of knowledge with canonical extractor if, for all $k \in N$, there exists an expected PPT extractor Ext such that, if $P^*$ interacting with $V$ produces an accepting transcript for $x \in L$ with non-negligible probability, then:*

- *On input $x$ and an accepting transcript $(\pi^1, \pi^2, \pi^3)$, Ext rewinding multiple times $P^*$ and playing each time a new random challenge obtains, with overwhelming probability, a constant number $k$ of accepting transcripts, all with the identical $\pi^1$ but distinct challenges;*
- *On input $k$ accepting transcripts $(x, \pi^1, \pi_i^2, \pi_i^3)_{i\in[k]}$ such that for each $j, z \in [k]$, $j \neq z$, $\pi_j^2 \neq \pi_z^2$, Ext outputs a witness $w$ such that $Rel_L(x,w) = 1$, with probability 1 and in a strict polynomial number of steps.*

Overall, canonical Extractability refines the notion of extractability by imposing stricter conditions on how the witness $w$ can be extracted. Specifically, it refers to a scenario where the extractor can operate in a black-box manner and there is a clear, well-defined method for extraction that works across various implementations of the proof system.

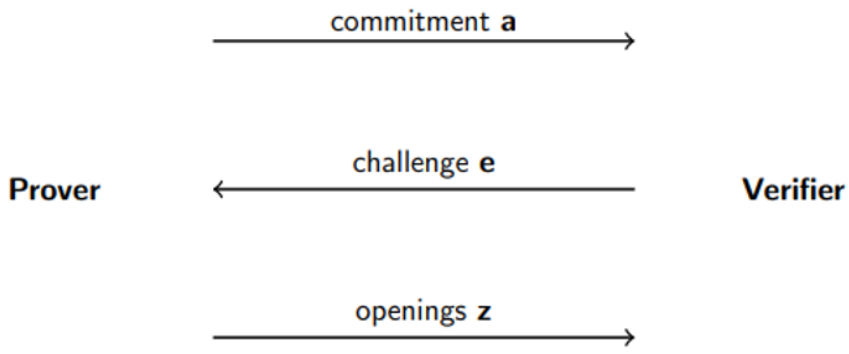


Figure 2.2: 3-move Sigma Protocol

## 2.5 Σ-Protocols

The concept of the Σ-protocol generalises the scope of identification protocols as originally introduced by Schnorr [39], Guillou and Quisquater [24], among others. It is a special case of the SHVZK which follows a 3-move structure as shown in Figure 2.2.

**Definition 9 (Σ-protocol)** *Let $L$ be an NP-language with the associated relation $Rel_L$. An interactive protocol $\Pi = (P,V)$ is a Σ-protocol if it follows the following 3-move form:*

*Given a common input $x$ and witness $w$ which is a private input for $P$ such that $Rel_L(x,w) = 1$:*

1. *$P$ sends a message $a$.*
2. *$V$ sends a random $t$-bit string $e$.*
3. *$P$ sends a reply $z$, and $V$ decides to accept or reject based on $x$, $a$, $e$, and $z$.*

*And satisfying the following three properties:*

- ***Completeness**: For every $x \in L$ and $w$ such that $Rel_L(x,w) = 1$, it holds that $V$ accepts the proof with probability 1.*
- ***Special Soundness**: For any given input $x$ and any two distinct accepting interactions based on $x$, denoted as $(a,e,z)$ and $(a,e',z')$ where $e \neq e'$, it is possible to effectively compute a witness $w$ such that $RelL(x,w) = 1$.*
- ***Special Honest-Verifier Zero-Knowledge**: As defined in Definition 7.*

Contrary to a typical SHVZK protocol, which primarily possesses a soundness characteristic, a Σ-protocol is enhanced with the property of Special Soundness. This feature not only guarantees the protocol's soundness but also introduces extractability, allowing for extracting the witness from successful proofs.

## 2.6 Non-malleable Interactive Arguments/Proofs

Non-Malleable Zero Knowledge (NMZK) is defined similarly to non-malleable commitments, as discussed in section 2.3. We continue to employ a tag-based definition, but now our focus shifts to argument/proof systems.

Let $\Pi = (P,V)$ be an interactive argument or proof system for an NP language $L$, with its corresponding relation $Rel_L$. Suppose $x \in L$ is the public input of the protocol with a length of $\lambda$, and $P$ possesses a private input $w$ such that $Rel_L(x,w) = 1$. Consider a PPT man-in-the-middle adversary, MIM, that participates in two simultaneous sessions: one on the left session and one on the right session. MIM receives an auxiliary input $aux \in \{0,1\}^*$. In the left session, the MIM verifies the validity of the statement $x$ by interacting with $P$, choosing an identity $id$ for itself. In the right session, MIM attempts to prove the validity of a chosen statement $\tilde{x}$ with an identity $\tilde{id}$ of MIM's choice. Let the random variable $view^{MIM}(x,aux)$ denote MIM's view in this experimental setup.

**Definition 10 (Non-Malleable Zero Knowledge [12])** *An argument/proof system $\Pi = (P,V)$ for an NP-language $L$ with relation $Rel_L$ is non-malleable zero knowledge (NMZK) if, for any man-in-the-middle adversary MIM that participates in one left session and one right session, there exists an expected PPT simulator $Sim(x,aux)$ such that:*

- *The two ensembles $\{Sim^1(x,aux)\}_{\lambda \in N, aux \in \{0,1\}^*}$ and $\{view^{MIM}(x,aux)\}_{\lambda \in N, aux \in \{0,1\}^*}$ are computationally indistinguishable with respect to $\lambda$. Here, $Sim^1(x,aux)$ refers to the first output of $Sim(x,aux)$.*
- *Given the output $(view, \tilde{w})$ from $Sim(x,aux)$. Let $\tilde{x}$ represent the instance from the right session within view, id and $\tilde{id}$ be the identities used in the left and right sessions, respectively. If the right session ends in acceptance and $id \neq \tilde{id}$, then the pair $(\tilde{x}, \tilde{w})$ must satisfy the relation $Rel_L$.*

# Chapter 3

# Overview of the Protocol $\Pi_{NM}$

In this chapter, we discuss the 9-round protocol which we implemented. The primary goal of this project is the practical implementation of the protocol rather than a detailed validation of its theoretical correctness. Therefore, this section will only provide a brief overview of the protocol's fundamental architecture. For the details of the proof and how reduction works, readers are referred to [12].

## 3.1 The Naive Protocol and Limitation

We start with a 3-round public-coin special honest-verifier zero-knowledge proof of knowledge (SHVPoK) $\Pi_{SHVZK}$ for a common input $x \in L$, where $L$ is an NP-language. The execution transcript of the proof is represented by three components: $(\pi_1, \pi_2, \pi_3)$, with $\pi_2$ acting as the challenge.

To protect the protocol from malicious verifiers while maintaining its zero-knowledge essence, the simulator (that does not know the witness for $x$) needs to be able to decide the challenge $\pi_2$ upfront. This is achieved by computing $\pi_2$ from two sub-challenges, $c_0$ and $c_1$, through an XOR operation. The steps are as follows:

- **Verifier's Sub-Challenge ($c_0$)**: After receiving $\pi_1$ from the prover, the verifier selects a sub-challenge $c_0$ through an extractable commitment scheme, denoted as $\Pi_{3Ext}$.
- **Prover's Sub-Challenge ($c_1$)**: Following the commitment phase of $\Pi_{3Ext}$, the prover chooses and sends the second sub-challenge $c_1$. This allows the prover to contribute to the challenge without direct knowledge of $c_0$. The opening to $c_0$ is played right after $c_1$ is sent.
- **Challenge Construction**: The conclusive challenge $\pi_2$ is then formulated as the XOR of $c_0$ and $c_1$, expressed as $\pi_2 = c_0 \oplus c_1$. After the unveiling of $c_0$ by the verifier, the prover computes and sends $\pi_3$, thereby finalizing the protocol's execution.
- **Simulation**: The simulator, by running the extractor of $\Pi_{3Ext}$, can set the challenge $\pi_2$ through an adaptive computation of $c_1$ (specifically, $c_1 = \pi_1 \oplus c_0$).

While the above mechanism allows for a secure challenge generation that safeguards against malicious verifiers and provides zero knowledge, it does not achieve non-malleability since all the sub-protocols described are malleable. Recall that the core method of establishing non-malleability involves demonstrating that a simulator can extract a witness from the statement an adversary is proving in a right session, even when the adversary receives a simulated proof as they act as a verifier in a left session. This approach suggests that the protocol achieves simulation-extractability, which inherently implies non-malleability. This is because the simulator, once it obtains the witness while interacting with the adversary, can effectively become an independent adversary capable of proving the statement to an external verifier.

Assume that $\Pi_{SHVZK}$ is associated with a canonical extractor. This extractor, by performing a series of rewinds, seeks to collect additional pairs of responses $(\tilde{\pi}_2^i, \tilde{\pi}_3^i)$, specifically, requiring just one more pair if the protocol exhibits special soundness. The goal is to ensure that the sequence $(\tilde{\pi}_1, \tilde{\pi}_2^i, \tilde{\pi}_3^i)$ is accepted, and that the values of $\tilde{\pi}_2^i$ are unique compared to each other and to $\tilde{\pi}_2$. During the simulation, the challenge $\pi_2$ is predetermined (this challenge is used to operate the SHVZK simulator for generating $(\pi_1, \pi_2, \pi_3)$), and forced to be distinct across all simulated transcripts as outlined. However, if an adversary manages to replicate the simulator's strategy, they would potentially be able to use a specific challenge $\tilde{\pi}_2$ across multiple executions of the protocol during rewinding, thus preventing the successful extraction.

## 3.2 Achieving Non-malleability with $\Pi_{BGRRV}^{3R}$

The critical drawback identified in the naive protocol is its vulnerability to malleability. To address this, we integrate a weak-non-malleable commitment scheme, denoted as $\Pi_{BGRRV}^{3R}$, into the $\Pi_{SHVZK}$.

**Introduction to $\Pi_{BGRRV}^{3R}$.** $\Pi_{BGRRV}^{3R}$ represents the first three rounds of the particular commitment scheme, denoted as $\Pi_{BGRRV}$ from [7]. This scheme is designed to be weak non-malleable, a property that, while requiring less strict criteria than full non-malleability, still offers a meaningful level of security under certain conditions. We briefly recall how $\Pi_{BGRRV}$ works: Initially, the sender commits to a message $m$ along with a random group element $r$ through a non-interactive commitment scheme. Following this, the receiver contributes by sending another random group element $\alpha$, to which the sender responds with $a = r\alpha + m$, effectively binding the commitment with the receiver's input.

Additionally, for the purpose of our discussion, we will assume $\Pi_{BGRRV}^{3R}$ to have intractability for now, with the specifics of integrating this feature to be explained in the next section. Weak non-malleability ensures security under the assumption that the adversary can only produce valid (well-formed) commitments. We will later demonstrate why this level of non-malleability suffices for our protocol.

**Achieve Non-malleability**. In contrast to the naive protocol where a verifier's commitment could be potentially malleable, the $\Pi_{BGRRV}^{3R}$ protocol introduces a layer of protection by ensuring that the initial commitment to the sub-challenge $c_0$ exhibits weak non-malleability. It means that an adversary, even after observing the commitment,

is not capable of creating a related commitment that could potentially influence the challenge $\pi_2$ in their favour. The strength of $\Pi^{3R}_{BGRRV}$ lies in its use of the canonical extractor from $\Pi_{SHVZK}$. It works as follows: the extractor commits to a random string $\tilde{c}_0$ using $\Pi^{3R}_{BGRRV}$. Upon securing an accepting transcript $\pi_1, \tilde{\pi}_2, \tilde{\pi}_3$, where $\tilde{\pi}_2 = \tilde{c}_0 \oplus \tilde{c}_1$, the extractor attempts to rewind the adversary. This rewind alters the committed string from $\tilde{c}_0$ to a newly random $\tilde{c}'_0$, enabling the protocol to complete a new right session with the adversary and generate a new transcript $\tilde{\pi}_1, \tilde{\pi}'_2, \tilde{\pi}'_3$. This process of modifying the commitment of sub-challenge, from $\tilde{c}_0$ to $\tilde{c}'_0$, guarantees the unpredictability and distinctiveness of $\pi_2$ and its variations $\pi'_2$. If an adversary successfully makes $\pi'_2 = \pi_2$, it would signal a compromise of the $\Pi^{3R}_{BGRRV}$ protocol's weak non-malleability.

It's important to outline that the requirement for weak non-malleability in the commitment scheme is enough for this context. For an adversary to be considered successful, they must not only commit but also accurately open the commitment within the right session. Thus, any attempt at a malleability attack resulting in a malformed commitment—which contrasts with the integrity ensured by full non-malleability—would not equate to success in our NMZK argument system.

## 3.3 Making $\Pi^{3R}_{BGRRV}$ Extractable

The $\Pi^{3R}_{BGRRV}$ protocol inherently lacks *standard extractability* due to its design, which allows a corrupted sender to alter the commitment process. Standard extractability is defined as the ability to rewind a corrupted sender in an interaction with an honest receiver and extract a message $m$ corresponding to their commitment. This process ensures that if the commitment is valid and can be opened, the extracted message will be accurate; however, if not, the correctness of the extracted message cannot be guaranteed. In the case of $\Pi^{3R}_{BGRRV}$, when attempting to extract the committed message by rewinding the sender, the extractor initiates a new interaction by sending an alternate second-round message, $\alpha'$, and subsequently receiving $a'$ in response. At this point, the extractor can interpolate between the points $(a, \alpha)$ and $(a', \alpha')$ to derive a new message $m'$. A corrupted sender may use different values for $r$ and $m$ to compute $a'$, resulting in $m'$ being extracted which does not match the original $m$. This issue arises due to the hiding property of the non-interactive commitment, which prevents the extractor from confirming the accuracy of the extracted message. Consequently, even if the sender's original commitment was valid, the protocol cannot ensure successful extraction of the correct message $m$, thereby failing to meet the criteria for standard extractability.

To address this limitation, a simple modification [12] leads to the development of the $\Pi_{5Ext}$, which integrates a three-round extractable commitment scheme $\Pi_{3Ext}$. This adjustment replaces the non-interactive commitments in $\Pi^{3R}_{BGRRV}$ with the goal of preserving weak non-malleability while achieving *standard extractability*. This modification allows for the successful retrieval of the message $m$ from a well-formed commitment by leveraging the extractability of $\Pi_{3Ext}$, the extractor of $\Pi_{5Ext}$ simply runs the extractor of $\Pi_{3Ext}$. Moreover, $\Pi_{5Ext}$ is also made to be delayed-input.

## 3.4 Final Protocol $\Pi_{NM}$

**Theorem 1** *[12] Assuming the existence of One-Way Functions (OWFs), there exists a 10-round Non-Malleable Zero-Knowledge (NMZK) protocol, which makes black-box use of OWFs. With the assumption of the existence of one-to-one OWFs, a 9-round NMZK protocol can be achieved.*

This theorem distinguishes between the two scenarios based on their round complexity. The distinction arises because the initial round of the protocol $\Pi_{NM}$ may require computing a non-interactive commitment, which, in turn, requires an additional preliminary round for its instantiation with One-Way Functions (OWFs).

Concluding the discussions in this chapter, the details of the final protocol are presented in Figure 3.1. It can be considered as a compiler from 3-round public-coin special honest-verifier zero-knowledge $\Pi_{SHVZK}$ to non-malleable zero-knowledge [12].

Notation: Let $\Pi = (P, V)$ be a 3-round public-coin SHVZK proof of knowledge for the language $L$ and associated relation $\text{Rel}_L$. Let $(\pi^1, \pi^2, \pi^3)$ be the three round messages of $\Pi$. Let $\Pi_{5\text{Ext}}$ be the 5-round delayed-input extractable commitment scheme. Let us denote with $(\text{com}_1, \text{com}_2, \text{com}_3, \text{com}_4, \text{com}_5)$ the transcript of the commit phase of $\Pi_{5\text{Ext}}$ and with **dec** the corresponding opening information.
Public parameters: $\lambda, x \in L$.
$P_{NM}$'s Private input: $w$ such that $(x, w) \in \text{Rel}_L$.

- **Round 1**. $P_{NM}$ on input $(x, w)$, computes the first round $\pi^1$ of $\Pi$ and sends $\pi^1$ to $V_{NM}$.
- **Round 2**. Upon receiving $\pi^1$, $V_{NM}$, plays the role of C, computes the first round $\text{com}_1$ of $\Pi_{5\text{Ext}}$ and sends $\text{com}_1$ to $P_{NM}$.
- **Round 3**. On input $\text{com}_1$, $P_{NM}$ computes the second round $\text{com}_2$ of $\Pi_{5\text{Ext}}$ and sends $\text{com}_2$ to $V_{NM}$.
- **Round 4**. On input $\text{com}_2$, $V_{NM}$ computes the third round $\text{com}_3$ of $\Pi_{5\text{Ext}}$ and sends $\text{com}_3$ to $P_{NM}$.
- **Round 5**. On input $\text{com}_3$, $P_{NM}$ computes the fourth round $\text{com}_4$ of $\Pi_{5\text{Ext}}$ and sends $\text{com}_4$ to $V_{NM}$.
- **Round 6**. On input $\text{com}_4$, $V_{NM}$ samples at random $c_1 \in \{0,1\}^\lambda$. $V_{NM}$ computes the fifth round $\text{com}_5$ of $\Pi_{5\text{Ext}}$ w.r.t. message $c_1$ and the corresponding opening **dec** of $\Pi_{5\text{Ext}}$. $V_{NM}$ sends $\text{com}_5$ to $P_{NM}$.
- **Round 7**. $P_{NM}$ samples $c_2 \in \{0,1\}^\lambda$ and sends $c_2$ to $V_{NM}$.
- **Round 8**. $V_{NM}$ sends $(c_1, \textbf{dec})$ to $P_{NM}$.
- **Round 9**. On input $(c_1, \textbf{dec})$, $P_{NM}$ acts as follows. If **dec** is not a valid opening for $(\text{com}_1, \text{com}_2, \text{com}_3, \text{com}_4, \text{com}_5)$ w.r.t. committed message $c_1$ then $P_{NM}$ aborts. Otherwise, $P_{NM}$ sets $\pi^2 = c_1 \oplus c_2$ and computes the 3rd round $\pi^3$ of $\Pi$. $P_{NM}$ sends $\pi_3$ to $V_{NM}$.

**Verification procedure.** On input $\pi^3$, $V_{NM}$ computes $\pi^2 = c_1 \oplus c_2$. If $(x, \pi^1, \pi^2, \pi^3)$ is an accepting transcript for $\Pi$, then $V_{NM}$ accepts, otherwise aborts.

Figure 3.1: Description of $\Pi_{NM} = (P_{NM}, V_{NM})$ from [12]

# Chapter 4

# Overview and Implementation Details of Sub-protocols

In this chapter, the implementation details of the protocol $\Pi_{NM}$ are presented, focusing specifically on the components for which the author of this dissertation was responsible. $\Pi_{NM}$ is composed of four building blocks as shown in Figure 4.1, each essential for its overall functionality:

1. A non-interactive statistically binding commitment scheme: Naor's commitment scheme from [34].
2. A weak non-malleable commitment: $\Pi_{BGRVV}$ from [7].
3. An extractable commitment: $\Pi_{3Ext}$ from [38].
4. A 3-round public-coin special honest-verifier zero-knowledge (SHVZK) protocol, with $\Sigma$-protocols as the chosen implementation.

Among these, the author's efforts were concentrated on the latter two components. This involved the practical implementation of both the extractable commitment, $\Pi_{3Ext}$ and the $\Sigma$-protocols (specifically, including Schnorr's protocol [39] and an optimised version of ZKBoo, ZKBoo++ [11]). This section will discuss the details of these sub-protocols and the key implementation decisions made.

We implemented all the protocols in Rust and our implementation is available at `https://github.com/non-malleable-zero-knowledge/non-malleable-zero-knowledge.git`.

## 4.1 Extractable Commitment $\Pi_{3Ext}$

We implemented the 3-round public-coin extractable commitment scheme, $\Pi_{3Ext} = (C_{3Ext}, R_{3Ext})$, based on the framework presented in section 4 of [38]. This extractable commitment, $\Pi_{3Ext}$, is thoroughly illustrated in Figure 4.2. Within this Figure, "Com" represents the commitment phase of a non-interactive, statistically-binding commitment scheme.

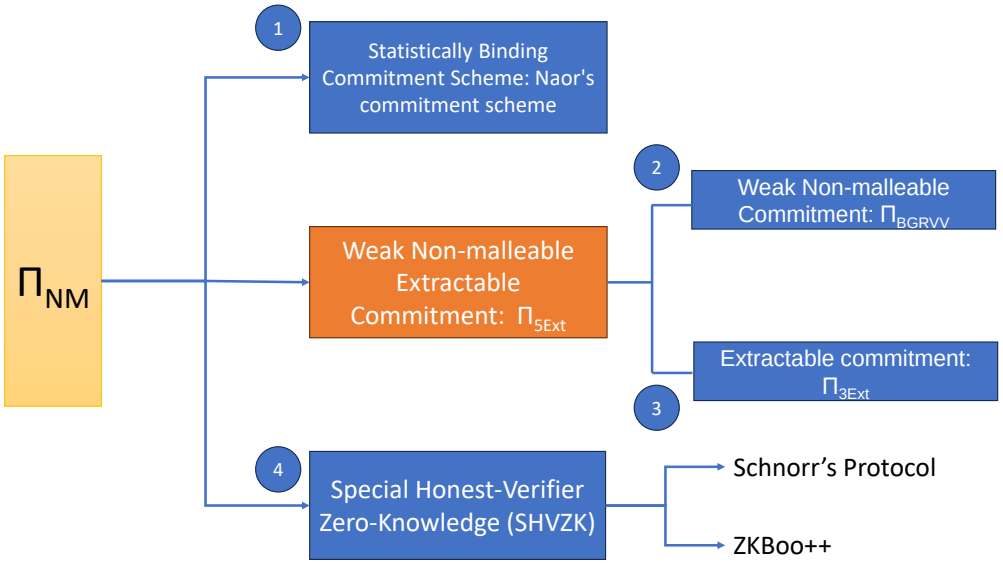


Figure 4.1: Essential Building Blocks of $\Pi_{NM}$

**Introduction to $\Pi_{3Ext}$.** In the described extractable commitment scheme, two entities participate in a secure communication protocol: the committer, $C_{3Ext}$, holds a private message $m$, and the receiver, $R_{3Ext}$, participates in verifying the commitment. Initially, $C_{3Ext}$ creates a commitment to $\lambda$ pairs of strings, with each pair consisting of a randomly chosen string and its bitwise XOR with the message $m$. These commitments are then sent to $R_{3Ext}$, who responds with a $\lambda$-length binary challenge. $C_{3Ext}$ then opens the commitments specified by this challenge. The decommitment phase follows, where $C_{3Ext}$ reveals the original message and the necessary information to open the commitments, allowing $R_{3Ext}$ to confirm that the message is consistent with the initial commitment by performing the XOR operation on the opened strings.

**Implementation Details**. The multi-bit version of the Naor commitment scheme [33] was chosen for the statistical binding commitment scheme "COM". Despite the existence of more efficient schemes like hasher, we selected Naor's scheme in order to make the valid comparison with our baseline protocol $\Pi_{KLP}$ [30], which employs Naor's scheme in their framework.

In our actual implementation, we've optimized the process by substituting the bitwise XOR operation specified in the original protocol with modular addition over elements of a prime field. Specifically, during the initial round, $v_i^0$ is a randomly chosen field element from a prime field, while $v_i^1$ is the result of the modular addition of $v_i^0$ and the message $m$ (i.e. $v_i^1 = m + v_i^0 \mod p$). This modification necessitated a corresponding change in the verification phase to check the equality of the modular subtraction of each pair of $v_i^1$ and $v_i^0$ with the message $m$ (i.e. $m = v_i^1 - v_i^0 \mod p$). We decided to use the "PrimeField" traits from the "ark-ff" crate for this purpose, which allows for the creation of a prime field with any chosen prime $p$.

The protocol was implemented interactively in a modular manner. We have encapsulated the functionality of the committer $C_{3Ext}$ and the receiver $R_{3Ext}$ within separate classes,

NOTATION: Let Com be a statistically-binding commitment scheme.
PUBLIC PARAMETERS: $\lambda$.
PRIVATE INPUT: $\mathcal{C}_{\mathsf{3Ext}}$ holds a private message $\boldsymbol{m} \in \{0,1\}^{\lambda}$.

*Commitment phase:* It consists of the following steps.

**Round 1** ($\mathcal{C}_{\mathsf{3Ext}} \to \mathcal{R}_{\mathsf{3Ext}}$)**.** $\mathcal{C}_{\mathsf{3Ext}}$ commits using Com to $\lambda$ pairs of strings $\{(\boldsymbol{v}_i^0, \boldsymbol{v}_i^1)\}_{i\in[\lambda]}$, such that, for each $i \in [\lambda]$, $(\boldsymbol{v}_i^0, \boldsymbol{v}_i^1) = (\boldsymbol{\nu}_i, \boldsymbol{m} \oplus \boldsymbol{\nu}_i)$ and $\boldsymbol{\nu}_i$ are random strings in $\{0,1\}^{|\boldsymbol{m}|}$.
Let $\boldsymbol{c}$ be the list of the resulting $\lambda$ pairs of commitments, $\mathcal{C}_{\mathsf{3Ext}}$ sends $\boldsymbol{c}$ to $\mathcal{R}_{\mathsf{3Ext}}$.

**Round 2** ($\mathcal{R}_{\mathsf{3Ext}} \to \mathcal{C}_{\mathsf{3Ext}}$)**.** Upon receiving $\boldsymbol{c}$ from $\mathcal{C}_{\mathsf{3Ext}}$, $\mathcal{R}_{\mathsf{3Ext}}$ sends a random challenge $\boldsymbol{e} = (e_1, \ldots, e_\lambda)$, with each $e_i \in \{0,1\}$, for $i \in [\lambda]$.

**Round 3** ($\mathcal{C}_{\mathsf{3Ext}} \to \mathcal{R}_{\mathsf{3Ext}}$)**.** Upon receiving $\boldsymbol{e}$ from $\mathcal{R}_{\mathsf{3Ext}}$, $\mathcal{C}_{\mathsf{3Ext}}$ opens the commitments $\boldsymbol{v}_i^{e_i}, i \in [\lambda]$.

*Decommitment phase: ($\mathcal{C}_{\mathsf{3Ext}} \to \mathcal{R}_{\mathsf{3Ext}}$):* $\mathcal{C}_{\mathsf{3Ext}}$ sends $\boldsymbol{m}$ and the opening for all $\lambda$ pairs of strings. $\mathcal{R}_{\mathsf{3Ext}}$ checks that $\boldsymbol{m} = \boldsymbol{v}_1^0 \oplus \boldsymbol{v}_1^1 = \ldots = \boldsymbol{v}_\lambda^0 \oplus \boldsymbol{v}_\lambda^1$.

Figure 4.2: Description of $\Pi_{3Ext} = (C_{3Ext}, R_{3Ext})$ [12]

with stateful information contained in the structs. Instances can be instantiated using the "new" function. Each round of the protocol, from commitment to decommitment (including the verification actions of the receiver), is encapsulated within distinct functions. Such a modular approach also allows for easier integration with Naor's scheme and $\Pi_{BGRVV}$ to build $\Pi_{5Ext}$. For example, the intermediate class "CommitmentScheme" served initially as a placeholder for Naor's scheme, exactly mirroring its state information. Then Naor's scheme directly substituted this placeholder class as the project progressed to the merging phase. The challenge string is generated pseudorandomly through "thread-rng", with seed automatically seeded from "OsRng" (which retrieves randomness from the operating system).

Finally, note that two elements need to be verified by $R_{3Ext}$ during the verification phase:

1. All the openings sent must be valid according to the random challenge string sent in round 2.
2. The correctness of the modular subtractions for each pair of $v_i^1$ and $v_i^0$ correlating to message $m$.

The first verification can easily be omitted since it is not explicitly stated in the protocol description. Initially, this was forgotten to be implemented and the error was spotted after the whole implementation for this protocol had been finished. If the verification process fails, the program will output the error message "verification failed" and then terminate, indicating the rejection by the verifier.

### Basic Setup

- Let $G$ be a cyclic group of prime order $q$ with generator $g$.
- The secret key $x$ is a random number chosen from $\{1, 2, \ldots, q-1\}$.
- The public key $y$ is computed as $y = g^x \mod q$, where $x$ is the secret key.

### The Protocol

1. **Commitment Phase**:
   - The prover selects a random value $r$ from $\{1, 2, \ldots, q-1\}$.
   - They compute a commitment $t = g^r \mod q$ and send $t$ to the verifier.
2. **Challenge Phase**:
   - The verifier sends a random challenge $c$ to the prover. The challenge $c$ is also chosen from $\{1, 2, \ldots, q-1\}$.
3. **Response Phase**:
   - The prover computes the response $s = r + cx \mod q$.
   - This response $s$ is sent to the verifier.

### Verification

- Upon receiving $s$, the verifier checks if $g^s \equiv t \cdot y^c \mod q$.
- This equation works because $g^s = g^{r+cx} = g^r \cdot g^{cx} = t \cdot y^c$.

Figure 4.3: Description of Schnorr's Protocol, note that x is the witness here.

## 4.2 Σ-protocol

For the 3-round public-coin Special Honest-Verifier Zero-Knowledge (SHVZK), we have implemented two Σ-protocols: Schnorr's protocol [39], and the more recent ZKBoo++ [11] protocol, each with unique strengths dealing with different types of statements. The choice between the protocols depends on the statement we aim to prove. A more detailed discussion of the difference between these protocols will be presented in the following section.

### 4.2.1 Schnorr's Protocol

**Overview of Schnorr's Protocol**. Recall that Schnorr's protocol is a simple Σ-protocol which follows 3-move format as shown in Figure 4.3. Briefly speaking, Schnorr's protocol utilises the property of modular calculation to prove the knowledge of the witness without revealing any information about the witness itself. It is widely used due to its remarkable efficiency since all calculations involved are simple, fast, and straightforward. For instance, it can be used to prove the knowledge of a discrete logarithm.

**Implementation Details**. The Schnorr protocol implementation features an interactive

design, with the prover, verifier, and each round's message encapsulated within separate classes for clarity and modularity. Within the prover's class, key elements such as the cyclic group's generator $g$, the witness $x$, and the randomness $r$ are organized into the prover's struct. The "Group" trait from the "ark-ec" crate is employed to handle the prime-order group generation and operations on group elements. Furthermore, the "ark-ec" crate offers the "AffineRepr" trait, which enables a unique representation of elliptic curve points, thus allowing operations on elements from elliptic curves over finite fields. For the second round message, we generate challenge value pseudorandomly using "thread-rng". Comprehensive tests for the protocol have been developed using random elements from the "BLS12-377 curve", utilising the "ark-bls12-377" crate. The corresponding error message will be printed in case of verification failure.

### 4.2.2 ZKBoo++

Although Schnorr's protocol is highly efficient and well-suited for proving arithmetic statements, its application is limited when it comes to proving statements from the general NP language, such as proving the preimage of a SHA-256 output. To make $\Pi_{NM}$ a general-purpose protocol capable of handling a broader range of proofs, we replace Schnorr's protocol with the optimised version of ZKBoo. ZKBoo is specifically designed to provide practically efficient zero-knowledge arguments for Boolean circuits [18]. Given that any NP language can be expressed as a combination of algebraic circuits by expressing the underlying relation as a circuit, ZKBoo is therefore a general-purpose protocol and satisfies our requirements.

**Overview of ZKBoo**. First, we recall how ZKBoo works. ZKBoo is based on the MPC-in-the-head paradigm of Ishai et al. [28]. ZKBoo extends the idea of [28] by replacing MPC with a proposed technique called "circuit decomposition". It improves the efficiency of cryptographic proofs by transitioning from multi-party computation (MPC) protocols to a more flexible framework. This approach involves breaking down a computation circuit, designed to evaluate a function $\phi(x) = y$ for a given relation $R$, into smaller, manageable components without the restrictions of MPC protocols. At its core, the decomposition involves a "Share" function that divides the input into three distinct shares for a fixed number of players, specifically set at three. It also includes three "Output" functions, each assigned to one of the players, which process all input shares alongside some randomness to generate individual output shares. Finally, a Reconstruct function aggregates these output shares to produce the final output of the circuit. This method is crafted to ensure both correctness, meaning the final output accurately reflects the computation of $\phi$ on $x$, and 2-privacy, a property that guarantees the secrecy of the witness x even if the views of any two players are disclosed, thereby preventing from leaking the sensitive information.

Then a circuit decomposition is used to generate the proof as follows: Given that the prover knows a secret witness $x$ that satisfies $y = \phi(x)$ for a given function $\phi$, while the verifier only knows $y$. In the Commit phase, the prover generates three random tapes $(k_1, k_2, k_3)$, runs a decomposition of the function $\phi$ three times to produce three views $(w_1, w_2, w_3)$ and corresponding output shares $(y_1, y_2, y_3)$. Each view, combined with its random tape, is committed to producing commitments $(c_1, c_2, c_3)$. These commitments

**Protocol $\Pi^*_\phi$**

Let $\phi : X \to Y$ be a function and $\mathcal{D}$ a related $(2,3)$-decomposition as defined in Definition 3.

Input: $\mathbf{x} \in X$

1. Sample random tapes $\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3$;
2. Compute $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \leftarrow \mathsf{Share}(\mathbf{x}; \mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3)$;
3. Let $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3$ be vectors with $N+1$ entries;
   - Initialize $\mathbf{w}_i[0] = \mathbf{x}_i$ for all $i \in \{1,2,3\}$;
   - For $j = 1, \ldots, N$, compute:
     - For $i = 1, 2, 3$, compute
       $$\mathbf{w}_i[j] = \phi_i^{(j)}\left((\mathbf{w}_m[0..j-1], \mathbf{k}_m)_{m \in \{i, i+1\}}\right)$$
4. Compute $\mathbf{y}_i = \mathsf{Output}_i(\mathbf{w}_i, \mathbf{k}_i)$ for $i \in \{1,2,3\}$;
5. Compute $\mathbf{y} = \mathsf{Rec}(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3)$;

Output: $\mathbf{y} \in Y$

Figure 4.4: Description of Circuit Decomposition in ZKBoo [18]

and the output shares are then sent to the verifier.

During the Prove phase, the verifier randomly selects an index e from $\{1,2,3\}$ and communicates it to the prover, who responds by opening the commitments adjacent to $e$ ($c_e$ and $c_{e+1 \mod 3}$), revealing the corresponding views and random tapes. This partial opening is the response $z$, which includes the necessary elements for the verifier to check the proof.

The Verify phase consists of the verifier conducting the following three checks:

1. The reconstructed output from the output shares matches the known value $y$.
2. The opened views need to correspond to the correct output shares.
3. The challenge was correctly formed according to the protocol.

If any of these checks fail, the verifier rejects the proof; otherwise, they accept it, concluding that the prover possesses the witness $x$ without actually learning what it is.

**Theorem 2** *[18] The ZKBoo protocol (Figure 4.5) is a $\Sigma$-protocol with 3-special soundness.*

**Implementation details**. We implemented the optimized version of ZKBoo called ZKBoo++ introduced in [11] for its efficiency. ZKBoo++ has introduced six optimisations (details can be found in Table B.1) over the original ZKBoo protocol and effectively reducing proof sizes by more than half without incurring additional costs [11]. Since implementing ZKBoo from scratch is massive work, considering the time constraints set, our work is therefore based on the available implementation from `https://github.com/geometryresearch/zkboo.git`. This repository provides a non-interactive version of ZKBoo++, utilizing the Fiat-Shamir (FS) heuristic [16]. To construct an interactive protocol, we modified the source code by removing the FS transform and subsequently employed the modules for the interactive prover and ver-

**ZKBoo Protocol**

The verifier and the prover have input $\mathbf{y} \in L_\phi$. The prover knows $\mathbf{x}$ such that $\mathbf{y} = \phi(\mathbf{x})$. A (2,3)-decomposition of $\phi$ is given. Let $\Pi^*_\phi$ be the protocol related to this decomposition.

**Commit:** The prover does the following:

1. Sample random tapes $\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3$;
2. Run $\Pi^*_\phi(\mathbf{x})$ and obtain the views $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3$ and the output shares $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3$;
3. Commit to $\mathbf{c}_i = \mathsf{Com}(\mathbf{k}_i, \mathbf{w}_i)$ for all $i \in [3]$;
4. Send $\mathbf{a} = (\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$.

**Prove:** The verifier choose an index $\mathbf{e} \in [3]$ and sends it to the prover. The prover answers to the verifier's challenge sending opening $\mathbf{c}_e, \mathbf{c}_{e+1}$ thus revealing $\mathbf{z} = (\mathbf{k}_e, \mathbf{w}_e, \mathbf{k}_{e+1}, \mathbf{w}_{e+1})$.

**Verify:** The verifier runs the following checks:

1. If $\mathsf{Rec}(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3) \neq \mathbf{y}$, output reject;
2. If $\exists\, i \in \{e, e+1\}$ s.t. $\mathbf{y}_i \neq \mathsf{Output}_i(\mathbf{w}_i)$, output reject;
3. If $\exists j$ such that
$$\mathbf{w}_e[j] \neq \phi_e^{(j)}(\mathbf{w}_e, \mathbf{w}_{e+1}, \mathbf{k}_e, \mathbf{k}_{e+1})$$
output reject;
4. Output accept;

Figure 4.5: Description of ZKBoo [18]

ifier. The biggest difference is that the challenge is now randomly generated by the verifier instead of by the prover from utilising the FS transform. Apart from this, the remainder of the codebase remains largely unchanged. While our modified interactive logic somewhat follows from the original ZKBoo protocol, we still report ZKBoo++ being implemented due to the implementation of optimisations from ZKBoo++ in the fundamental data structure.

Regarding the commitment scheme used, contrary to previous protocols where Naor's commitment scheme was implemented, this implementation employs SHA-256 as the commitment function, under the assumption that SHA-256 is a collision-resistant hash function and SHA-256(·, r) is a pseudo-random function (PRF) (with key r). This aligns with their original implementation, which implements a generic type commitment through the hash functions in the "sha3" crate, allowing for the use of any algorithm (hash functions) available within this crate. Replacing it with Naor's scheme would change the design of the entire structure, thus we decided to keep it. Specifically, we chose to employ the "Keccak256" algorithm from the "sha3" crate. Furthermore, The impact of this decision on comparative analysis with the baseline protocol $\Pi_{KLP}$ should be minimal since their protocol contains instantiation of OR-composition of Ligero which also uses SHA-256 as the hash function to commit.

The modular nature of our implementation permits the straightforward replacement of components which was previously based on Schnorr's protocol with the interactive ZKBoo++. The main difference lies in how to set the witness, as ZKBoo++ requires to build a circuit representation for the statements being proved first. Benchmark tests were conducted separately for both versions of $\Pi_{NM}$.

# Chapter 5

# Results and Discussion

In this chapter, we report and discuss the results from the benchmark tests and compare them with the baseline protocol $\Pi_{KLP}$.

## 5.1 Results Comparison

The benchmark tests were conducted on a machine equipped with an AMD Ryzen 9 5900HX (3.30GHz) processor. For $\Pi_{NM}$ with Schnorr's protocol as SHVZK, the witness has been set to a random scalar field element from "BLS12-377" elliptic curve, and the benchmarking results under synchronise settings are shown in Table 5.1, where k is the size of tag used in weak non-malleable commitment $\Pi_{BGRVV}$, $\lambda$ is the security parameter and $\beta$ is the maximum length of the message the corresponding tag can handle (calculated as $\beta = 2*(k+1)-1$ [7]).

Table 5.1: Average Benchmarking Results for $\Pi_{NM}$ with Schnorr's Protocol

| $(k,\lambda)$ | Avg P Time (ms) | Avg P Outlier rate% | Avg V Time (ms) | Avg V Outlier rate% | Comm. (MB) | $\beta(F)$ |
|---|---|---|---|---|---|---|
| (16,128) | 1.1722 | 7.0% | 1.1858 | 8.3% | 0.094698 | 33 |
| (32,128) | 3.6677 | 5.7% | 3.6943 | 5.3% | 0.351978 | 65 |
| (64,128) | 13.6560 | 5.3% | 13.5593 | 5.7% | 1.358058 | 129 |

For general-purpose protocol with ZKBoo++, we used $\Pi_{NM}$ to prove the preimage of a SHA-256 output, and the results are shown in Table 5.2.

Table 5.2: Averages Benchmarking Results for $\Pi_{NM}$ with ZKBoo++

| $(k,\lambda)$ | Avg P Time (ms) | Avg P Outlier rate% | Avg V Time (ms) | Avg V Outlier rate% | Comm. (MB) | $\beta(F)$ |
|---|---|---|---|---|---|---|
| (16,40) | 51.612 | 3.33 | 38.859 | 6.67 | 1.3509 | 33 |
| (32,40) | 52.172 | 8.00 | 39.372 | 7.67 | 1.6082 | 65 |
| (64,40) | 63.193 | 1.00 | 50.466 | 4.00 | 2.6143 | 129 |
| (16,80) | 100.478 | 4.33 | 70.820 | 7.00 | 2.5891 | 33 |
| (32,80) | 106.543 | 8.33 | 76.896 | 4.67 | 2.8464 | 65 |
| (64,80) | 112.923 | 8.00 | 83.979 | 7.33 | 3.8525 | 129 |
| (16,128) | 157.64 | 11.67 | 116.343 | 8.00 | 4.0823 | 33 |
| (32,128) | 166.393 | 6.33 | 121.88 | 6.00 | 4.3396 | 65 |
| (64,128) | 175.367 | 2.00 | 128.703 | 5.00 | 5.3456 | 129 |

The benchmarks were conducted three times for each parameter set, with each trial comprising a sample size of 100. Detailed results of these benchmarks are presented in Appendix A. The reported averages were derived from these three tests. A confidence level of 0.95 was used to identify outliers; that is, any sample point falling outside the confidence interval was classified as an outlier. For instance, with the parameter set (16,40), the outlier rate for the prover averaged 3.33%. This indicates that among the 300 test samples, $300 \times 3.33\% = 10$ elements were identified as outliers. A lower outlier rate not only signifies greater stability in the results but also implies higher confidence in the protocol's performance when deployed in an online environment.

We compared our results with the results from Table 1 of [30] and created the Table 5.3. The results were also visualised in Figure 5.1 and Figure 5.2. From the comparison, it is clear that $\Pi_{NM}$ has made a significant improvement compared to $\Pi_{KLP}$ both in terms of the communication complexity and the execution time for the prover and verifier. Taking (64,80) as an example, $\Pi_{NM}$ takes 112.923ms for prover and 83.979 ms for verifier with communication $= 3.8525MB$ to prove a witness of SHA-256, while $\Pi_{KLP}$ takes over 5000ms for prover and 2000ms for verifier and requires communication of $28.84MB$ to achieve the same security level. This indicates an average of $35\times$ improvement of execution time over the prover and verifier and about $7.5\times$ improvement of communication sizes.

Table 5.3: Comparative Benchmarking Results with $\Pi_{KLP}$

| Params. | $\Pi_{NM}$ | | | $\Pi_{KLP}$ | | |
|---|---|---|---|---|---|---|
| $(k,\lambda)$ | Avg P Time (ms) | Avg V Time (ms) | Comm. (MB) | P Time (ms) | V Time (ms) | Comm. (MB) |
| (32,40) | 52.172 | 39.372 | 1.6082 | 1,680 | 740 | 19.68 |
| (32,80) | 106.543 | 76.896 | 2.8464 | 3,560 | 1,490 | 24.88 |
| (64,80) | 112.923 | 83.979 | 3.8525 | 5,040 | 2,230 | 28.84 |

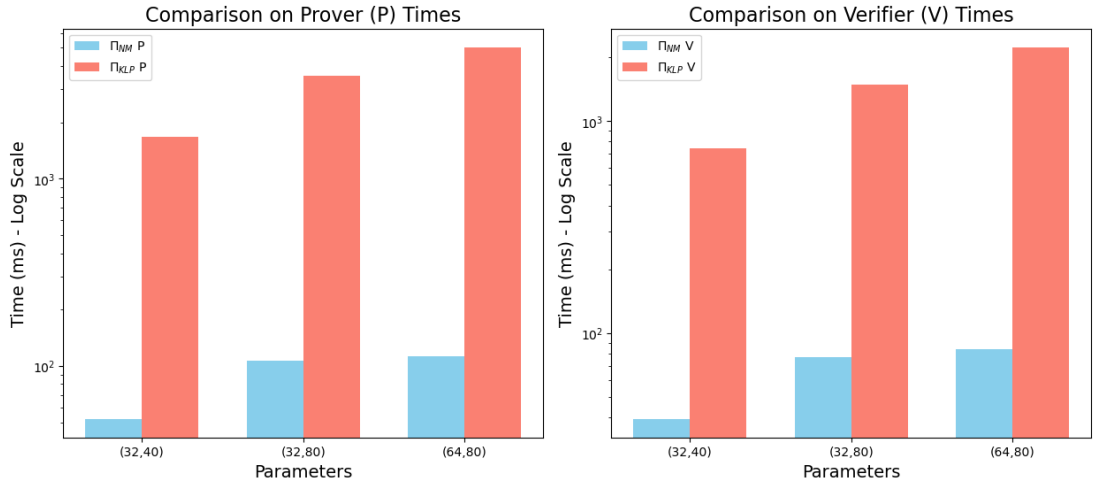


Figure 5.1: Comparative Benchmarking Results for Prover and Verifier Execution Times. The left plot is prover execution time comparison in log scale while the right plot is the verifier execution time comparison in log scale.

These results are consistent with the theoretical analysis in section 1.3 of [12], proving

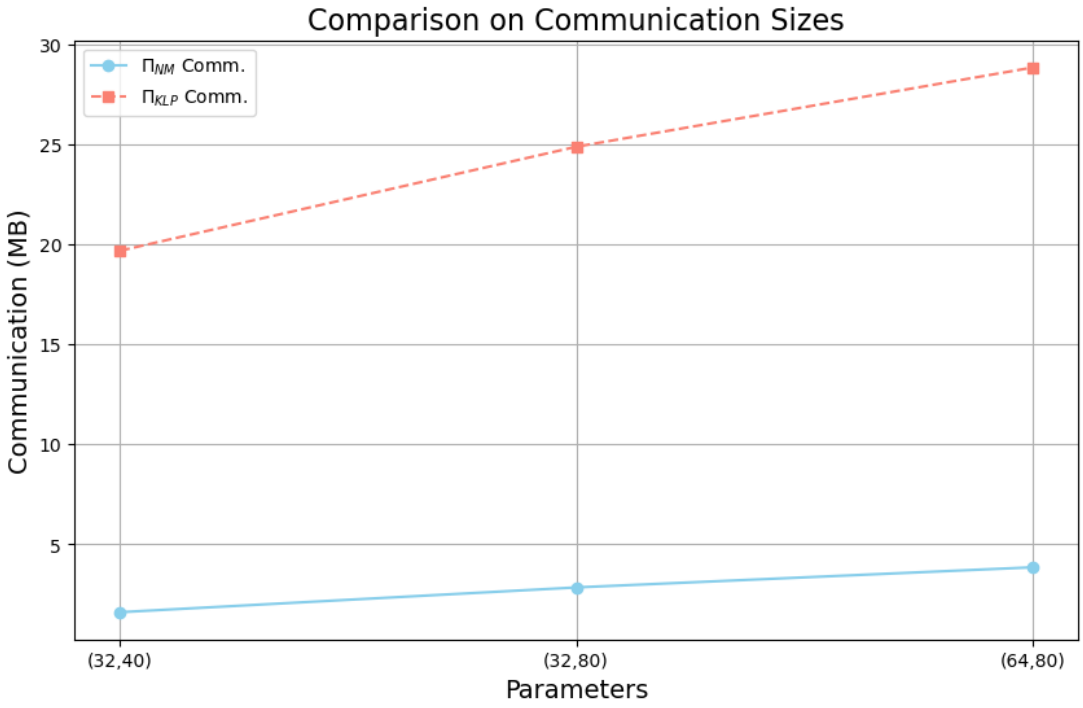


Figure 5.2: Comparison of Communication Sizes

the successful implementation of the NMZK protocol $\Pi_{NM}$. Additionally, the practical outcomes also further validate the analytical results.

## 5.2 Efficiency Analysis

Generally speaking, the computation costs required by a single run of $\Pi_{NM}$ are mainly contributed from the following three components:

- A single run of $\Pi_{SHVZK}$ (Schnorr's protocol or ZKBoo++)
- A single run of $\Pi_{3Ext}$
- A single run of $\Pi^{3R}_{BGRVV}$

Among these, $\Pi_{3Ext}$ and $\Pi^{3R}_{BGRVV}$ require black-box use of any one-way function only and can be instantiated efficiently [30, 7]. Therefore, the efficiency of $\Pi_{NM}$ primarily depends on the instantiation 3-round public-coin honest-verifier zero-knowledge protocol ($\Pi_{SHVZK}$). When $\Pi_{SHVZK}$ employs ZKB++ for proving the preimage of an SHA-256 output, a singular execution of ZKB++ dominates the computational costs of the construction; In scenarios such as proving knowledge of a discrete logarithm via Schnorr's protocol, the computational costs of the framework are primarily governed by $\Pi_{3Ext}$ and $\Pi_{3RBGRRV}$, thereby avoiding the need for computationally expensive protocols like ZKBoo/ZKBoo++/Ligero. This contrasts sharply with $\Pi_{KLP}$, which requires multiple executions of Ligero regardless of the nature of the statements being proved, leading to these executions overwhelmingly contributing to the protocol's computational expenses.

Moreover, $\Pi_{NM}$ is designed to complete within a maximum of 9 rounds without specific optimizations, whereas $\Pi_{KLP}$ extends beyond 28 rounds. It is expected that $\Pi_{NM}$ would have more efficient executions when employed on the Internet due to the better round complexity.

# Chapter 6

# Conclusion and Future Work

In this dissertation, we present an in-depth discussion of the protocol $\Pi_{NM}$, introduced by Dr. Michele et al. [12]. This includes a comprehensive overview of the necessary theoretical background information, alongside a detailed discussion of the important implementation decisions made and the details of the sub-protocols for which the author was responsible. Comparative benchmark tests were conducted against the first non-black-box NMZK protocols proposed recently. The empirical findings indicate that $\Pi_{NM}$ significantly improves the performance in both computation time and communication complexities.

Future work on the project would include further improving the codebase, we believe there are still lots of minor optimizations that can be made regarding the structure of the implementation. The impact of increasing the number of threads could also be investigated. Additionally, employing the protocol on the Internet should be considered.

In general, the results of the work of [12] and this dissertation not only address a significant gap in cryptographic literature but also establish a new benchmark for the design and application of NMZK protocols, potentially catalyzing further exploration and innovation in the field.

# Bibliography


[1] Scott Ames et al. “Ligero: Lightweight Sublinear Arguments Without a Trusted Setup”. In: CCS ’17. Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 2087–2104. ISBN: 9781450349468. DOI: 10.1145/3133956.3134104. URL: https://doi.org/10.1145/3133956.3134104.

[2] B. Barak. “Constant-round coin-tossing with a man in the middle or realizing the shared random string model”. In: *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.* 2002, pp. 345–355. DOI: 10.1109/SFCS.2002.1181957.

[3] B. Barak. “How to go beyond the black-box simulation barrier”. In: *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. 2001, pp. 106–115. DOI: 10.1109/SFCS.2001.959885.

[4] B. Barak and O. Goldreich. “Universal arguments and their applications”. In: *Proceedings 17th IEEE Annual Conference on Computational Complexity*. 2002, pp. 194–203. DOI: 10.1109/CCC.2002.1004355.

[5] Boaz Barak, Manoj Prabhakaran, and Amit Sahai. “Concurrent Non-Malleable Zero Knowledge”. In: *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS’06)*. 2006, pp. 345–354. DOI: 10.1109/FOCS.2006.21.

[6] Eli Ben-Sasson et al. “On the concrete efficiency of probabilistically-checkable proofs”. In: *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*. STOC ’13. Palo Alto, California, USA: Association for Computing Machinery, 2013, pp. 585–594. ISBN: 9781450320290. DOI: 10.1145/2488608.2488681. URL: https://doi.org/10.1145/2488608.2488681.

[7] Hai Brenner et al. “Fast Non-Malleable Commitments”. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. CCS ’15. Denver, Colorado, USA: Association for Computing Machinery, 2015, pp. 1048–1057. ISBN: 9781450338325. DOI: 10.1145/2810103.2813721. URL: https://doi.org/10.1145/2810103.2813721.

[8] Brandon Broadnax et al. “Concurrently Composable Security with Shielded Super-Polynomial Simulators”. In: *Advances in Cryptology – EUROCRYPT 2017*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Cham: Springer International Publishing, 2017, pp. 351–381. ISBN: 978-3-319-56620-7.

[9] R. Canetti. “Universally composable security: a new paradigm for cryptographic protocols”. In: *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. 2001, pp. 136–145. DOI: 10.1109/SFCS.2001.959888.

[10] Ran Canetti et al. “Universally composable two-party and multi-party secure computation”. In: *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*. STOC ’02. Montreal, Quebec, Canada: Association for Computing Machinery, 2002, pp. 494–503. ISBN: 1581134959. DOI: 10.1145/509907.509980. URL: https://doi.org/10.1145/509907.509980.

[11] Melissa Chase et al. “Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’17. Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 1825–1842. ISBN: 9781450349468. DOI: 10.1145/3133956.3133997. URL: https://doi.org/10.1145/3133956.3133997.

[12] Michele Ciampi et al. *Black-Box (and Fast) Non-Malleable Zero Knowledge (Unpublished)*. 2024.

[13] Michele Ciampi et al. “Four-Round Concurrent Non-Malleable Commitments from One-Way Functions”. In: *Advances in Cryptology – CRYPTO 2017*. Ed. by Jonathan Katz and Hovav Shacham. Cham: Springer International Publishing, 2017, pp. 127–157. ISBN: 978-3-319-63715-0.

[14] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. “Proofs of partial knowledge and simplified design of witness hiding protocols”. In: *Annual International Cryptology Conference*. Springer. 1994, pp. 174–187.

[15] Danny Dolev, Cynthia Dwork, and Moni Naor. “Non-malleable cryptography”. In: *Proceedings of the twenty-third annual ACM symposium on Theory of computing*. 1991, pp. 542–552.

[16] Amos Fiat and Adi Shamir. “How To Prove Yourself: Practical Solutions to Identification and Signature Problems”. In: *Advances in Cryptology — CRYPTO’ 86*. Ed. by Andrew M. Odlyzko. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 186–194. ISBN: 978-3-540-47721-1.

[17] David Gabay, Kemal Akkaya, and Mumin Cebe. “Privacy-preserving authentication scheme for connected electric vehicles using blockchain and zero knowledge proofs”. In: *IEEE Transactions on Vehicular Technology* 69.6 (2020), pp. 5760–5772.

[18] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. “ZKBoo: Faster Zero-Knowledge for Boolean Circuits”. In: *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016, pp. 1069–1083. ISBN: 978-1-931971-32-4. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/giacomelli.

[19] Oded Goldreich. *The Foundations of Cryptography*. Vol. Volume 1: Basic Techniques. Cambridge University Press, 2001.

[20] Shafi Goldwasser, Silvio Micali, and Chales Rackoff. “The knowledge complexity of interactive proof-systems”. In: *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 203–225. ISBN: 9781450372664. URL: https://doi.org/10.1145/3335741.3335750.

[21] Vipul Goyal, Omkant Pandey, and Silas Richelson. “Textbook non-malleable commitments”. In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. 2016, pp. 1128–1141.

[22] Vipul Goyal et al. “An algebraic approach to non-malleability”. In: *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE. 2014, pp. 41–50.

[23] Vipul Goyal et al. “Constructing Non-malleable Commitments: A Black-Box Approach”. In: *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*. 2012, pp. 51–60. DOI: 10.1109/FOCS.2012.47.

[24] Louis C Guillou and Jean-Jacques Quisquater. “A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory”. In: *Advances in Cryptology—EUROCRYPT’88: Workshop on the Theory and Application of Cryptographic Techniques Davos, Switzerland, May 25–27, 1988 Proceedings 7*. Springer. 1988, pp. 123–128.

[25] R. Impagliazzo. “A personal view of average-case complexity”. In: *Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference*. 1995, pp. 134–147. DOI: 10.1109/SCT.1995.514853.

[26] Yuval Ishai et al. “Black-box constructions for secure computation”. In: *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*. 2006, pp. 99–108.

[27] Yuval Ishai et al. “Efficient non-interactive secure computation”. In: *Advances in Cryptology–EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings 30*. Springer. 2011, pp. 406–425.

[28] Yuval Ishai et al. “Zero-knowledge from secure multiparty computation”. In: *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 2007, pp. 21–30.

[29] Dakshita Khurana, Rafail Ostrovsky, and Akshayaram Srinivasan. “Round Optimal Black-Box “Commit-and-Prove””. In: *Theory of Cryptography*. Ed. by Amos Beimel and Stefan Dziembowski. Cham: Springer International Publishing, 2018, pp. 286–313. ISBN: 978-3-030-03807-6.

[30] Allen Kim, Xiao Liang, and Omkant Pandey. “A New Approach to Efficient Non-Malleable Zero-Knowledge”. In: *Advances in Cryptology – CRYPTO 2022*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Cham: Springer Nature Switzerland, 2022, pp. 389–418. ISBN: 978-3-031-15985-5.

[31] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. “Concurrent non-malleable commitments from any one-way function”. In: *Theory of Cryptography: Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008. Proceedings 5*. Springer. 2008, pp. 571–588.

[32] Daniele Micciancio and Erez Petrank. “Simulatable commitments and efficient concurrent zero-knowledge”. In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2003, pp. 140–159.

[33] Moni Naor. “Bit Commitment Using Pseudo-Randomness”. In: *Advances in Cryptology — CRYPTO’ 89 Proceedings*. Ed. by Gilles Brassard. New York, NY: Springer New York, 1990, pp. 128–136. ISBN: 978-0-387-34805-6.

[34] Moni Naor. “Bit commitment using pseudorandomness”. In: *Journal of cryptology* 4 (1991), pp. 151–158.

[35] Rafail Ostrovsky, Omkant Pandey, and Ivan Visconti. “Efficiency preserving transformations for concurrent non-malleable zero knowledge”. In: *Theory of Cryptography Conference*. Springer. 2010, pp. 535–552.

[36] Somnath Panja and Bimal Kumar Roy. “A secure end-to-end verifiable e-voting system using zero knowledge based blockchain”. In: *Cryptology ePrint Archive* (2018).

[37] Rafael Pass and Alon Rosen. “New and improved constructions of non-malleable cryptographic protocols”. In: STOC ’05. Baltimore, MD, USA: Association for Computing Machinery, 2005, pp. 533–542. ISBN: 1581139608. DOI: 10.1145/1060590.1060670. URL: https://doi.org/10.1145/1060590.1060670.

[38] Rafael Pass and Hoeteck Wee. “Black-box constructions of two-party protocols from one-way functions”. In: *Theory of Cryptography: 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings 6*. Springer. 2009, pp. 403–418.

[39] Claus-Peter Schnorr. “Efficient signature generation by smart cards”. In: *Journal of cryptology* 4 (1991), pp. 161–174.

[40] Wei Wu et al. “Blockchain based zero-knowledge proof of location in iot”. In: *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE. 2020, pp. 1–7.

# Appendix A

# Details of Benchmark Tests

This appendix includes the details of all benchmarks conducted on $\Pi_{NM}$.

Table A.1: Detailed Benchmarking Results for $\Pi_{NM}$ with Schnorrs's Protocol

| $(k,\lambda)$ | P Time (ms) | P Outlier rate% | V Time (ms) | V Outlier rate% | Comm. (MB) | $\beta(F)$ |
|---|---|---|---|---|---|---|
| (16,128) | 1.1582 | 4% | 1.1817 | 12% | 0.094698 | 33 |
| | 1.1629 | 8% | 1.1901 | 6% | | |
| | 1.1954 | 9% | 1.1856 | 7% | | |
| (32,128) | 3.6638 | 7% | 3.5820 | 5% | 0.351978 | 65 |
| | 3.5961 | 8% | 3.6521 | 7% | | |
| | 3.7433 | 2% | 3.8489 | 4% | | |
| (64,128) | 13.742 | 4% | 13.523 | 9% | 1.358058 | 129 |
| | 13.649 | 6% | 13.212 | 4% | | |
| | 13.577 | 6% | 13.943 | 4% | | |

Table A.2: Detailed Benchmarking Results for $\Pi_{NM}$ with ZKBoo++

| $(k,\lambda)$ | P (ms) | P Outlier rate% | V (ms) | V Outlier rate% | Comm. (MB) | $\beta(F)$ |
|---|---|---|---|---|---|---|
| (16,40) | 49.189 | 4% | 36.551 | 10% | 1.3509339 | 33 |
| | 54.496 | 2% | 41.299 | 8% | | |
| | 51.152 | 4% | 38.728 | 2% | | |
| (32,40) | 52.355 | 6% | 38.205 | 13% | 1.6082139 | 65 |
| | 51.218 | 14% | 39.559 | 5% | | |
| | 52.942 | 4% | 40.352 | 5% | | |
| (64,40) | 64.958 | 0% | 53.660 | 2% | 2.614294 | 129 |
| | 62.085 | 3% | 48.838 | 5% | | |
| | 62.537 | 0% | 48.899 | 5% | | |
| (16,80) | 103.50 | 5% | 73.143 | 1% | 2.589146 | 33 |
| | 99.193 | 5% | 69.998 | 12% | | |
| | 98.741 | 3% | 69.319 | 8% | | |
| (32,80) | 110.59 | 7% | 80.564 | 4% | 2.846426 | 65 |
| | 106.76 | 11% | 76.604 | 7% | | |
| | 102.28 | 7% | 73.519 | 3% | | |
| (64,80) | 113.28 | 10% | 85.025 | 3% | 3.852506 | 129 |
| | 113.39 | 7% | 83.754 | 8% | | |
| | 112.10 | 7% | 83.158 | 11% | | |
| (16,128) | 161.05 | 14% | 118.96 | 9% | 4.082284 | 33 |
| | 157.48 | 12% | 118.29 | 5% | | |
| | 154.39 | 9% | 111.78 | 10% | | |
| (32,128) | 174.77 | 11% | 128.20 | 9% | 4.339564 | 65 |
| | 158.34 | 7% | 114.55 | 9% | | |
| | 166.07 | 1% | 122.89 | 0% | | |
| (64,128) | 169.23 | 5% | 132.02 | 0% | 5.345644 | 129 |
| | 183.59 | 1% | 126.04 | 10% | | |
| | 173.28 | 0% | 128.05 | 5% | | |

# Appendix B

# Optimisation on ZKBoo++ over ZKBoo

ZKBoo++ introduced in [11] has made six optimisations over the original ZKBoo [18] which successfully reduces the proof size by half while maintaining the same level of computation costs. The optimisations made are summarised below:

| **Optimization Made** | **Summary** |
|---|---|
| O1: The Share Function | Shares are sampled pseudorandomly, enhancing compact proofs by requiring the prover to include only the seed for verification, leading to an expected reduction in proof size. |
| O2: Not Including Input Shares | Input shares derived pseudorandomly from a seed no longer needs to be included in every view, reducing the average number of input shares sent. |
| O3: Not Including Commitments | Commitments for two of the three views can be recomputed by the verifier, eliminating the need to send them and instead relying on challenge computation for verification. |
| O4: No Additional Randomness for Commitments | The seed value for random tape also serves as randomization for commitments, ensuring they are hiding without additional randomness. |
| O5: Not Including the Output Shares | Output shares are not sent as they can be deterministically computed from other proof components, reducing redundancy and relying on the verifier's computation for efficiency. |
| O6: Not Including $View_e$ | The verifier does not check every wire in $View_e$ but recomputes and verifies using commitments, significantly reducing the proof size and computational time. |

Table B.1: Summary of Optimizations