A PASSWORD-AUTHENTICATED THRESHOLD-BASED GROUP KEY AGREEMENT PROTOCOL LEVERAGING ZERO-KNOWLEDGE PROOFS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ADNAN KILIÇ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

JULY 2024

Approval of the thesis:

A PASSWORD-AUTHENTICATED THRESHOLD-BASED GROUP KEY AGREEMENT PROTOCOL LEVERAGING ZERO-KNOWLEDGE PROOFS

submitted by **ADNAN KILIÇ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Naci Emre Altun
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Prof. Dr. Ertan Onur
Supervisor, **Computer Engineering, METU** _____

Dr. Cansu Betin Onur
Co-supervisor, **Institute of Applied Mathematics, METU** _____

Examining Committee Members:

Prof. Dr. İbrahim Körpeoğlu
Computer Engineering, Bilkent University _____

Prof. Dr. Ertan Onur
Computer Engineering, METU _____

Assoc. Prof. Dr. Pelin Angın Ülküer
Computer Engineering, METU _____

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

Assoc. Prof. Dr. Oğuz Yayla
Institute of Applied Mathematics, METU _____

Date:22.07.2024

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, Surname: Adnan KILIÇ

Signature :

ABSTRACT

A PASSWORD-AUTHENTICATED THRESHOLD-BASED GROUP KEY AGREEMENT PROTOCOL LEVERAGING ZERO-KNOWLEDGE PROOFS

KILIÇ, Adnan
Ph.D., Department of Computer Engineering
Supervisor: Prof. Dr. Ertan Onur
Co-Supervisor: Dr. Cansu Betin Onur

July 2024, 94 pages

The challenge we address in this thesis is to design a peer-to-peer threshold-based group key generation protocol where there is no centralized authority and peers do not learn anything about the generated key other than their shares albeit they are involved in the key generation. To address this challenge, we propose a password-authenticated threshold-based group key agreement protocol that resembles secret sharing. However, the construction of our proposal is the opposite. In legacy secret-sharing schemes, a trusted third party shares a secret value that it initially knows. In our proposal, principals collaborate to create shares without the knowledge of the constructed group key in a peer-to-peer fashion. We employ zero-knowledge protocols for verifying the authenticity of principals' contributions to the protocol. Zero-knowledge proofs help principals verify the veracity of data without sharing the data. In addition to the threshold-based group key agreement protocol, we also propose a novel zero-knowledge proof scheme based on equal partition problems on groups and prove that this novel zero-knowledge proof scheme is secure. We present the security and performance analysis of the proposed protocols.

Keywords: threshold cryptography, password-authenticated key agreement, zero-knowledge proof

ÖZ

SIFIR BİLGİ İSPATLARINDAN YARARLANAN PAROLA DOĞRULAMALI EŞİK TABANLI GRUP ANAHTAR ANLAŞMA PROTOKOLÜ

KILIÇ, Adnan
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Ertan Onur
Ortak Tez Yöneticisi: Dr. Cansu Betin Onur

Temmuz 2024 , 94 sayfa

Bu tezde ele aldığımız zorluk, merkezi bir otorite olmadan, paydaşların kendi payları dışında herhangi bir bilgi öğrenemedikleri, eşik-tabanlı grup anahtarı oluşturma protokolü tasarlamaktır. Bu zorluğu ele alan, sır paylaşma prensibini andıran bir yapıya sahip parola-doğrulamalı, eşik-tabanlı grup anahtar anlaşma protokolü öneriyoruz. Ancak önerdiğimiz protokol yapısı itibariyle geleneksel sır paylaşma protokollerinden ayrılmaktadır. Geleneksel sır paylaşma protokollerinde, güvenilir bir üçüncü taraf başlangıçta bildiği bir sırrı protokolün paydaşlarıyla paylaşır. Bizim önerdiğimiz protokolde ise, paydaşlar grup anahtarını oluşturmak üzere birbirlerinee kendi paylarını aktararak işbirliği yaparlar. Paydaşların protokole katkılarının doğruluğunu kanıtlamak için sıfır bilgi ispatlarını kullanıyoruz. Sıfır bilgi ispatları, paydaşların verileri birbirleriyle paylaşmadan doğruluğunu kanıtlamalarına yardımcı olmaktadır. Eşik-tabanlı grup anahtar anlaşma protokolünün yanı sıra, gruplar üzerinde eşit bölünme problemine dayanan yeni sıfır bilgi ispatları öneriyoruz. Ayrıca önerilen sıfır bilgi ispatlarının güvenli olduğunu kanıtlıyoruz. Önerilen protokollerin güvenli ve uygula-

mada verimli olduğunu gösteren güvenlik ve performans analizlerini sunuyoruz.

Anahtar Kelimeler: eşik kriptografi, parola doğrulamalı anahtar anlaşma, sıfır-bilgi ispatı



To my mum

ACKNOWLEDGMENTS

I want to express my sincere gratitude to my advisors, Prof. Dr. Ertan Onur and Dr. Cansu Betin Onur, for their invaluable guidance, encouragement, support, and patience. Their friendly attitude and understanding kept me motivated, especially during challenging moments in the theoretical aspects of my work.

A special thanks goes to my friends Abdullah Alchihabi, Çağlar Seylan, Eda Ceren Güngör, Ezgi Ekiz Emiroğlu, Hüsnü Yıldız, Kadir Ekmekci, Koray Uzun, Ömer Ekmekci, Özgür Kaya, and Sibel Soylu. Their friendship, support, and encouragement were indispensable in finding the motivation to complete this thesis.

I am also grateful to my colleagues Ali Işıklı, Berkan Ufuk, Evren Eser, Halil Kemal Taşkın, Murat Demircioğlu, and others—an outstanding group of individuals who provided unwavering support and alleviated any concerns.

Heartfelt appreciation goes to my family - my mom Hatice Kılıç, my sister Fatma Kılıç, and my brother Cengiz Kılıç. Their constant love and support have been my pillars of strength.

Lastly, I offer my heartfelt gratitude to my father. Facing the challenges of sustaining creativity in his absence has been tough, and I earnestly hope he would be proud of my accomplishments.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xv

LIST OF FIGURES xvi

LIST OF ABBREVIATIONS xvii

CHAPTERS

1 INTRODUCTION 1

1.1 Password-Authenticated Threshold Group Key Share (PATS) 1

1.2 Zero-knowledge Proofs 3

1.3 Motivations and Challenges 4

1.4 Our Contributions 6

1.5 Outline 7

2 RELATED WORK 9

2.1 Password-Authenticated Key Exchange 9

2.1.1 Password-Authenticated Group Key Exchange (PGKE) Protocols 9

2.1.2 Threshold Secret Sharing 10

2.1.3 Group Key Agreement (GKA) Protocols 12

2.2 Zero-Knowledge Protocols 13

2.2.1 Performance Metrics 14

2.2.2 Code-Based Cryptographic Identification Schemes 17

3 PATS: PASSWORD-AUTHENTICATED (2,3)-THRESHOLD KEY EXCHANGE 19

3.1 PATS Inspired by PAKE 19

3.2 Design Challenges 20

3.3 Distributed (2,3)-PATS 21

3.3.1 Distributed (2,3)-PATS in Finite Fields 21

3.3.2 Distributed (2,3)-PATS using Elliptic Curves 24

3.4 TTP-verified (2,3)-PATS 25

3.5 (t, n)-PATS 25

4 ZERO-KNOWLEDGE PROOF SCHEMES ON EQUAL PARTITION ON GROUPS PROBLEM 29

4.1 Equal Partition Problem on Groups (EPPG) 29

4.2 Three-pass ZKP Based on Equal Partition Problem (EPPZKP) 31

4.2.1 Setup Phase 31

4.2.2 EPPZKP on Elliptic Curves 33

4.3 Two Challenge ZKP Based on Equal Partition Problem (2EPPZKP) . 34

5 SECURITY AND PERFORMANCE ANALYSIS 37

5.1 Security Analysis of PATS 37

5.1.1 Adversary Model 37

5.1.2 Security Requirements 38
5.1.3 Security Analysis 39
5.2 Performance Analysis of PATS 43
5.2.1 PATS in Finite Fields 43
5.2.1.1 Computational Load 43
5.2.1.2 Run-time Analysis 45
5.2.2 PATS using Elliptic Curves 46
5.2.2.1 Computational Load 46
5.2.2.2 Run-time Analysis 48
5.2.3 (t, n)-PATS 49
5.2.3.1 Computational Load 49
5.2.4 Limitations 51
5.2.5 Comparison 52
5.3 Analysis of Zero-Knowledge Protocols 53
5.3.1 Security Analysis 53
5.3.1.1 Security Requirements 53
5.3.1.2 Security Analysis of EPPZKP 55
5.3.1.3 Security Analysis of 2EPPZKP 60
5.3.2 Performance Analysis 66
5.3.2.1 Computational Performance 66
5.3.2.2 Experimental Performance 72
6 CONCLUSION 75
6.1 Future Work 76

APPENDICES

A 77

A.1 Notation 77

A.2 Zero-knowledge Argument of Knowledge 77

A.3 Commitment Schemes 80

REFERENCES 83

CURRICULUM VITAE 93

LIST OF TABLES

TABLES

Table 2.1 Comparison of the state-of-the-art ZKP schemes based on various performance metrics where C is the number of gates in the circuit with depth d, and n is the size of its input. 16

Table 5.1 The analysis of computational cost of PATS in Finite Fields. 45

Table 5.2 The analysis of the computational cost of PATS using elliptic curves. 48

Table 5.3 The analysis of the computational cost of (t, n)-PATS in Finite Fields. 52

Table 5.4 Comparable security strengths, and run-time of finite-field cryptography and curves. 53

Table 5.5 Query pairs and answers. 65

Table 5.6 The analysis of the computational cost of EPPZKP. 69

Table 5.7 The analysis of the computational cost of EPPZKP on elliptic curves. 70

Table 5.8 The analysis of the computational cost of 2EPPZKP. 71

Table 5.9 Comparable security strengths, and run-time of EPPZKP for the selected curves. 72

LIST OF FIGURES

FIGURES

Figure 1.1 PATS scenario for the time capsule. 2

Figure 3.1 Distributed (2,3)-PATS in finite fields. 22

Figure 3.2 TTP-verified (2,3)-PATS in finite fields. 26

Figure 4.1 EPPZKP: Three-pass ZKP Based on Equal Partition Problem on Groups. 33

Figure 4.2 Two Challenge ZKP Based on Equal Partition Problem (2EP-PZKP) 36

Figure 5.1 Run-time of (2,3)-PATS in finite fields, the results of 100 experiments within the 95% confidence interval are shown. 46

Figure 5.2 Run-time of PATS using ECC, the results of 100 experiments within the 95% confidence interval are shown in the figure. 49

Figure 5.3 Run-time of EPPZKP using ECC. 73

LIST OF ABBREVIATIONS

2EPPZKP	Two Challenge ZKP Based on Equal Partition Problem
3SAT	Boolean satisfiability problem
AC	Arithmetic Circuit
AKE	Authenticated Key Exchange
CGKA	Centralized Group Key Agreement
Comm	Commitment
CRT	Chinese Remainder Theorem
DDH	Decision Diffie-Hellman
DGKA	Distributed Group Key Agreement
E	Edge
ECC	Elliptic Curve Cryptography
ECRH	Extractable Collision-Resistant Hash
EPP	Equal Partition Problem
EPPG	Equal Partition Problem on Groups
EPPZKP	Three-pass ZKP Based on Equal Partition Problem
FFC	Finite-Field Cryptography
G	Graph
G3C	Graph Three-Colorability
GI	Graph Isomorphism
GKA	Group Key Agreement
HSP	The Hidden Subgroup Problem
IOP	Interactive Oracle Proofs
IoT	Internet of Things
IP	Interactive Proof

J-PAKE	Password Authenticated Key Exchange by Juggling
KDF	Key Derivation Function
LWE	Learning with errors
MAC	Media Access Control
NIST	The National Institute of Standards and Technology
NIZK	Non-interactive Zero-Knowledge
NP	Nondeterministic Polynomial Time
PAKE	Password-Authenticated Key Exchange
PATS	Password-authenticated Threshold Group Key Share
PE	Password Equivalent
PGKE	Password-Authenticated Group Key Exchange
PKI	Public Key Infrastructure
R1CS	Rank-1 Constraint Satisfaction
SAE	Simultaneous Authentication of Equals
SCDH	Square Computational Diffie-Hellman Assumption
SIS	Short Integer Solution
SNARK	Succinct Non-Interactive ARgument of Knowledge
STARK	Scalable Transparent Zero-knowledge Argument of Knowledge
SVP	Shortest Vector Problem
TTP	Trusted Third Party
UANET	Unmanned Aerial Vehicle (UAV) Ad Hoc Network
V	Vertex
VSS	Verifiable Secret Sharing
ZKP	Zero-knowledge proof

CHAPTER 1

INTRODUCTION

In this chapter, we present the idea behind password-authenticated key exchange protocols. After illustrating the need with a few examples, the different mechanisms of our protocol, Password-Authenticated Threshold Group Key Share (PATS) become clear. Then, we explain zero-knowledge proofs with their practical applications. We give brief information about the problem that is employed in our protocols. We present the challenges we tackled and the motivations for designing a password-authenticated threshold group key-sharing protocol leveraging zero-knowledge proofs and show our approach to the problem. We provide a list of contributions that will be explained in the following chapters.

1.1 Password-Authenticated Threshold Group Key Share (PATS)

There is a dealer (a trusted third party, TTP) and n parties in legacy secret-sharing schemes. A secret value v, which the dealer knows, is split into n different pieces called shares again by the dealer. The shares are sent to the corresponding parties over a secure channel. The dealer knows and controls all of these n shares. A secret sharing scheme is called (t, n)-threshold if t or more parties can recover v, but $(t-1)$ or fewer parties cannot efficiently do it. A (t, n)-threshold secret sharing scheme is said to be perfect if no information is disclosed when fewer than t parties try to construct the secret value v.

In this thesis, we present a password-authenticated $(2, 3)$-threshold group key-sharing mechanism that resembles secret sharing schemes, albeit having a different construction involving parties in the generation of the shares. The significant difference is the

lack of prior knowledge of the secret used as a group key in this work. Our approach can be better understood with a simple use case. Assume that Alice, Bob, and Carol write their wishes about the future on a piece of paper, store it in a time capsule secured with a padlock, and promise each other that they will meet after ten years, open the capsule together using the physical key to the padlock and check whether or not their wishes will have come true. They lock the time capsule so that none of them can change the bet alone without others knowing it. They also want to open the capsule if a majority of principals is present after ten years. They solve this problem by *jointly* generating a physical key from some components. Assume that there are three components, say c_1, c_2, and c_3, required to construct the physical key, and each component is cloned. Each party receives two distinct components; for example, Alice gets c_1, c_2; Bob gets c_1, c_3; and Carol gets c_2, c_3. All the unique components (original or cloned) c_1, c_2, and c_3 must be present to construct the physical key, which can be created by combining the components held by any two parties. None of the parties can create the key alone since there is always a missing component. This scenario is illustrated in Figure 1.1.

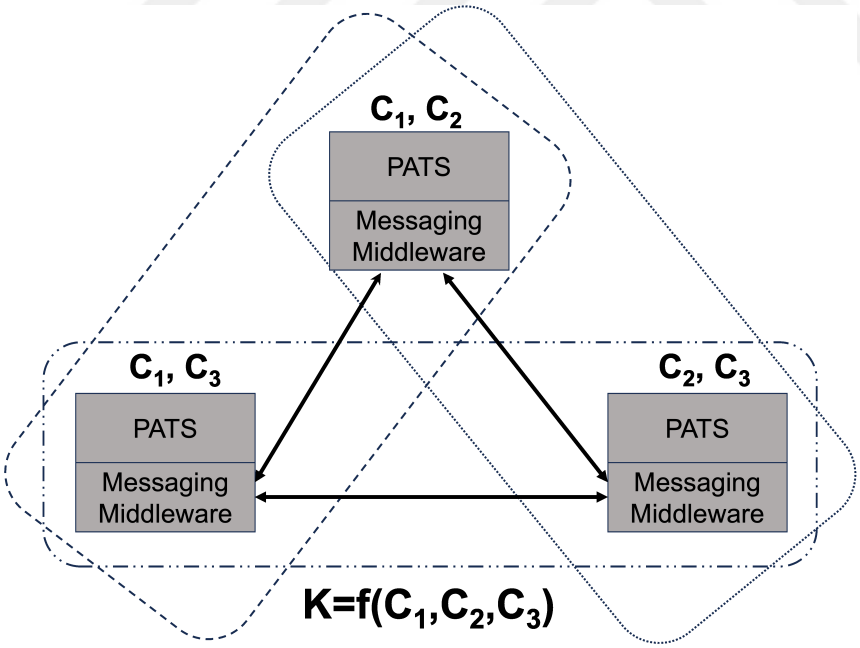


Figure 1.1: PATS scenario for the time capsule.

Several practical scenarios may illustrate the need for the involvement of parties to generate shares of a group key. This approach may be employed in the business

world, the digital services industry, or the financial industry. Let us give examples of centralized and distributed cases. In the distributed group key-sharing scenario, assume Alice, Bob, and Carol are critical infrastructure managers. Out of these three managers, at least two of them have to be present to execute a critical transaction. If Alice goes on a vacation, she must delegate her responsibility to Carol. In such a scenario, instead of Alice providing Bob with her secret, Carol will use her share of the key. This is achieved using a (2,3)-threshold group key-sharing mechanism, which involves input from all parties. None of the parties shall run the transaction alone. Carol and Bob can collaborate and run the transaction when Alice is off duty. The same scenario is applicable when any single party is off duty. In this scenario, parties do not need a centralized or trusted party while initializing the key shares or constructing the shared key using key shares. In the centralized TTP-verified group key-sharing scenario, assume some transaction requires the approval of the TTP, such as a banking transaction. For example, Alice, Bob, and Carol, who have a say in a joint bank account, can only transfer money with the approval of the majority. If at least two out of three parties approve the transaction, the bank acting as the TTP approves and executes the transaction. Although the bank regulates the key shares, the users (not the bank) must determine shares jointly without knowing others' shares for secrecy. Another example is a password manager (keychain) that requires multiple devices to reveal a saved password which is encrypted using PATS. In this case, a user who would like to reveal a saved password from a password manager has to interact with at least two out of the three predefined devices to collect two shares, generate the key, and decrypt the saved password. In all these scenarios, (i) none of the parties shall act alone, and (ii) the lack of a single party shall be tolerated.

1.2 Zero-knowledge Proofs

Zero-knowledge proofs (ZKP) are invaluable cryptographic tools employed in the context of privacy and confidential computing. ZKP allows Alice to convince Bob of the truth of a statement without disclosing any information about the data underlying the statement. ZKPs allow Alice to prove to Bob that she has a certain piece of information or she has conducted a computation without revealing the actual infor-

mation or computation. ZKPs [1] [2] [3] are proofs that are both convincing and yet yield nothing beyond the validity of the information. ZKP has many application areas such as identification, authentication, trustless privacy-preserving transactions (cryptocurrencies or smart contracts), secure data auditing, voting, verifiable confidential computing, auctions, or key management. All in all, ZKP is an indispensable tool in modern cryptography for addressing security, privacy, and trust challenges.

Any NP-complete problem can be employed to design a ZKP [2] and we employ the equal partition problem (EPP) in this thesis. The goal of the partition problem is to determine whether it is possible to divide a set of integers into equal-sum subsets. When this partition is restricted to the case that the disjoint subsets have equal cardinality, then it is called the equal partition problem. EPP has many application areas in various domains such as load balancing, and resource allocation. We formulate EPP on groups, specifically elliptic curves, and refer to it as the equal partition problem on groups (EPPG) and use it to construct two ZKP schemes.

The state of the art of ZKPs is evolving, with a focus on their practical applications and usability. Proposals such as zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) have considerably enhanced their efficiency and scalability, making ZKPs more feasible for use in practice. ZKPs are being actively explored in various application areas. zk-SNARKs are widely accepted for their efficiency and compactness and are used in cryptocurrencies like Zcash. Variants of zk-SNARKs such as Groth16 or Halo are efficient in proof generation. Bulletproofs are efficient and well-suited for confidential transactions. Bulletproofs+ significantly reduces proof sizes and computational complexity for implementing privacy-preserving applications. STARKs provide a high level of security, scalability, and post-quantum security. zk-STARKs (Zero-Knowledge Scalable Transparent ARguments of Knowledge) balance security and scalability.

1.3 Motivations and Challenges

In all these aforementioned scenarios, the challenge is to design a peer-to-peer threshold-based group key generation protocol where there is no central authority and peers

do not learn anything about the generated key other than their shares although they are involved in the key generation. In this challenge, one of the peers should not act as an authority on behalf of the other peers. Based on this challenge, we can state our research question as to whether it is possible to construct a threshold-based group key involving only peers without any central authority or trusted third party and without any trusted setup. We address this research question and propose a password-authenticated (t, n)-threshold group key share (PATS) mechanism.

Yao et al. [4], Liu et al. [5], and Abdalla et al. [6] have proposed protocols that leverage a trusted server responsible for generating shares for each party. In contrast to performing operations on the client (or participant) side, Camenisch et al. [7] introduced an approach aimed at increasing the number of servers required for the verification step. Nevertheless, this type of solution is susceptible to phishing attacks.

Differing from the legacy key-sharing approaches, PATS does not require a centralized authority or trusted setup. Thus, it can be employed in peer-to-peer networks, ad hoc networks, or the Internet of Things (IoT) applications. As a simple example application area assumes that sensors in an IoT application are equipped with a low-entropy password that reflects they belong to a group. However, using this password for securing communications will not be robust since its entropy is low. PATS allows us to secure this application by generating shares of a high-entropy threshold-based group key employing the low-entropy password in a distributed manner without granting higher privileges to any one of the sensors (peers). In scenarios that require a centralized authority or trusted setup, the existing solutions give extra capabilities to this authority to generate the shares of parties. This approach will cause concerns in the case of a breach happening to the centralized authority.

The idea behind the different versions of PATS relies on the password-authenticated key agreement. There are different applications of password-authenticated protocols in the literature, such as password-authenticated group key exchange (PGKE) protocols, (t, n)-threshold secret sharing, or threshold password-authenticated key exchange protocols. A group key exchange protocol that is based on password authentication is called Group PAKE. There is "a fairy-ring dance" [8] construction to transform any PAKE protocol into a Group PAKE protocol. Group PAKE protocols

typically involve either a shared password managed by a common trusted server or the use of pairwise passwords between group members. Similarly, PATS assumes a pre-configured common password known to all principals. To the best of our knowledge, existing Group PAKE protocols do not implement threshold-based key construction. Therefore, as a notable difference from legacy Group PAKE protocols, PATS implements threshold-based group key construction without the need for any trusted third party. On the other hand, the idea behind (t, n)-threshold secret sharing protocols is that each party possesses a portion of the secret, called a share. The shares of at least t parties are sufficient to reconstruct the secret. In these schemes, a secret that is known in advance is used to create shares whereas we employ the reverse operation in PATS for different environments, either centralized or distributed. Moreover, there is always a trusted third party (dealer) in secret sharing protocols. In threshold-password-authenticated key exchange protocols, the concept of thresholding is different from that of secret sharing. In such schemes, a client is authenticated by multiple servers, and acquiring authentication from a subset of authentication servers is the idea behind thresholding. These are completely different constructions compared to PATS. Thresholding in PATS resembles the idea of (t, n)-threshold secret sharing.

1.4 Our Contributions

The two main contributions of this thesis are:

1. We propose a zero-knowledge (2,3)-threshold password-authenticated key exchange scheme that we call PATS which involves parties in group key sharing. This protocol takes advantage of zero-knowledge proofs compared to well-known threshold key exchange schemes and will tolerate the existence of semi-trusted parties. Although PATS resembles threshold secret-sharing schemes, it has a different structure and approach, as demonstrated by the aforementioned use cases. There is no sharing of prior confidential information in PATS, and a trusted setup is not required. In legacy secret sharing, there is a secret known to the dealer at the beginning of the protocol, and the shares of the secret are provided to the parties by the dealer. However, the secret (key) is not known in advance in PATS. PATS allows parties to construct a secret (key) using their

shares obtained at the end of the protocol. In other words, legacy secret sharing starts from the secret and constructs the shares. PATS follows the opposite approach, where parties communicate to establish their shares securely over insecure channels. The shares will help them (or a trusted third party) construct the group key. When there is a data breach in a dealer of a legacy secret sharing scheme, the share of each party may be exposed. We present centralized and distributed PATS variants and then generalize PATS to (t, n)-threshold scheme. PATS supports distributed operations and facilitates the verification of the key by a trusted third party that may also partake in group key sharing.

2. We propose two zero-knowledge proofs in Section 4. The first protocol, a Three-pass ZKP based on the Equal Partition Problem (EPPZKP), is a Stern-type ZKP. The second protocol, Two Challenge ZKP based on the Equal Partition Problem (2EPPZKP), is inspired by the 5-pass identification scheme presented by Sakumoto et al. [9] and reduces the soundness error. We present a detailed security and performance analysis of EPPZKP.

1.5 Outline

The remainder of this thesis is structured as follows:

- Chapter 2 reviews the literature concerning protocols for password-authenticated group key exchange and those employing zero-knowledge proofs. Password-authenticated key exchange protocols exhibit diverse utility, including applications in group key exchange (PGKE), threshold secret sharing, and group key agreement (GKA). Following the exposition of password-authenticated group key exchange protocols, we subsequently introduce zero-knowledge protocols, providing a comparative analysis of various cutting-edge protocols.

- Chapter 3 presents (2,3)-threshold password-authenticated key exchange (PATS) schemes with zero-knowledge properties, where shares are collaboratively generated through public channels. PATS adopts the overarching strategy of the password-authenticated key agreement protocol (PAKE), which

we will introduce. Following the explanation of the (2,3)-threshold PATS, we extend this framework to encompass n parties, introducing the (t, n)-threshold PATS. Two variations of this scheme are presented: one implemented in finite fields and the other utilizing elliptic curves. Subsequently, we outline the centralized approach, necessitating the presence of a TTP.

- Chapter 4 introduces the Equal Partition Problem on Groups (EPPG). Subsequently, we outline a three-pass Zero-Knowledge Proof (ZKP) based on EPPG. The algebraic setup phase clarifies the theoretical aspects comprehensively. Following this, we apply the protocol to elliptic curves, discussing its practical implementation. To minimize soundness errors, we then introduce a second protocol—a two-challenge ZKP based on EPPG. A thorough examination, along with a step-by-step explanation, is provided for each protocol.

- Chapter 5 presents an adversary model that describes the capabilities of potential attackers, outlining the typical security requirements for Password-Authenticated Key Exchange (PAKE) protocols, and demonstrating how PATS aligns with these security requirements. The subsequent section of the chapter is dedicated to inspecting the proposed zero-knowledge protocols. A zero-knowledge proof (ZKP) protocol is expected to fulfill three key properties: completeness, soundness, and zero-knowledge. While establishing completeness is relatively straightforward in our context, demonstrating soundness necessitates the definition of extractors. To establish zero-knowledge(ness), we leverage the utilization of simulators

- Chapter 6 presents the concluding remarks and the future work of the thesis.

CHAPTER 2

RELATED WORK

In this chapter, we present the related work of password-authenticated group key exchange protocols and zero-knowledge proofs. Password-authenticated key exchange protocols have different applications, such as group key exchange (PGKE), threshold secret sharing, and group key agreement (GKA). The main idea behind these approaches is to include each party in the key exchange. Threshold secret sharing enables situations in which some missing parties can be tolerated. After presenting password-authenticated group key exchange protocols, we present zero-knowledge protocols and compare several state-of-the-art protocols.

2.1 Password-Authenticated Key Exchange

In this section, we first review the password-authenticated group key exchange protocols and then present threshold secret-sharing approaches by comparing the related work to PATS [10]. In the second part of the chapter, we present different applications of zero-knowledge protocols.

2.1.1 Password-Authenticated Group Key Exchange (PGKE) Protocols

A group key can be generated by employing PAKE protocols [4, 5, 8, 11, 12, 13, 14]. Two parties can use an Authenticated Key Exchange (AKE) protocol to communicate securely. If the number of parties in a communication increases, a Group AKE protocol can be established. Instead of any Public Key Infrastructure (PKI), a Group AKE protocol can be based on password authentication, called Group PAKE (or PGKE).

PGKE protocols enable parties to use low-entropy passwords to resist attacks coming from insiders. In the literature, PAKE protocols have essential contributions. However, PGKE has not received the same attention.

Asokan and Ginzboorg [15] proposed a PGKE protocol for ad hoc networks but did not provide formal security proof of the protocol. Bresson et al. [16] offered a provable security framework using Diffie-Hellman key exchange to assess the security of PGKE protocols. Yao et al. [4] proposed a PGKE protocol in which each party shares a different password with a trusted server. Using a similar approach to Bresson et al. [16], they provided formal proof of the protocol's security. Similarly, Liu et al. [5] proposed a password-based authenticated key exchange protocol for groups, called nPAKE, in which each party shares a password with a trusted server and a temporary encryption key with the adjacent parties with the help of the server. The protocol's security is based on the chosen based computational Diffie-Hellman (CCDH) assumption problem. It is also demonstrated that the protocol is secure against dictionary attacks and man-in-the-middle attacks. In 2015, Hao et al. [8] presented a "fairy-ring dance" construction that transforms any PAKE protocol into a PGKE protocol. The paper provides two protocols, SPEKE+ and J-PAKE+, requiring 2 and 3 rounds respectively. They showed that both protocols are secure against offline dictionary attacks while providing forward secrecy and known-session security. In 2018, Wei et al. [14] proposed a compiler that can convert any Group KE protocol into a secure PGKE protocol with two extra rounds of communication overhead. The paper included the security proof of the compiler in the standard model.

2.1.2 Threshold Secret Sharing

Threshold secret sharing was invented separately by Blakley [17] and Shamir [18]. In (t,n)-threshold secret sharing, there are n parties that each hold a part of the secret, called shares, and t of the parties' part will reconstruct the secret. It is called threshold since less than t parties will not gain any information about the secret.

There are several well-known secret-sharing schemes applying different approaches, such as Shamir's Secret Sharing [18], Blakley's Secret Sharing [17], and Mignotte's and Asmuth-Bloom's Schemes [19]. Shamir's secret sharing is based on Lagrange in-

terpolation over finite fields, using the fact that to uniquely determine a polynomial of degree less than or equal to $t-1$, t points are enough. This scheme is perfect since no information is leaked by the shares. Additionally, it is ideal since the length of each share is equal to the resulting secret. Blakley's secret sharing is based on hyperplane geometry. The dealer distributes a t-dimensional hyperplane equation over a field to each party. The intersection of these hyperplanes reveals the secret. Asmuth-Bloom secret sharing is based on the Chinese Remainder Theorem (CRT). Krawczyk's secret sharing is based on the existence of a one-way function. Having equal-length shares in a secret-sharing scheme provides information-theoretic security. Instead of information-theoretic security, this work depends on computational security against resource-bounded adversaries.

A secret sharing scheme can be verifiable or publicly verifiable. In a verifiable secret sharing scheme (VSS), parties and the dealer verify the consistency of their shares in a two-phase protocol. The notion of verifiability was first introduced by Chor et al. [20]. In the sharing phase, the dealer distributes the secret to n parties such that fewer than t parties cannot deduce any information, similar to traditional threshold secret sharing. In the reconstruction phase, each party provides their information to reconstruct the secret.

There are several well-known verifiable secret-sharing schemes, such as Chor-Goldwasser-Micali-Awerbuch's scheme [20], Feldman's scheme [21], and Benaloh's scheme [22]. Chor et al. [20] introduce two key cryptographic primitives: verifiable secret sharing (VSS) and a method to achieve simultaneity in broadcast networks despite faults. Feldman [21] presents an efficient, non-interactive protocol for verifiable secret sharing (VSS), which allows a secret to be securely shared among a group of processors such that a quorum is required to reconstruct the secret. Benaloh [22] discusses secret sharing homomorphisms, a property of secret sharing schemes that allows multiple secrets to be combined through direct computation on their shares without revealing the underlying secrets themselves.

Threshold secret sharing is also used in password-based single-sign-on authentication schemes. Zhang et al. [23] proposed a secure and efficient single-sign-on authentication scheme for mobile users. Their protocol provides stronger security compared

to existing schemes under the defined adversary model. Jiang et al. [24] proposed a password-based single-sign-on authentication protocol based on lattices. Utilizing lattices, their proposed protocol is quantum-resistant and resistant to offline password-guessing attacks.

There are threshold secret-sharing protocols that take advantage of authentication servers. In this case, when t out of the n authentication servers communicate, the authentication is successful. Abdalla et al. [6] proposed a Threshold Password-based Authenticated Key Exchange (GTPAKE) protocol in which authentication servers act as two different parties: a gateway, which handles communication with the clients, and a back-end server, which verifies the client's identity. Camenisch et al. [7] proposed a protocol where the verification process is distributed across multiple servers, ensuring the password remains secure as long as the number of breached servers is less than the threshold. This type of threshold password-authenticated secret sharing, called TPASS, was previously proposed by Bagherzandi et al. [25] and Camenisch et al. [26]. However, these proposals are vulnerable to phishing attacks in which users interact with malicious servers.

2.1.3 Group Key Agreement (GKA) Protocols

In group communications, group key agreement (GKA) protocols are widely used to provide secure communication. There are two types of group key agreement protocols, namely centralized group key agreement (CGKA) and distributed group key agreement (DGKA). In their paper, Zhang et al. [27] proposed a threshold authenticated group key protocol based on Shamir's secret sharing for Unmanned Aerial Vehicle (UAV) Ad Hoc Network (UANET). Since the environment for UAVs requires dynamic interaction and wireless communication would cause temporary disconnection from a group, having a stable group key agreement is important to address these problems. Their solution considers the disconnection of group members, as well as a group key recovery protocol in case a disconnected tries to reconnect and obtain the latest group key.

2.2 Zero-Knowledge Protocols

zk-SNARK (Zero-Knowledge Succinct Non-Interactive ARgument of Knowledge) [28] is a non-interactive ZKP protocol initially proposed by Bitansky et al. in 2014. They showed that if there are extractable collision-resistant hash functions (ECRHs) and an appropriate private information retrieval scheme, then there exist SNARKs for NP. Additionally, in this work, they propose candidates for ECRH constructions. One of these is based on the hardness of the discrete logarithm problem, while the other two are based on hard problems on lattices, namely, knapsack problems. In 2016, Groth [29] constructed an efficient zk-SNARK for Quadratic Arithmetic Programs using bilinear groups. Zcash uses Groth's construction [30]. A downside of zk-SNARKs is that they use non-public randomness in their setup phase, meaning they require a trusted setup. Additionally, zk-SNARKs are not quantum-safe [31]. A remedy to these problems is zk-STARK.

zk-STARK (Scalable Transparent Zero-Knowledge ARgument of Knowledge) [32] was introduced by Ben-Sasson et al. in 2018. It is an Interactive Oracle Proofs (IOP) system. zk-STARK is more transparent, as it requires no trusted setup. zk-STARKs rely on collision-resistant hash functions. The zk-STARK-friendly hash function [33, 34] is the focus of extensive research. By relying on hash functions, zk-STARKs are quantum resistant. A major disadvantage of zk-STARKs is their proof size compared to zk-SNARKs. Recent works have attempted to reduce the proof length.

As presented in Table 2.1, zk-SNARK's algorithmic complexity for the prover is $\mathcal{O}(Clog(C))$ and for the verifier is $\mathcal{O}(1)$, which are lower compared to zk-STARK's complexity of $\mathcal{O}(Cpolylog(C))$ and $\mathcal{O}(polylog(C))$, respectively. The proof size of zk-SNARKs is $\mathcal{O}(1)$, whereas it is $\mathcal{O}(polylog(C))$ for zk-STARKs.

Aurora [35] is a zk-SNARK proposed by Ben-Sasson et al. in 2019. They developed the protocol for Rank-1 Constraint Satisfaction (R1CS), which is an NP-complete language. Aurora employs a public (transparent) setup phase. It is lightweight and quantum-safe. For the same number of constraints defined in R1CS, they reduced the proof size to 20 times shorter than previous zk-SNARK proposals. Aurora uses an

IOP for solving the univariate version of the sum check problem [36].

Hyrax [37] is another zk-SNARK variant proposed by Wahby et al. in 2017. They convert an interactive proof of arithmetic circuit (AC) satisfiability into a ZKP scheme. Hyrax's proofs are sublinear in circuit size (succinct), do not require a trusted setup phase, and are secure under the discrete log assumption.

Ligero [38] is a zero-knowledge argument based on a chosen collision-resistant hash function. By making it non-interactive in the random oracle model, efficient zk-SNARKs can be obtained that do not require a trusted setup or public-key encryption.

Bulletproofs [39] is a short zero-knowledge proof depending on the hardness of the discrete logarithm problem and has no trusted setup. It uses Pedersen vector commitment and achieves a very short proof size using a groundbreaking inner product algorithm. It can be made non-interactive using the Fiat-Shamir heuristic. One disadvantage of Bulletproofs is that it takes more time to verify a Bulletproof than to verify a SNARK proof.

Libra [40] is a zero-knowledge proof scheme that offers both optimal prover time with a succinct proof size and $\mathcal{O}(d \log C)$ verification time. Unlike other proposals, Libra employs a one-time setup phase that does not need to be repeated per statement. It relies on the GKR protocol [41].

2.2.1 Performance Metrics

The efficiency of ZKP implementations can be compared based on the following performance metrics [42]. Here, we list the most commonly used metrics and present in Table 2.1 a comparison of the state-of-the-art ZKP schemes based on these metrics.

1. Proof size (succinctness): the size of the proof in comparison to the circuit size (C) representing the statement.
   (a) Fully succinct: $\mathcal{O}(1)$
   (b) Polylog succinct: $\mathcal{O}(\log^2 C)$
   (c) Square root succinct: $\mathcal{O}(\sqrt{C})$

(d) Depth-succinct: $\mathcal{O}(d \log C)$ assuming that the depth of the verification circuit is d.

(e) Non-succinct: the proof is not sublinear in C.

2. The time complexity for the trusted setup (if it exists)

3. The time complexity of the tasks executed by the prover P: the efficiency of the proof generation

4. The time complexity of the tasks executed by the verifier V: the efficiency of the proof verification

In addition to these metrics, round complexity, parallelizability, batching, memory consumption, number of operations in the algorithms, memory consumption, disk, and storage requirements can be considered additional performance metrics for comparing various ZKP proposals [42].

Table 2.1: Comparison of the state-of-the-art ZKP schemes based on various performance metrics where C is the number of gates in the circuit with depth d, and n is the size of its input.

Type	**ZKP**	**Ref.**	**Library**	**Lang.**	**Year**	**Trusted Setup**	**Trans-parent**	**Quantum-safe**	**Proof size**	**Prover complexity**	**Verifier complexity**
	Bulletproof	[39]	[e]	Rust	2017	None	✓	✗	$\mathcal{O}(\log C)$	$\mathcal{O}(C)$	$\mathcal{O}(C)$
	Libra	[40]	[a]	C++	2019	One-time	✓	✓	$\mathcal{O}(d \log C)$	$\mathcal{O}(C)$	$\mathcal{O}(d \log C)$
SNARK	General	[43]	[g]	C++	2016	Per statement	✗	✗	$\mathcal{O}(1)$	$\mathcal{O}(C \log C)$	$\mathcal{O}(1)$
	Hyrax	[37]	[d]	Python	2017	None	✓	✗	$\mathcal{O}(\sqrt{n} + d \log C)$	$\mathcal{O}(C \log C)$	$\mathcal{O}(\sqrt{n} + d \log C)$
	Ligero	[38]	[f]	C++	2017	None	✓	✓	$\mathcal{O}(\sqrt{C})$	$\mathcal{O}(C \log C)$	$\mathcal{O}(C)$
	Aurora	[35]	[b]	C++	2018	None	✓	✓	$\mathcal{O}(\log^2 C)$	$\mathcal{O}(C \log C)$	$\mathcal{O}(C)$
STARK	General	[44]	[c]	C++	2018	None	✓	✓	$\mathcal{O}(\log^2 C)$	$\mathcal{O}(C \log^2 C)$	$\mathcal{O}(\log^2 C)$

[a] https://github.com/sunblaze-ucb/Libra

[b] https://github.com/scipr-lab/libiop

[c] https://github.com/elibensasson/libSTARK

[d] https://github.com/hyraxZK

[e] https://github.com/dalek-cryptography/bulletproofs

[f] https://github.com/scipr-lab/libiop

[g] https://github.com/scipr-lab/libsnark

2.2.2 Code-Based Cryptographic Identification Schemes

Stern [45] introduced an innovative identification scheme using the syndrome decoding (SD) problem for error-correcting codes, highlighting practicality and security through zero-knowledge proofs. This scheme simplifies traditional protocols by leveraging a fixed $(n\text{-}k)$-matrix H over a binary field, providing efficient error correction and involving an interactive zero-knowledge protocol for secure identification.

Building on Stern's work, Véron [46] enhanced the efficiency and practicality of the original scheme. By using a generator matrix of a random linear binary code, Véron's scheme improves transmission rate and reduces computational complexity, making it suitable for low-power devices like smart cards.

Gaborit and Girault [47] revisited Stern's protocol, drastically reducing the size of public keys while maintaining high security by using double-circulant matrices. This approach made the scheme practical for lightweight cryptographic applications by reducing public key sizes to as low as 347 bits.

Cayrel et al. [48] further improved Stern's scheme by employing q-ary codes, which reduce communication complexity and public key size. They introduced a 5-pass identification protocol, cutting down the public key size from Stern's 15 Kbytes to 4 Kbytes and ensuring zero-knowledge properties against quantum attacks.

Aguilar et al. [49] introduced an enhanced version of Stern's protocol, reducing communication costs by 40% while maintaining strong security. This scheme achieves a lower cheating probability by using cyclic shifts of the public key syndrome and optimizing hashed commitments, resulting in smaller key sizes.

Dau, Tran, and Kim [50] proposed a code-based signature scheme leveraging Lyubashevsky's lattice-based framework. The scheme ensures the Hamming weight of each row of the private key matrix is below the Gilbert-Varshamov bound, enhancing security and maintaining robustness against attacks.

Marco Baldi et al. [51] introduce the Restricted Syndrome Decoding Problem (R-SDP), limiting vector entries to a subset of the finite field. They prove R-SDP's NP-completeness and describe solving algorithms, adapting Zero-Knowledge Iden-

tification (ZK-ID) schemes to rely on R-SDP, resulting in compact public keys and reduced communication costs.

Kales and Zaverucha [52] present a generic forgery attack on signature schemes derived from five-round identification schemes using the Fiat-Shamir transform. Their attack exploits parallel repetition, demonstrating vulnerabilities in MQDSS post-quantum signature scheme, breaking it with approximately 2^{95} operations.

Bettaieb et al. [53] address flaws in the zero-knowledge proofs of Véron and AGS identification protocols based on the General Syndrome Decoding (GSD) problem. They propose a masking technique that rectifies these issues without performance penalties, leading to the development of the Masked Véron and Masked AGS protocols with several performance improvements.

Baldi et al. [54] propose a novel identification and signature scheme leveraging the Restricted Syndrome Decoding Problem (R-SDP). The scheme modifies existing zero-knowledge protocols to use R-SDP, resulting in reduced communication costs and smaller public key sizes, making the protocols more efficient and secure against attacks.

Blocki and Blum [55] proposed a Stern-type zero-knowledge scheme utilizing the equal partition problem, and subsequently extended their protocol to address the subset sum problem. Building on their work, we explored the equal partition problem through the lens of group theory and introduced a novel zero-knowledge protocol grounded in this perspective.

CHAPTER 3

PATS: PASSWORD-AUTHENTICATED (2,3)-THRESHOLD KEY EXCHANGE

In this chapter, we present the details of the zero-knowledge (2,3)-threshold password-authenticated key exchange (PATS) schemes, where shares are constructed mutually over public channels. PATS employs the general approach of password-authenticated key agreement protocols (PAKE), which we will briefly introduce in the sequel. After discussing the (2,3)-threshold PATS, we will generalize this approach to n parties and present the (t, n)-threshold PATS. We will also define the distributed approach called distributed (2,3)-PATS, which operates without a trusted third party (TTP) and where each party generates shares of a group key. We present two variants of this scheme: the first is implemented in finite fields, and the second uses elliptic curves. Finally, we introduce the centralized approach that requires a TTP.

3.1 PATS Inspired by PAKE

A password-authenticated key exchange (PAKE) protocol provides two or more parties a secure authenticated communication over an insecure channel and establishes a high-entropy cryptographic key using a shared low-entropy secret such as a humanly-memorable password. While many password-based protocols risk offline attacks [56], the authenticated key exchange protocols prevent such vulnerabilities. Among many other PAKE protocols, treating PAKE as a secure two-party computation problem, J-PAKE stands out from the rest.

Password Authenticated Key Exchange by Juggling (J-PAKE) depends on zero-

knowledge proofs [57] [58] [59]. J-PAKE, standardized in ISO/IEC 11770-4 and RFC 8236 [60, 61], is a balanced PAKE protocol, and it does not require any PKI deployments, or Hash-to-Group function or a trusted setup. Such independence makes J-PAKE's implementation simple. The protocol is used over finite fields and elliptic curves. We give a brief overview of J-PAKE over a finite field.

J-PAKE relies on the hardness assumption of the Decision Diffie-Hellman (DDH) problem. Two parties, Alice and Bob, share a password s and agree on an element g of $\mathbb{Z}_p^*$ of prime order q. Both parties randomly choose their private key pairs, with the first component in the range $[0, q-1]$ and the second component in the range $[1, q-1]$. Suppose Alice chooses (x_1, x_2) and Bob chooses (x_3, x_4). They compute and interchange all g^{x_i} values over an unsecured network, along with zero-knowledge proofs of the exponents. In the second round, Alice computes $A = g^{(x_1+x_3+x_4)x_2 s}$ and Bob computes $B = g^{(x_1+x_2+x_3)x_4 s}$. They send these values to each other with a ZKP of their secret exponents $x_2 s$ and $x_4 s$. Now both can compute the keying material $K = (A/g^{x_2 x_4 s})^{x_4} = g^{(x_1+x_3)x_2 x_4 s} = (B/g^{x_2 x_4 s})^{x_2}$. Finally, the session key is derived by a hash (or key generation) function $k = H(K)$.

PATS is inspired by PAKE and relies on the hardness assumption of the DDH problem. It uses the juggling method of J-PAKE to construct shares and provides mutual authentication among three parties via a shared password.

3.2 Design Challenges

To develop PATS, security, and efficiency are the two important motivations that we focused on:

- **Security:** Using a PAKE protocol in the protocol's core allows us a way to establish a high-entropy cryptographic key from a low-entropy secret. While password-based protocols are prone to offline attacks, the authenticated key exchange protocols overcome this vulnerability. The protocol also takes advantage of zero-knowledge proofs compared to well-known threshold key exchange schemes.

- **Efficiency:** One variant of PATS relies on elliptic curves. Computation over elliptic curves uses smaller coefficients while providing similar security strength compared to finite field calculations.

According to Zipf's law [62] [63], the space of user-chosen passwords is constrained compared to the space of all possible passwords. It is stated that the advantage for guessing will be over 0.5 in some cases. In PATS, the final secret key has different components. For the TTP-verified version, the b_i values are provided by the TTP, and a_i values are generated by the parties. For the distributed version, the a_i and b_i values are generated by the parties. However, for any party P_i, accessing the values a_j and b_j for $j \neq i$ is equal to having a solution to the DDH problem. The secret key K also requires the value of s that is shared between parties. Therefore, Zipf's law does not have an impact on the value of K.

3.3 Distributed (2,3)-PATS

In some scenarios, such as the one mentioned earlier involving Alice and her friends' time capsule, the parties do not require any centralized entity (e.g., a TTP). Instead, they may run a distributed protocol for sharing the group key. Both ephemeral key integers, a_i and b_i, are selected by the party P_i, and there is no TTP verification. Whenever a group key is necessary, any two of the three parties can use their shares to construct the group key. This section presents two variants that use either finite fields or elliptic curves.

3.3.1 Distributed (2,3)-PATS in Finite Fields

Assume there are three parties, P_1, P_2, and P_3, who share a common low-entropy password s in the range $[1, q-1]$. The parties will establish a common (2,3)-threshold group key. Any two out of three parties can construct the group key collaboratively. Let p and q be large primes such that q is a divisor of $p-1$. Let g be an element of $\mathbb{Z}_p^*$ of order q. The communicating parties, P_1, P_2, and P_3 agree on the (p, q, g) value string, which can be hard-wired in software. Distributed (2,3)-PATS involves three

rounds, as shown in Figure 3.1.

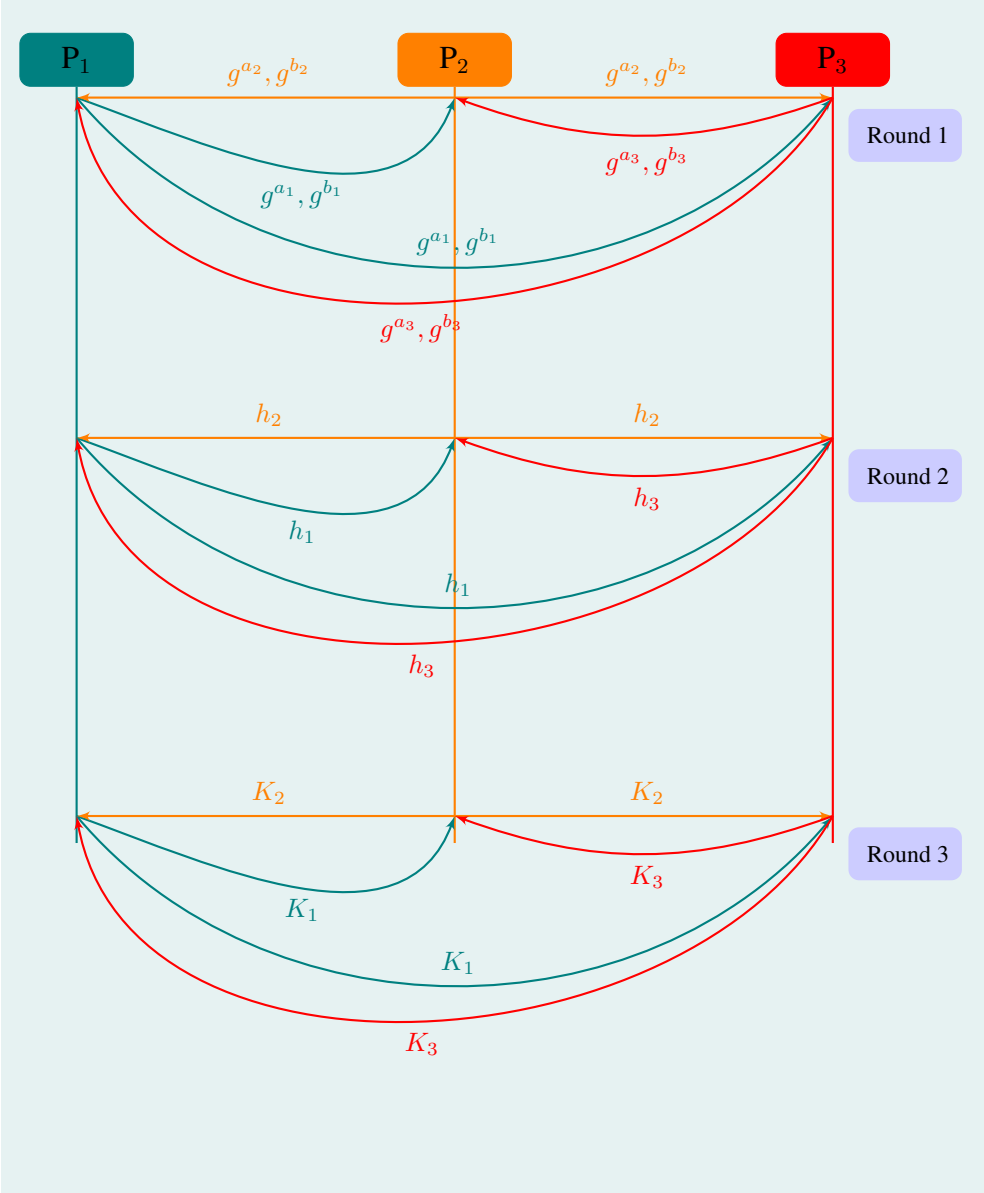


Figure 3.1: Distributed (2,3)-PATS in finite fields.

- **Round 1**: Each party P_i for $i = 1, 2, 3$ selects ephemeral private key integers a_i uniformly at random from $[0, q-1]$ and b_i uniformly at random from $[1, q-1]$. Each party then broadcasts $g^{a_i} \mod p$ and $g^{b_i} \mod p$ together with the zero-knowledge proofs of the exponents a_i and b_i. For example, Schnorr's non-interactive zero-knowledge (NIZK) proof, as used in J-PAKE, can be employed here [64].

When the round finishes, each party checks that the received ZKPs are valid. Additionally, each party P_i checks that $g^{b_j} \not\equiv 1 \mod p$ for $j \neq i$. If any of these checks fail, this session should be canceled.

- **Round 2**: Each party P_i computes and broadcasts $h_i = (g^{\sum_{j=1}^{3} a_j + \sum_{j \neq i} b_j})^{b_i s}$ together with the zero knowledge proofs of the exponent $b_i s$. Again at the end of the round, each party checks that the received ZKP is valid. Also, party P_i should ensure that the elements h_j and h_t are non-identity.

- **Round 3**: Each party P_i computes its secret shadow K_i where $K_i = (K_{ij}, K_{il}) = ((h_j(g^{b_j b_i s})^{-1})^{b_i}, (h_l(g^{b_l b_i s})^{-1})^{b_i})$. Notice that the indices commute $K_{ij} = K_{ji}$.

Whenever the necessity of a group key emerges, any two among three parties use their shares and construct the group key $K = K_{ij}K_{it}K_{jt} = K_{ji}K_{jt}K_{ti} = K_{ij}K_{it}K_{tj}$. Figure 3.1 shows the flow of the protocol. Note that, instead of directly using K, it can be used as input to a key generation function.

We have shown that the constructed group key $K = K_{ij}K_{it}K_{jt} = K_{ji}K_{jt}K_{ti} = K_{ij}K_{it}K_{tj}$. To show the correctness of the equalities, we need to prove that $K_{ij} = K_{ji}$ for $i \neq j$.

Claim 3.3.1 *For $i \neq j$, $K_{ij} = K_{ji}$.*

Proof 3.3.1

$$K_{ij} = (h_j(g^{b_j b_i s})^{-1})^{b_i} \tag{3.1}$$

$$= ((g^{\sum_{k=1}^{3} a_k + \sum_{k \neq j} b_k})^{b_j s}(g^{b_j b_i s})^{-1})^{b_i} \tag{3.2}$$

$$= (g^{\sum_{k=1}^{3} a_k b_j s + \sum_{\substack{k \neq i \\ k \neq j}} b_k b_j s})^{b_i} \tag{3.3}$$

$$= (g^{\sum_{k=1}^{3} a_k b_i s + \sum_{\substack{k \neq i \\ k \neq j}} b_k b_i s})^{b_j} \tag{3.4}$$

$$= ((g^{\sum_{k=1}^{3} a_k + \sum_{k \neq i} b_k})^{b_i s}(g^{b_i b_j s})^{-1})^{b_j} \tag{3.5}$$

$$= (h_i(g^{b_i b_j s})^{-1})^{b_j} \tag{3.6}$$

$$= K_{ji} \tag{3.7}$$

■

In practice, the final step will be combining the shares from at least two parties and generating the group key without revealing each share independently to other parties. Let us consider the time capsule example again. The time capsule is a trusted combiner that does not copy the shared values given as input. Each party uses its shares (in this case, the physical key components that are used to construct the key of the padlock) to open the padlock on the time capsule. To open the time capsule, at least two parties must come together. Neither the combiner nor the remaining parties have any idea about the value of each share of other parties.

3.3.2 Distributed (2,3)-PATS using Elliptic Curves

Initially, all parties agree on the domain parameters. Namely, they agree on the sequence (p, a, b, G, q, h) where p and q are large primes, the elliptic curve E is the curve satisfying the equation $y^2 = x^3 + ax + b$ defined over the prime field F_p, the base point G has order q, and h is the co-factor, a small integer. One may wish to use recommended elliptic curves by NIST for key establishment schemes in SP 800-56A [65]. As in the finite field case, we assume there are three parties P_1, P_2, and P_3 share a common low-entropy password s in the range $[1, q-1]$, and they execute the following three rounds:

- **Round 1**: Each party P_i selects an ephemeral private key integers a_i uniformly random from $[0, q-1]$, b_i uniformly random from $[1, q-1]$ and then broadcasts a_iG and b_iG, together with the zero-knowledge proofs of the multiples a_i, b_i. At the end of the round, each party P_i should check that the received ZKPs are valid and b_jG a non-identity element for $j \neq i$. If any of these checks fail, this session should be canceled.

- **Round 2**: Each party P_i computes and broadcasts $H_i = sb_i(\sum_{j=1}^{3} a_j + \sum_{j \neq i} b_j)G$ together with the zero-knowledge proofs of the multiple sb_i. At the end of the round, each party checks that the received ZKPs are valid and that the elements H_j and H_k are non-identity. If any of these checks fail, the session will be canceled.

- **Round 3**: Each party P_i computes their secret shares $K_i = (K_{ij}, K_{it})$ where

$K_i = (K_{ij}, K_{it}) = (b_i(H_j - b_j b_i sG), b_i(H_t - b_t b_i sG))$. Notice that the indices commute $K_{ij} = K_{ji}$.

3.4 TTP-verified (2,3)-PATS

In some scenarios, such as the banking use-case mentioned above, the secret will eventually be employed by a centralized entity (e.g., a TTP), for example, for encrypting a database or authenticating a transaction. In such scenarios, the TTP has to use the group key for conducting the operation where the TTP-verified (2,3)-PATS can be employed. The protocol runs as in the previous case. However, the only difference is that ephemeral key integer b_is are selected by the TTP and sent to the parties over a secure channel. Figure 3.2 shows the flow of the protocol.

Note that in the TTP-verified (2,3)-PATS, not only is the TTP involved in the process and has a say in the constructed shares but also all the parties involved in the process as well. The TTP verification has two steps. First, it calculates the shares: TTP has the values s, b_1, b_2, and b_3. Therefore, by gathering h_1, h_2, and h_3 values from the public channel, it can recover all shares by imitating the operations done by the parties. Secondly, it checks whether these values match the shares sent by the parties. Here the equality comes from the fact that the indices of the shared shares commute.

Notice that any two parties among three can recover K. Using K as input, the parties may apply a Key Derivation Function (KDF) to derive a common session key k.

In this section, we have presented two variants of the PATS protocol, employing finite fields, and elliptic curves, respectively. In the next section, we will generalize the scheme to a (t, n)-threshold scheme.

3.5 (t, n)-PATS

In this section, we generalize the (2,3)-PATS to (t,n)-threshold scheme. Considering the efficiency requirement, in cases where t is less than $n - 1$, the path followed in PATS will be cumbersome. We picture the scheme for distributed variants over finite

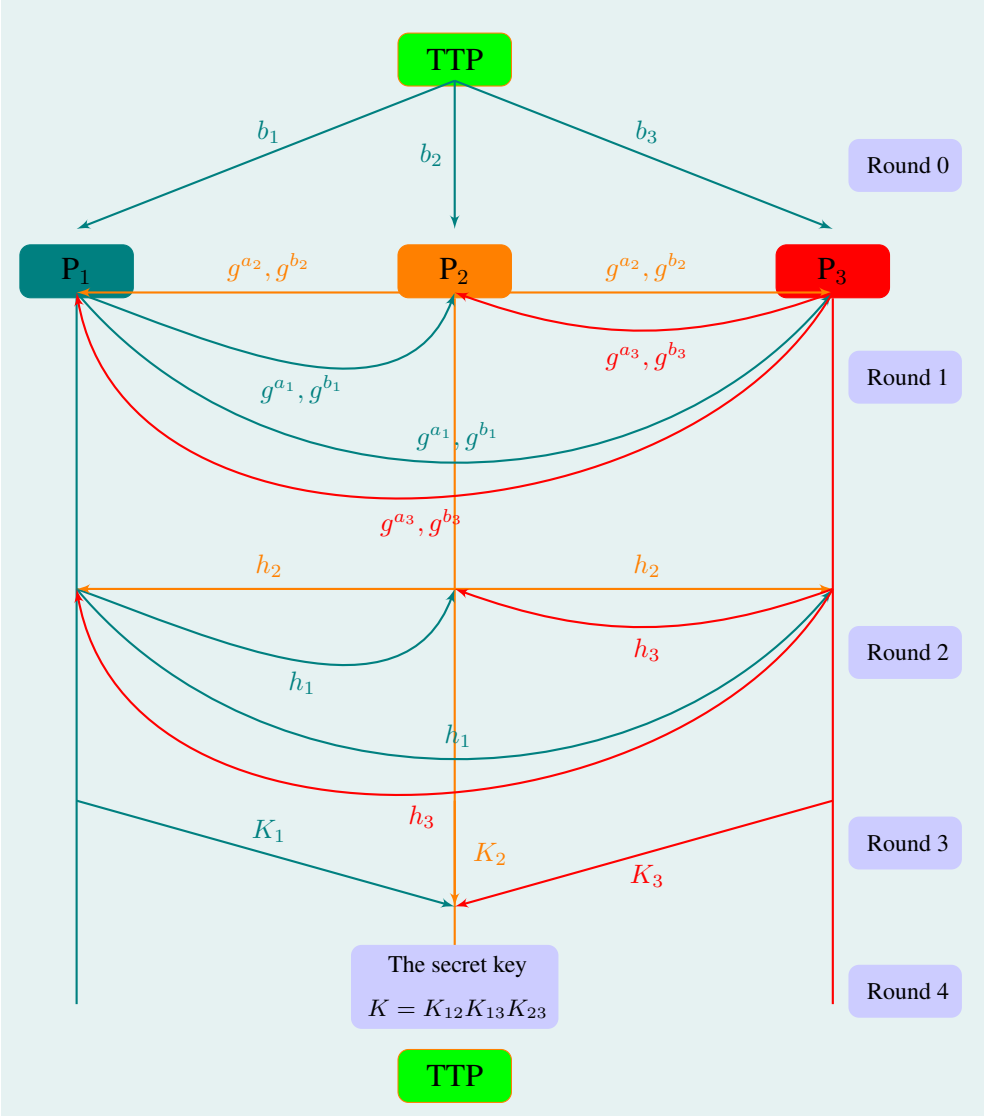


Figure 3.2: TTP-verified (2,3)-PATS in finite fields.

fields. The generalized version can be phrased straightforwardly on elliptic curves, mutatis mutandis as above, for centralized (t,n)-PATS over finite fields and elliptic curves. Assume there are n parties who share a common low-entropy password s in the range $[1, q-1]$, and will establish a common (t, n)-threshold group key. That is any t among n will have the group key. Here s can be a low-entropy group key or a session nonce. A trusted third party (dealer) and the communicating parties, $P_1, \ldots, P_n$ agree on (p, q, g) value string, which can be hard-wired in the software.

For $t = n - 1$, (t,n)-PATS procedure is:

- **Round 1**: Each party P_i selects ephemeral private key integers a_i and b_i uniformly at random from $[0, q-1]$ and $[1, q-1]$ respectively. Then broadcasts $g^{a_i} \mod p$ and $g^{b_i} \mod p$ together with the zero-knowledge proofs of the exponents a_i, b_i.

  When the round finishes, each party checks that the received ZKPs are valid. Also, each party P_i checks that $g^{b_j} \neq 1 \mod p$ for $j \neq i$. If any of these checks fail, this session should be canceled.

- **Round 2**: Each party P_i computes and broadcasts $h_i = (g^{\sum_{j=1}^{n} a_j + \sum_{j \neq i} b_j})^{b_i s}$ together with the zero-knowledge proofs of the exponent $b_i s$.

  Again at the end of the round, each party P_i checks that the received ZKPs and the received elements h_j for $j \neq i$ are non-identity.

- **Round 3**: Each party P_i computes their secret shares sequence $K_i = (K_{ij})_{j \neq i}$ where $K_{ij} = (h_j (g^{b_j b_i s})^{-1})^{b_i}$.

After the first 3 rounds end, $(n-1, n)$-PATS is completed. When $t = n-r$, taking $(n-1, n)$-PATS as the initial step, (t, n)-PATS will be calculated iteratively. Each iteration step consists of two rounds. Suppose in the $(k-1)^{th}$ iteration step each participant P_i computed its key shares $K_i = \{K_{ij_1 \ldots j_{k-1}}\}$ where j_i's are distinct entries from $I = \{1, \ldots, n\} \backslash \{i\}$. At k^{th} iteration step, in Round 1, each P_i calculates the exponent set $\{(g^{b_{j_1} \cdots b_{j_{k-1}}})^{b_i}\}$ where indices j_r ranges over I and distinct. After the first round of the k^{th} iteration step, the set $\{(g^{b_{j_1} \cdots b_{j_{k-1}}})^{b_i}\}_{j_r \in I}$ will be broadcasted with the keyshare sequence K_i. In the second round of the k^{th} iteration step, P_i computes the new key share sequence $K_i = \{K_{ij_1 \ldots j_k}\}_{j_r \in I}$ where $K_{ij_1 \ldots j_k} = [K_{j_1 \ldots j_k} \times [(g^{b_{j_1} \cdots b_{j_k}})^{b_i s}]^{-1}]^{b_i}$.

In this section, we showed how the (t, n)-threshold generalization of PATS works. In Chapter 5, we will first give the adversary model and define security requirements that PATS shall satisfy. The chapter will end with a detailed analysis of PATS, showing how the mentioned requirements are fulfilled.

CHAPTER 4

ZERO-KNOWLEDGE PROOF SCHEMES ON EQUAL PARTITION ON GROUPS PROBLEM

In this chapter, we first define the Equal Partition Problem on Groups (EPPG). We present a three-pass ZKP based on the EPPG. In the algebraic setup phase, the theoretic approach is explained in detail. Then, we present the practical application of the protocol on elliptic curves. Finally, to reduce the soundness error, we present the second protocol, two-challenge ZKP based on EPPG. We provide the details and a step-by-step explanation of each protocol.

4.1 Equal Partition Problem on Groups (EPPG)

In this section, we will define what the Equal Partition Problem is and the time complexity of the solutions for this problem.

Definition 4.1.1 (The Equal Partition Problem) *The Equal Partition Problem is to decide whether there exists a partition of the input set into two, such that the two proper subsets have the same cardinality and sum up to the same value. The problem is an NP-complete problem [66].*

The partitioning problem (not necessarily equal) on the set of numbers is well-studied. For a set of cardinality n, in the worst case, the time complexity of a naive search is $O(2^n)$. There are heuristic algorithms that can be used to approximate a solution, but they do not guarantee an exact partition and may not always find the optimal solution. Greedy Heuristic Algorithm and (better than that) Karmarkar-Karp Heuristic (Set Differencing) [67] both sort the numbers in the set in descending order and then

develop their strategy. These algorithms have time complexity $O(nlogn)$ but they may fail to find the optimal solution. These two algorithms are extended to complete forms that build a binary tree after sorting the numbers. The complete forms of the algorithms solve the general number partitioning problem optimally but they have still time complexity $O(2^n)$ in the worst case. Also, dynamic programming approaches are solving the problem in pseudo-polynomial time. But for large instances of the Partition Problem, it is infeasible to use this method.

We define the equal partition search problem on a commutative group (EPPG) as follows.

Definition 4.1.2 (Equal Partition Problem on Groups (EPPG)) *Given a subset $S = \{g_1, \ldots, g_n\}$ of a commutative group G and given an element $x \in G$ such that $x^2 = \prod_{g \in S} g$. Find a subset T of S satisfying $|T| = |S \setminus T| = \frac{n}{2}$ and*

$$x = \prod_{g \in T} g = \prod_{g \in S \setminus T} g. \tag{4.1}$$

One should note that EPPG has harder instances than EPP on numbers. As mentioned above, most significant algorithms solving EPP rely on sorting numbers. This does not apply to every commutative group; such as elliptic curve groups. In this article, we proposed two zero-knowledge protocols based on EPPG. Then specifically, we present the first protocol on elliptic curves and implement it. Sorting a given subset S from a group of cryptographically secure elliptic curves is an issue that needs to be discussed. We have an additive cyclic group, say with generator point P. Then the elements in S are of the form kP for some integer k. These integer multiples can be sorted. If one can solve EPP for the power set $Mlt = \{k \mid kP \in S\}$, then he or she can manage to solve the given EPPG on an elliptic curve.

However, a noticeable difficulty that the attacker must solve here is to obtain the set Mlt according to the element order of S. This requires solving $|S|$-many elliptic curve discrete logarithm problems.

4.2 Three-pass ZKP Based on Equal Partition Problem (EPPZKP)

We employ EPPG to design statistical zero-knowledge argument of knowledge schemes that we refer to as Three-pass ZKP Based on Equal Partition Problem (EPPZKP) and Two Challenge ZKP Based on Equal Partition Problem (2EPPZKP). We present EPPZKP in this section and 2EPPZKP in the subsequent section.

We present a statistical zero-knowledge argument of knowledge scheme on equal partition problem over an abstract commutative group. Our protocol is a Stern-like 3-pass protocol with soundness error $2/3$. As a particular example, we demonstrate this scheme over a cyclic group on an elliptic curve. Secondly, in 2EPPZKP, we reduce the challenge cases to $\{0, 1\}$ inspired by the 5-pass identification scheme presented by Sakumoto et al. [9] and reduce the soundness error from $2/3$ to $\frac{1}{2} + \frac{1}{2p-2}$.

4.2.1 Setup Phase

A commutative group G with a subset $S = \{g_1, \ldots, g_n\}$ of G on which equal partition problem hard is constructed. That is to say for a given element $x \in G$ it is hard to find a subset T of S satisfying $|T| = |S \setminus T| = \frac{n}{2}$ and $x = \prod_{g \in T} g = \prod_{g \in S \setminus T} g$. The set S and the element x are publicly known. A prover (Alice) knowing T will convince a verifier (Bob) that she knows a required partition without revealing any information about the partition.

We allocate multiplicative group notation and admit that S is given in an order. It may also be given in formulation. In this case, the prover will fix an order for the elements of S. Through communication with the verifier (Bob), she sends the elements in an order.

Step 1: For the ordered set $\mathbf{S} = \{g_1, \ldots, g_n\}$, Alice's knowledge is a bit-vector $\mathbf{t}$ such that $t_i = 1$ only if $g_i \in T$. Alice executes the following steps; she

(i) chooses n secret random elements $a_i \in G$,

(ii) chooses a random permutation $\alpha \in Sym(n)$,

(iii) computes elements $b_i = a_i g_i$ for all i,

(iv) constructs the permuted length-n tuples $\mathbf{S}^{\alpha} = (g_{\alpha_1}, \ldots, g_{\alpha_n})$, $\mathbf{a}^{\alpha} = (a_{\alpha_1}, \ldots, a_{\alpha_n})$, $\mathbf{b}^{\alpha} = (b_{\alpha_1}, \ldots, b_{\alpha_n})$, $\mathbf{t_r} := \mathbf{t}^{\alpha} = (t_{\alpha_1}, \ldots, t_{\alpha_n})$,

(v) evaluates the products $A = \prod_{i=1}^{n} a_{\alpha_i}^{t_{r_i}}$ and $B = \prod_{i=1}^{n} b_{\alpha_i}^{t_{r_i}}$

(vii) creates commitments to group elements A, B and to the tuples. $C_1 = Comm(A, B)$, $C_2 = Comm(\mathbf{S}^{\alpha})$, $C_3 = Comm(\mathbf{a}^{\alpha})$, $C_4 = Comm(\mathbf{b}^{\alpha})$, $C_5 = Comm(\mathbf{t_r})$

(vi) sends $C_1, \ldots, C_5$ to Bob.

Step 2: Bob chooses a uniform random challenge $b \in \{0, 1, 2\}$ and sends it to Alice.

Step 3: Depending on the value of the challange, Alice executes the following:

- If $b = 0$, then Alice reveals the tuples $\mathbf{S}^{\alpha}, \mathbf{a}^{\alpha}$ and $\mathbf{b}^{\alpha}$.
- If $b = 1$, she reveals the elements A, B and the tuples $\mathbf{a}^{\alpha}, \mathbf{t_r}$.
- If $b = 2$, she reveals A, B and the tuples $\mathbf{b}^{\alpha}, \mathbf{t_r}$.

Step 4: This is the verification step. If Alice fails in any of her claims, Bob rejects her knowledge and terminates the protocol.

- For $b = 0$, Bob checks whether $\mathbf{S}$ and $\mathbf{S}^{\alpha}$ has the same elements and $b_{\alpha_i} = a_{\alpha_i} g_{\alpha_i}$ for all i and the correctness of the commitments $C_2 = Comm(\mathbf{S}^{\alpha})$, $C_3 = Comm(\mathbf{a}^{\alpha})$, $C_4 = Comm(\mathbf{b}^{\alpha})$.
- For $b = 1$, he checks whether $B = Ax$, $A = \prod_{i=1}^{n} a_{\alpha_i}^{t_{r_i}}$, $\sum_{i=1}^{n} t_{r_i} = \frac{n}{2}$ and the correctness of the commitments

  $C_1 = Comm(A, B)$, $C_3 = Comm(\mathbf{a}^{\alpha})$, $C_5 = Comm(\mathbf{t_r})$
- For the last case $b = 2$, he checks the equalities $B = Ax$ and $B = \prod_{i=1}^{n} b_{\alpha_i}^{t_{r_i}}$, $\sum_{i=1}^{n} t_{r_i} = \frac{n}{2}$ and the given commitments $C_1 = Comm(A, B)$, $C_4 = Comm(\mathbf{b}^{\alpha})$, $C_5 = Comm(\mathbf{t_r})$

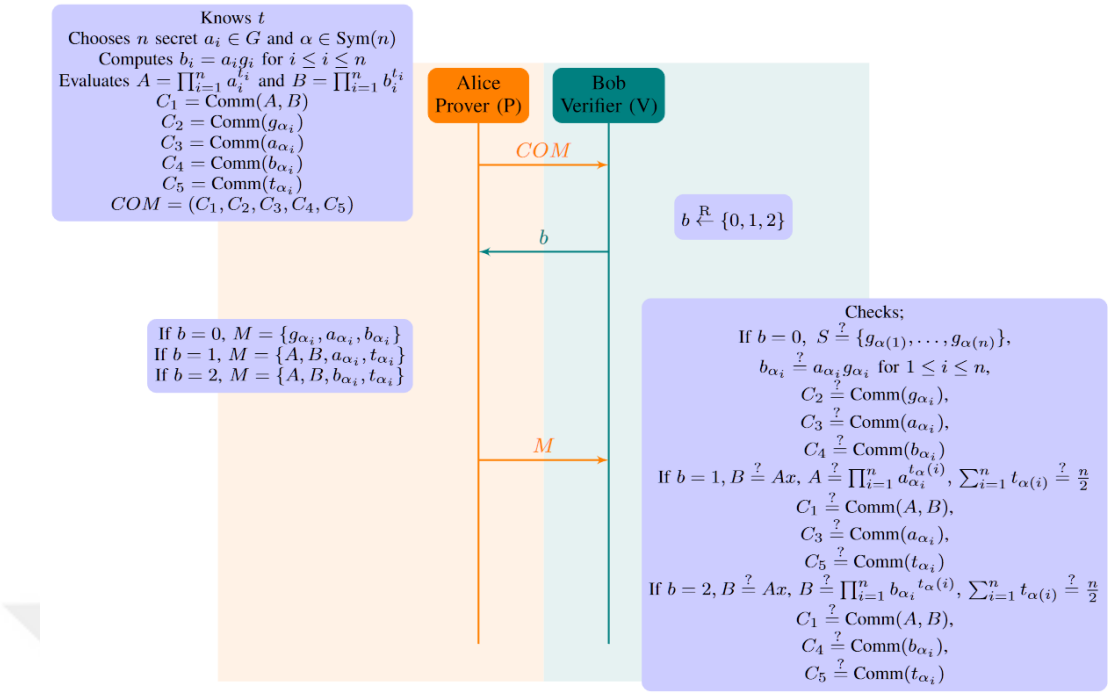


Figure 4.1: EPPZKP: Three-pass ZKP Based on Equal Partition Problem on Groups.

We demonstrate the above scheme on a cyclic group over an elliptic curve which is an attractive choice due to its high-security achievement with smaller cost and easy implementation. As a commitment scheme, we allocate a secure cryptographic hash function and denote it by H.

4.2.2 EPPZKP on Elliptic Curves

A cyclic group G with base point P is chosen on a secure elliptic curve. A subset $S = \{P_1, \ldots, P_n\}$ of G on which equal partition is hard is constructed. The ordered set S and the sum $Q = \sum_{P_i \in T} P_i$ for a solution T of the problem is publicly known. A prover (Alice) knowing T will convince a verifier (Bob) that she knows a required partition giving the sum Q without revealing any information about the partition.

Step 1: Alice's knowledge is a bit-vector $\mathbf{t}$ such that $t_i = 1$ only if $P_i \in T$. She chooses n secret random elements $a_i \in G$ and chooses a random permutation $\alpha \in Sym(n)$. She computes elements $b_i = P_i + a_i$ for all i and evaluates the points $A = \sum_{i=1}^{n} t_{r_i} a_{\alpha_i}$ and $B = \sum_{i=1}^{n} t_{r_i} b_{\alpha_i}$ where $\mathbf{t_r} = \mathbf{t}^{\alpha}$. As commitments she evaluates

hash values

$H_1 = H(A, B)$, $H_2 = H(\mathbf{S}^\alpha)$, $H_3 = H(\mathbf{a}^\alpha)$, $H_4 = H(\mathbf{b}^\alpha)$, $H_5 = H(\mathbf{t_r})$, where $\mathbf{S}^\alpha = (P_{\alpha_1}, \ldots, P_{\alpha_n})$ $\mathbf{a}^\alpha = (a_{\alpha_1}, \ldots, a_{\alpha_n})$, $\mathbf{b}^\alpha = (b_{\alpha_1}, \ldots, b_{\alpha_n})$ and sends them to Bob.

Step 2: Bob chooses a uniform random challenge $b \in \{0, 1, 2\}$ and sends it to Alice.

Step 3:

- If $b = 0$, then Alice reveals the sequences $\mathbf{S}^\alpha, \mathbf{a}^\alpha, \mathbf{b}^\alpha$.
- If $b = 1$, she reveals points A, B and the tuples $\mathbf{a}^\alpha, \mathbf{t_r}$.
- If $b = 2$, she reveals points A, B and the tuples $\mathbf{b}^\alpha, \mathbf{t_r}$.

Step 4:

- For $b = 0$, Bob checks whether sent sequence $\mathbf{S}^\alpha$ has the same points with the set S. Then he checks whether $b_{\alpha_i} = a_{\alpha_i} + P_{\alpha_i}$ for all i and checks the validity of hash values H_2, H_3, H_4.
- For $b = 1$, he checks whether $B = Q + A$ and whether $A = \sum_{i=1}^{n} t_{r_i} a_{\alpha_i}$ and $\sum_{i=1}^{n} t_{r_i} = \frac{n}{2}$ together with the correctness of the hash values H_1, H_3, H_5.
- For the last case $b = 2$, he checks the equalities $B = Q + A$ and $B = \sum_{i=1}^{n} t_{r_i} b_{\alpha_i}$, $\sum_{i=1}^{n} t_{r_i} = \frac{n}{2}$ and the validity of the corresponding hash values H_1, H_4, H_5.

4.3 Two Challenge ZKP Based on Equal Partition Problem (2EPPZKP)

G is a (multiplicative) cyclic group of prime order p. Given $S = \{g_1, \ldots, g_n\}$ is an ordered subset of G with a hard to find equal partition. If we denote the partition as T and $S \setminus T$, Alice's knowledge is a bit-vector $\mathbf{t}$ where $t_i = 1$ only if $g_i \in T$. The element $x = \prod_{i=1}^{n} g_i^{t_i}$ is publicly known.

Step 1: Alice chooses a random permutation $\alpha \in Sym(n)$ and a random length-n string $\mathbf{a}$ consisting of elements of G. She evaluates permuted strings $\mathbf{S_r} = \alpha(\mathbf{S}), \mathbf{t_r} = \alpha(\mathbf{t})$, the element $A = \prod_{i=1}^{n} a_i^{t_{r_i}}$ and the commitments $C_1 = Comm(\alpha, \mathbf{a}), C_2 = Comm(\mathbf{t_r})$. She sends (C_1, C_2, A).

Step 2: Bob selects a random positive integer d and sends it to Alice. To reduce unnecessary evaluations, we restrict $1 \leq d < p$.

Step 3: Alice evaluates and sends the string $\mathbf{u}$ defined by $u_i = a_i {S_{r_i}}^d$.

Step 4: Bob chooses a random challenge $b \in \{0, 1\}$ and sends it to Alice.

Step 5: If $b = 0$, Alice sends α. Otherwise, she sends $\mathbf{t_r}$.

Step 6:

- If $b = 0$, Bob checks whether $C_1 = Comm(\alpha, \mathbf{u}{\mathbf{S_r}}^{-\mathbf{d}})$.
- If $b = 1$, he checks whether the following three equalities hold:
  1. $\prod_{i=1}^{n} u_i^{t_{r_i}} = Ax^d$,
  2. $\sum_{i=1}^{n} t_{r_i} = \frac{n}{2}$
  3. $C_2 = Comm(\mathbf{t_r})$

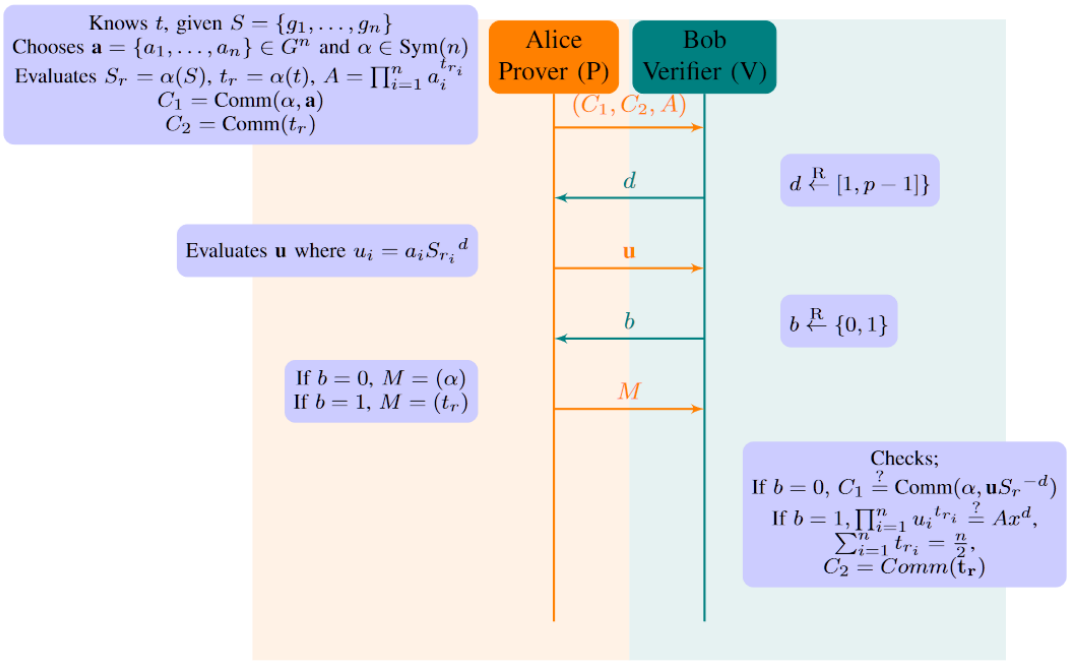


Figure 4.2: Two Challenge ZKP Based on Equal Partition Problem (2EPPZKP)

CHAPTER 5

SECURITY AND PERFORMANCE ANALYSIS

In this chapter, we start with discussing an adversary model stating the capabilities of an attacker, common security requirements of PAKE protocols, and how PATS satisfies the security requirements. In the second part of the chapter, we will focus on the analysis of proposed zero-knowledge protocols. A ZKP protocol must satisfy three properties, called completeness, soundness, and zero-knowledge(ness). Proving completeness is straightforward in our case, but to prove soundness we need to define extractors and to prove zero-knowledge(ness), we take advantage of simulators.

5.1 Security Analysis of PATS

In this section, we present the security analysis of PATS. To understand the requirements we first define an adversary model. The details of each security requirement and how PATS satisfies them are presented with formal proofs.

5.1.1 Adversary Model

To define security requirements and show that PATS satisfies them, it is important to understand the capabilities of an attacker. An adversary A is assumed to have the following capabilities:

1. Similar to conventional PAKE protocols, A is assumed to have full control over the communication channel between parties, eavesdropping, tampering, deleting, and changing messages gathered over the public channel.

2. A can generate all possible values of s since PATS assumes that low-entropy password s is in the range $[1, q-1]$.

3. A is able to learn the previous session keys a_i's and b_i's.

5.1.2 Security Requirements

PATS shall satisfy the following security requirements:

- **Offline dictionary attack resistance:** An active/passive attacker has no advantage in verifying the correctness of password guesses. This means that even if an attacker captures data and tries to guess the password offline, they cannot confirm their guesses are correct without access to the system. For example, even if an attacker obtains hashed passwords, they cannot verify their guesses without the system's hashing mechanism, making it computationally infeasible to crack the passwords.

- **Forward secrecy [68]:** If the password is disclosed later, the current session keys will remain secure. This ensures that past communications are protected even if the password is compromised in the future. For example, each message session generates a unique encryption key in secure messaging apps. Even if an attacker learns the password later, they cannot decrypt past messages because those session keys are unique and unrelated to the compromised password.

- **Known-session security:** If any session is disclosed, other established sessions will not be affected and will remain secure. This means that the compromise of one session's data does not lead to the compromise of data in other sessions. For example, if an attacker manages to decrypt the data from one session, they cannot use this information to decrypt other sessions because each session operates independently with its security parameters.

- **Online dictionary attack resistance:** An active attacker is limited to testing only one password per execution of the protocol. This makes it impractical for attackers to guess passwords through repeated attempts, as the protocol restricts the number of guesses that can be made. For instance, a system can

implement rate limiting or account lockout mechanisms after a certain number of failed attempts, thereby significantly reducing the chances of a successful online dictionary attack.

- **Man-in-the-Middle attack resistance [5]:** Man-in-the-middle attack happens when an active attacker stays in the middle of the communication to relay and alter messages. At the same time, parties believe that they directly communicate with each other. Resistance to this type of attack is achieved through mechanisms like mutual authentication and message integrity checks, which ensure that both parties can verify each other's identity and the integrity of the messages being exchanged. This prevents the attacker from altering or injecting messages without detection.

- **Replay attack resistance [69, 70]:** Replay attack is a man-in-the-middle attack in which the communication is delayed or replayed by a passive attacker. Resistance to replay attacks involves using nonces or timestamps in the protocol to ensure that messages cannot be reused. For example, each message may include a unique identifier or a timestamp that ensures it is valid only for a specific session or time. If an attacker tries to replay a message, the system detects the reuse of the identifier or outdated timestamp and rejects the message.

5.1.3 Security Analysis

Previously, we have defined an adversary model. According to this model, first, an attacker *A* is assumed to have full control over the communication channel between parties, being able to eavesdrop, tamper, delete, and change messages. Since the protocols include zero-knowledge proofs of the exponents a_i's and b_i's, and they depend on the DDH problem, accessing each value to generate K_i's is not feasible. Second, *A* can generate all possible values of s. However, s is not the only component of the secret key K. *A* should also know a_i and b_i values, as well as the value of g. Finally, *A* can learn the previous session keys a_i's and b_i's. Since every time PATS runs, parties are expected to select new a_i and b_i values. Knowing any previous session keys would not provide any advantage.

PATS provides several security features, some of them coming from the existence of zero-knowledge proofs and the use of threshold group key sharing in the protocol. Being a threshold key exchange scheme, failure or malfunction of some entities will not affect the scheme. Owing to the following lemma, we can ensure the resistance of PATS to offline dictionary attacks.

Lemma 5.1.1 *For any party P_k, if $k \neq i$ the exponent $X_i = \sum_{j=1}^{n} a_j + \sum_{j \neq i} b_j$ is a secret random value over $\mathbb{Z}_q$.*

Proof 5.1.1 *a_i and b_i values are uniform randomly selected over the intervals $[0, q-1]$ and $[1, q-1]$, respectively. P_k only knows the value of a_k; the other summands of X_i are unknown to P_k. The value $X_i - a_k$ ranges over the interval $[0, q-1]$. Although b_i values can not be zero, for the sum $\sum_{j \neq i} b_j$, the probability of non-zero values are equal and the probability difference $P(0) - P(x \neq 0) = \frac{1}{(q-1)^2}$ is negligible. Therefore, the value $X_i - a_k$ can be regarded as uniformly distributed over $\mathbb{Z}_q$.* ■

Theorem 5.1.2 *PATS is secure against offline dictionary attacks under the Decision Diffie-Hellman assumption.*

Proof 5.1.2 *We examine this security in two cases; active and passive attacks. First, we will show that a malicious attacker pretending as a participant P_j but does not have the password s, can not distinguish the ciphertext h_i from a random non-identity element of G. This case corresponds to active attacks. At the end of the second round of the protocol, the information that the attacker gathers from P_i is the values $g^{a_i}, g^{b_i}, h_i = (g^{X_i})^{b_i s}$ and the corresponding ZKP's of exponents.*

In the second round, P_i sends $h_i = (g^{X_i})^{b_i s}$ where X_i as given in Lemma 5.1.1. Here P_i allocates the element g^{X_i} as a random generator. The randomness follows from the Lemma 5.1.1. It is well-known that any non-trivial element of a prime order group is a generator [71]. The element g^{X_i} is non-trivial with high probability and this can be guaranteed by P_i by simply checking it. Therefore g^{X_i} is a generator. The password s and the random value b_i are in the interval $[1, q-1]$ and are unknown random values for the attacker. Therefore, by the DDH assumption, the malicious

party cannot distinguish the value h_i, from a random non-identity element in the group.

In the second case, suppose a passive attacker gathers the information floating on the air while two authentic participants P_i and P_j are communicating. The attacker has no better knowledge than the first case. So, it can neither correlate the values h_i and h_j nor distinguish them from a random non-identity element in the group. ■

We know that the original Diffie-Hellman key exchange method is vulnerable to a man-in-the-middle attack. Since round 1 and round 2 of the suggested protocols include zero-knowledge proof of the values b_i and $b_i s$, an active attacker will not be successful without knowing the value of s; this provides **resistance to the man-in-the-middle attacks**.

We previously mentioned that our protocol relies on the hardness assumption of the Decision Diffie-Hellman problem (DDH). Moreover, we noted that p and q are large primes such that q is a divisor of $p-1$. DDH is believed to be intractable in all the groups when $p = kq + 1$ for prime numbers p and q and $q > p^{1/10}$ [72].

In the implementation, we considered p to be a safe prime such that $p = 2q+1$, where q is also a prime [73]. This assumption makes PATS rely on the DDH problem's hardness assumption since $q = (p-1)/2 > p^{1/10}$ for sufficiently big values of p.

Suppose a malicious entity was able to disclose the value of K. In that case, the security of other sessions will not be disrupted. This comes from the necessity that each party in the protocol is expected to select a private integer a_i uniformly at random from $[0, q-1]$ at each run of the scheme. This feature is known as **known-session secrecy** [57].

Using a zero-knowledge protocol for the exponents a_i and b_i is accepted to be secure since a party does not have to reveal these values to other parties and parties who receive g^{a_i} and g^{b_i} will verify that these values belong to the sender.

It is not surprising that PATS, in the spirit of JPAKE, provides similar security requirements. The footprints in JPAKE give the idea, while below we show that PATS provides forward secrecy. Theorem 5.1.3 shows that PATS provides **forward secrecy**

under the Square Computational Diffie-Hellman Assumption (SCDH).

Theorem 5.1.3 *In PATS protocol, under the SCDH assumption, the current session key components will remain secure if the password is disclosed later.*

Proof 5.1.3 *Suppose the password s is captured by a malicious attacker. We will observe that the previously derived K_{ij} components can not be disclosed if the SCDH assumption holds. Note that the PATS protocol is designed in such a way that no key component is the identity element. We consider the key component K_{12}. Mutatis mutandis one may derive the same conclusion for the other key components. To ease the formulation set $A = a_1 + a_2 + a_3$. The idea is if an attacker can obtain $K_{12} = g^{(A+b_3)b_1b_2}$ from the data set $\{g^{a_1}, g^{a_2}, g^{a_3}, g^{b_1}, g^{b_2}, g^{b_3}, g^{(A+b_1+b_2)b_3}, g^{(A+b_1+b_3)b_2}, g^{(A+b_2+b_3)b_1}\}$ then it can compute g^{x^2} from g and g^x for any uniform randomly chosen x value in range $[1, q-1]$ which contradicts the SCDH assumption.*

Let attacker have an algorithm which can return $K_{12} = g^{(A+b_3)b_1b_2}$ for the ordered input set $\{g^A, g^{b_1}, g^{b_2}, g^{b_3}, g^{(A+b_1+b_2)b_3}, g^{(A+b_1+b_3)b_2}, g^{(A+b_2+b_3)b_1}\}$. So for g^x with random exponent x in $[1, q-1]$ the algorithm gives the evaluation

$$f(g^x) = g^{[(-2x+a)(x+d)](x+b)(x+c)} = g^{-x^3+(-c-b+a+d)x^2+(-bc+ac+ab+dc+db)x+abc+bcd} \tag{5.1}$$

for the given data set $\{g^{-2x+a}, g^{x+b}, g^{x+c}, g^{x+d}, g^{(x+a+b+c)(x+d)}, g^{(x+a+b+d)(x+c)}, g^{(x+a+c+d)(x+b)}\}$ where a, b, c, d are arbitrary chosen in $\mathbb{Z}_q$. Similarly, we determine $f(g^{x+1})$. It follows that using the given information one can evaluate the element:

$$g^{x^2} = [f(g^x)f(g^{x+1})^{-1}g^{(2a-2b-2c+2d)x+(-1+a-b-c+d-bc+ac+ab+dc+db+abc+bcd)}]^{\frac{1}{3}}. \tag{5.2}$$

■

PATS is information-theoretically secure [74] since less than the required number of shares will provide no information about the secret. Moreover, the share of each party is as long as the final secret K.

An active attacker is limited to testing/checking only one s value per execution of the protocol since parties will refresh a_i and b_i values at each run of the protocol. PATS prevents **online dictionary attacks**.

In this section, we have given an adversary model and analyzed the security requirements that PATS satisfies. In the next section, the performance analysis of PATS, including computational load and run-time analysis will be provided for each variant of the protocol defined previously. The section will include a comparison of two variants in terms of run-time.

5.2 Performance Analysis of PATS

In this section, we present the performance analysis of PATS. Firstly, we discuss the computation overhead of PATS and then present the running time results of PATS.

Both protocols are implemented in Python on a PC (8 CPU Cores, 16 GB RAM) running macOS Monterey. ZKP proof is not included in the implementation, and parties are not separated to cause messaging costs; i.e., messaging delays are not incorporated.

Prime numbers of different lengths are selected using OpenSSL prime generation with the 'safe' parameter. If the number generated is p, this feature helps us check that $(p-1)/2$ is also a prime number.

5.2.1 PATS in Finite Fields

We first analyze the PATS in finite fields. To better understand the performance, we examine computational load and run-time analysis.

5.2.1.1 Computational Load

A party P_i needs to do the following computations in a round of PATS protocol in finite fields:

- g^{a_i}
- g^{b_i}
- $b_i s$
- $g^{\sum_{j=1}^{3} a_j + \sum_{j \neq i} b_j} = g^{a_1} \times g^{a_2} \times g^{a_3} \times g^{b_j} \times g^{b_k}$
- $h_i = (g^{\sum_{j=1}^{3} a_j + \sum_{j \neq i} b_j})^{(b_i s)}$
- $K_i = (K_{ij}, K_{it})$ where $K_{ij} = (h_j \times (g^{b_j})^{-b_i s}))^{b_i}$

In Round 1, it takes two exponentiations to compute $g^{a_i} \mod p$ and $g^{b_i} \mod p$, and two additional exponentiations will be necessary to compute commitments for the zero-knowledge proofs of the multiples a_i and b_i if Schnorr non-interactive ZKP will be applied. To prove that $g^x = y$, the prover must perform one exponentiation as a commitment. After Round 1, every party should verify two Schnorr ZKPs. For each ZKP, it takes one exponentiation and one multiplication for the verifier to verify and one multiplication and one addition for the prover to generate a response. In Round 2, there will be four multiplications of previous calculations and broadcast messages, one multiplication to compute the power $b_i s$ and one exponentiation to compute h_i value. To compute the zero-knowledge proof of the exponent $b_i s$, additional exponentiation will be necessary to compute a commitment for the zero-knowledge proofs of the multiple $b_i s$. To verify this Schnorr ZKP, each verifier computes one exponentiation and one multiplication, and the prover computes one multiplication to generate a response. In Round 3, to reduce the cost of inverse/-exponents, we suppose that the participant P_i evaluates $q - b_i s$ then gets the value $K_{ij} = \left[h_j \left[(g^{b_j})^{b_i s}\right]^{-1}\right]^{b_i} = \left[h_j (g^{b_j})^{q - b_i s}\right]^{b_i}$. So each key share component costs two exponentiations, one multiplication, and one addition. Regarding communication, TTP-verified (2,3)-PATS starts with $n = 3$ broadcast messages coming from TTP to parties. In Round 1, each party sends a broadcast message to transfer the values $g^{a_i} \mod p$ and $g^{b_i} \mod p$. After Round 1, every party should verify two Schnorr ZKPs, which requires one commitment, one response from the prover, and one challenge from the verifier. In Round 2, two broadcast messages will be sent per party to transfer the values h_i. To compute the zero-knowledge proof of the exponent $b_i s$, the

prover must send two messages, and the verifier must communicate with one message. In Round 3, two broadcast messages will be per party to transfer the values K_i. For the Distributed (2,3)-PATS, the number of messages will be equal to the total number of messages in TTP-verified (2,3)-PATS since there are three broadcast messages in Round 0 of the TTP-verified version. However, in Round 3, it does not require broadcast. Instead, all parties send their share only to TTP. A breakdown of the cost and number of broadcast messages of each round is summarized in Table 5.1.

Table 5.1: The analysis of computational cost of PATS in Finite Fields.

	Cost breakdown	No. of operations per party	No. of broadcast messages
0	Round 0		3^*
1	Round 1	2 E, 2(ZKP)	6 + 2[3]
2	Round 2	1 E, 5 M, (ZKP)	6 + [3]
3	Round 3	4 E, 2 M, 2 A	$6 - 3^*$
4	Total	7 E, 7 M, 2 A, 3(ZKP)	18 + 3[3]

[1] E: exponentiation in $\mathbb{Z}_q$

[2] M: multiplication in $\mathbb{Z}_q$

[3] A: addition in $\mathbb{Z}_q$

[4] (ZKP) = (3 E, 2 M, 1 A): # of operations required for a single ZKP

[5] [x] : # of messages required for a single ZKP

[*] TTP-verified (2,3)-PATS requires 3 broadcast messages from TTP in Round 0, and 3 fewer messages in Round 3

5.2.1.2 Run-time Analysis

The implementation of the proposed protocol in finite fields uses p values of different lengths, and 256-bit q values. As seen in Figure 5.1, the length of p affects the run-time of the protocol significantly. The run-time is close to 0.02 seconds when p is of the length 512 bits, while it reaches close to 0.33 seconds for a random p value of

length 3072 bits.

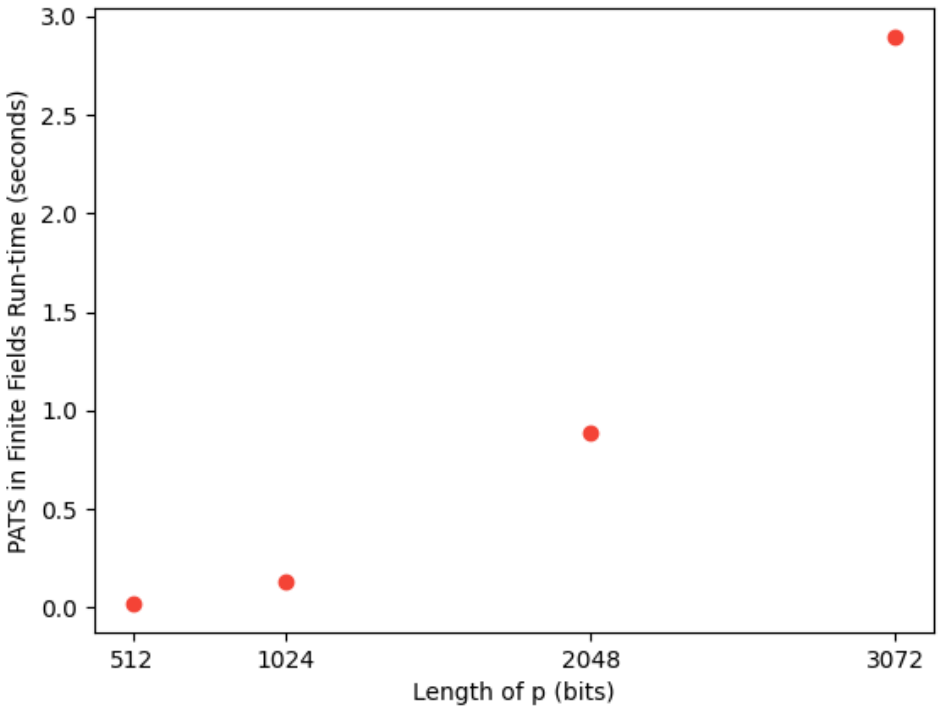


Figure 5.1: Run-time of (2,3)-PATS in finite fields, the results of 100 experiments within the 95% confidence interval are shown.

5.2.2 PATS using Elliptic Curves

Now, we analyze the PATS using elliptic curves. To better understand the performance, we examine computational load and run-time analysis.

5.2.2.1 Computational Load

A party P_i needs to do the following computations in a round of PATS protocol using elliptic curves:

- a_iG
- b_iG

- sb_i
- $(\sum_{j=1}^{3} a_j + \sum_{j \neq i} b_j)G = a_1G + a_2G + a_3G + b_jG + b_kG$
- $H_i = sb_i(\sum_{j=1}^{3} a_j + \sum_{j \neq i} b_j)G$
- $K_i = (K_{ij}, K_{it})$ where $K_{ij} = b_i(H_j - b_j b_i sG)$

In Round 1, it takes two elliptic curve scalar multiplications to compute a_iG and b_iG, and two additional elliptic curve scalar multiplications will be necessary to compute commitments for the zero-knowledge proofs of the multiples a_i and b_i if Schnorr non-interactive ZKP is applied. To prove that $Q = [a]P$ where Q and P are two points and a is a scalar, the prover must perform one scalar multiplication as a commitment. After Round 1, every party should verify two Schnorr ZKPS. Each ZKP takes two scalar multiplications for the verifier to verify and one multiplication and one addition for the prover to generate a response. In Round 2, there will be four point additions of previous calculations and broadcast messages, one multiplication to compute the power sb_i and one scalar multiplication to compute H_i value. To compute the zero-knowledge proof of the exponent b_is, an additional scalar multiplication will be necessary to compute a commitment for the zero-knowledge proof. To verify this Schnorr ZKP, each verifier computes one scalar multiplication and one point addition, and the prover computes one multiplication to generate a response. Round 3 requires three scalar multiplications (one of them is multiplying by -1) and one point addition for each K_{ij} value.

Regarding communication, TTP-verified (2,3)-PATS starts with $n = 3$ broadcast messages from TTP to parties. In Round 1, each party sends a broadcast message to transfer the values a_i and b_i. After Round 1, every party should verify two Schnorr ZKPS, which requires one commitment, one response from the prover, and one challenge from the verifier. In Round 2, two broadcast messages will be sent per party to transfer the values h_i. To compute the zero-knowledge proof of the exponent b_is, the prover must send two messages, and the verifier must communicate with one message. In Round 3, two broadcast messages will be per party to transfer the values K_i. For the Distributed (2,3)-PATS, the number of messages will be equal to the total number of messages in the TTP-verified (2,3)-PATS since there are 3 broadcast

Table 5.2: The analysis of the computational cost of PATS using elliptic curves.

	Cost breakdown	No. of operations per party	No. of broadcast messages
0	Round 0		3^*
1	Round 1	2 SM, 2(ZKP)	6 + 2[3]
2	Round 2	1 SM, 4 PA, 1 M, (ZKP)	6 + [3]
3	Round 3	3 SM, 1 PA	$6 - 3^*$
4	Total	6 SM, 5 PA, 1 M, 3(ZKP)	18 + 3[3]

[1] SM: elliptic curve scalar multiplication

[2] PA: elliptic curve point addition

[3] M: multiplication

[4] A: addition

[5] (ZKP) = (3 SM, 1 PA, 1 M, 1 A): # of operations required for a single ZKP

[6] [x] : # of messages required for a single ZKP

[*] TTP-verified (2,3)-PATS requires 3 broadcast messages from TTP in Round 0, but 3 fewer messages in Round 3.

messages in Round 0 of the TTP-verified version but in Round 3, it does not require broadcast. Instead, all parties send their share only to TTP. A breakdown of each round's cost and the number of broadcast messages is summarized in Table 5.2.

5.2.2.2 Run-time Analysis

The protocol is implemented in Python on a PC (8 CPU Cores, 16 GB RAM) running macOS Monterey. ZKP proof is not included in the implementation, and parties are not separated to cause messaging costs; i.e., messaging delays are not incorporated.

The implementation of the proposed protocol uses different curves, including P-192, P-224, P-256, P-384, and P-521 recommended in the FIPS PUB 186-4 Digital Signature Standard (DSS) report [75]. The length of the prime field p changes with each

selected curve and the run-time of the protocol is affected. However, the run-time for the P-521 curve is still less than half of a second as seen in Figure 5.2.

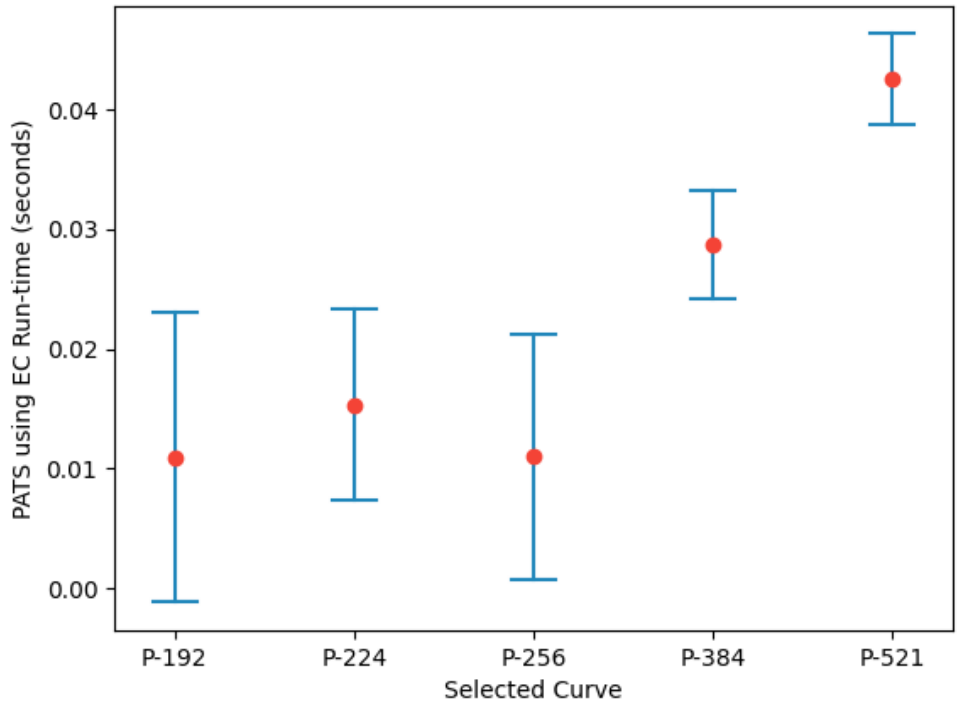


Figure 5.2: Run-time of PATS using ECC, the results of 100 experiments within the 95% confidence interval are shown in the figure.

5.2.3 (t, n)-PATS

In this section, we analyze the generalized version (t, n)-PATS. To better understand the performance, we examine computational load and run-time analysis.

5.2.3.1 Computational Load

A party P_i needs to do the following computations in $(n-1, n)$-PATS protocol in finite fields:

- g^{a_i}

- g^{b_i}
- $b_i s$
- $g^{\sum_{j=1}^{n} a_j + \sum_{j \neq i} b_j}$
- $h_i = (g^{\sum_{j=1}^{n} a_j + \sum_{j \neq i} b_j})^{b_i s}$
- $K_i = (K_{ij})_{j \neq i}$ where $K_{ij} = (h_j \times ((g^{b_j})^{b_i s})^{-1})^{b_i}$

In Round 1, it takes two exponentiation to compute $g^{a_i} \mod p$ and $g^{b_i} \mod p$, and two additional exponentiation will be necessary to compute commitments for the zero-knowledge proofs of the multiples a_i and b_i if Schnorr non-interactive ZKP will be applied. To prove that $g^x = y$, the prover must perform one exponentiation as a commitment. After Round 1, every party should verify two Schnorr ZKPs. For each ZKP, it takes one exponentiation and one multiplication for the verifier to verify and one multiplication and one addition for the prover to generate a response.

In Round 2, the party multiplies $2n - 1$ many group elements obtained and gathered from previous calculations and broadcasted messages. This requires $2n - 2$ group multiplication. A scalar multiplication to compute the power $b_i s$ and an exponentiation to compute h_i value evaluated. To calculate the zero-knowledge proof of the exponent $b_i s$, additional exponentiation will be necessary to compute a commitment for the zero-knowledge proofs of the multiple $b_i s$. To verify this Schnorr ZKP, each verifier computes one exponentiation and one multiplication, and the prover computes one multiplication to generate a response.

In Round 3, the key share sequence of length $n-1$ is evaluated. As in the $(2, 3)$-PATS the exponent $-b_i s$ value is obtained by one addition. To be explicit, P_i evaluates $q - b_i s$. This value will be used for each component in this step and any further iterative steps. After obtaining the $q - b_i s$ value, each component K_{ij} of the key share takes two exponentiation and one multiplication operation.

When the first three rounds end, $(n - 1, n)$-PATS is completed. For $t = n - r$, the corresponding (t, n)-PATS will be calculated iteratively in r-steps. Here $(n - 1, n)$-PATS is the initial step. Each iteration step consists of two rounds. Let us discuss the computational cost of the k-th step. In Round 1, each P_i calculates the exponent

set $\{(g^{b_{j_1}\cdots b_{j_{k-1}}})^{b_i}\}$ where indices j_r ranges over I and distinct. This requires $\binom{n-1}{k-1}$ exponentiation since the base of the exponentiation was already calculated in the previous step. After the first round of the k^{th} iteration step, the set $\{(g^{b_{j_1}\cdots b_{j_{k-1}}})^{b_i}\}_{j_r\in I}$ will be broadcasted with the keyshare sequence K_i obtained in $(k-1)^{th}$ step. In the second round of the k^{th} iteration step, P_i computes the new $K_i = \{K_{ij_1\ldots j_k}\}$ sequence for distinct $j_r \in I$, where $K_{ij_1\ldots j_k} = [K_{j_1\ldots j_k} \times [(g^{b_{j_1}\cdots b_{j_k}})^{b_i s}]^{-1}]^{b_i}$. Recall that the value $q - b_i s$ was already obtained in the initial step. Therefore each component requires one multiplication and two exponentiation. As the length of new key share sequence is $\binom{n-1}{k}$, computational cost of the second round is $\binom{n-1}{k}(1M, 2E)$.

Regarding communication, TTP-verified (t, n)-PATS starts with n broadcast messages coming from TTP to parties. In Round 1, each party sends a broadcast message to transfer the values $g^{a_i} \mod p$ and $g^{b_i} \mod p$. After Round 1, every party should verify two Schnorr ZKPs, which requires one commitment, one response from the prover, and one challenge from the verifier. In Round 2, two broadcast messages will be sent per party to transfer the values h_i. To compute the zero-knowledge proof of the exponent $b_i s$, the prover must send two messages, and the verifier must communicate with one message. In Round 3, two broadcast messages will be per party to transfer the values K_i. For the repeated steps of (t, n)-PATS, $2(n - t - 1)$ broadcast messages will be sent from each party to deliver the sequence $K_i = (K_{ij})_{j\neq i}$ and $g^{b_j b_i} \mod p$. For the Distributed (t, n)-PATS, the number of messages will be equal to the total number of messages in TTP-verified (t, n)-PATS since there are n broadcast messages in Round 0 of the TTP-verified version. However, in Round 3, it does not require broadcast. Instead, all parties send their share only to TTP. A breakdown of the cost and number of broadcast messages of each round is summarized in Table 5.3.

5.2.4 Limitations

Considering the efficiency requirement for (t, n)-threshold scheme, in cases where t is less than $n - 1$, we think the path followed in PATS will be cumbersome. We decided to picture the scheme for distributed variants over finite fields. The generalized version can be phrased straightforwardly on elliptic curves, mutatis mutandis.

Table 5.3: The analysis of the computational cost of (t, n)-PATS in Finite Fields.

	Cost breakdown	No. of operations per party	No. of broadcast messages
0	Initial Step Round 0		n^*
1	Initial Step Round 1	2 E, 2(ZKP)	$n(n-1)+2$[3]
2	Initial Step Round 2	1 E, (2n-1)M, (ZKP)	$n(n-1)+$[3]
3	Initial Step Round 3	1A, (n-1) (2E, M)	$n(n-1)-n^*$
4	Initial Step Total	1A, (2n+1)E, (3n-2)M, 3(ZKP)	$3n(n-1)$
5	k^th Iteration - Round 1	$\binom{n-1}{k-1}E$	
6	k^th Iteration - Round 2	$\binom{n-1}{k}(2E, M)$	$2(n-t-1)$

[1] E: exponentiation in $\mathbb{Z}_q$

[2] M: multiplication in $\mathbb{Z}_q$

[3] A: addition in $\mathbb{Z}_q$

[4] (ZKP) = (3 E, 2 M, 1 A): # of operations required for a single ZKP

[5] [x] : # of messages required for a single ZKP

[*] TTP-verified (t, n)-PATS requires n broadcast messages from TTP in Round 0, and n fewer messages in Round 3

5.2.5 Comparison

Table 5.4, compiled from the report NIST 800-57 Recommendation for Key Management [76], provides security strength level for the approved algorithms of finite-field cryptography and elliptic-curve cryptography. Column 2 states the average run-time of PATS in finite fields / PATS using elliptic curves, respectively. Column 3 states the minimum size of parameters in standardized finite-field cryptography. Column 4 gives the range of f for algorithms based on elliptic-curve cryptography. Using curve P-521 provides 256-bit security strength and a very efficient protocol for PATS. Note that, protocols with a security strength of less than 112 bits are no longer approved, and 192-bit and 256-bit key strengths for FFC algorithms are not considered by NIST due to interoperability and efficiency problems [76].

Table 5.4: Comparable security strengths, and run-time of finite-field cryptography and curves.

Security Strength	Run-time	FFC (DSA, DH, MQV)	ECC (ECDSA, EdDSA, DH, MQV)
≤ 80	0.134/0.011	L = 1024 N = 160	f = 160 - 223 (P-192)
112	0.888/0.015	L = 2048 N = 224	f = 224 - 255 (P-224)
128	2.899/0.011	L = 3072 N = 256	f = 256 - 383 (P-256)
192	-/0.029	L = 7680 N = 384	f = 384-511 (P-384)
256	-/0.043	L = 15360 N = 512	f = 512+ (P-521)

5.3 Analysis of Zero-Knowledge Protocols

In this section, the details security analysis of the proposed zero-knowledge proofs will be given. First, we go over the requirements for a protocol to be a zero-knowledge proof. Then, we formally prove that the protocols satisfy all three properties.

5.3.1 Security Analysis

After presenting the security requirements, we prove that the proposed zero-knowledge proofs satisfy these security requirements.

5.3.1.1 Security Requirements

Informally, an interactive proof scheme (IP) is a two-party protocol between a prover and a verifier such that the requirements **completeness** and **soundness** are satisfied.

Completeness is whenever the statement is TRUE, a prover can convince the verifier. Soundness means whenever the statement is FALSE, a dishonest prover cannot convince the verifier. More formally, an *interactive proof system* [77] for a language L over $\{0,1\}^*$ is a protocol between an interactive pair of Turing machines (P, V) where P, called the prover, has infinite power and V, called the verifier, has polynomial time computation power. An interactive proof system should satisfy the properties:

- Completeness: $Pr[V(x) = Accept \mid x \in L] \geq 1 - \frac{1}{n^k}$
- Soundness: $Pr[V(x) = Accept \mid x \notin L] \leq \frac{1}{n^k}$

where the probabilities are taken over fair coin tosses; $n = |x|$ is the sufficiently large input length and k is a positive constant. The proof system (P, V) is named **public coin**, a.k.a **Arthur-Merlin game**, if V sends each coin tosses. The general approach in an IP is reducing the probability of error by repeating the protocol many times.

Some protocols may require an initial phase called trusted setup. The setup phase can be executed by a trusted third party (TTP) or by employing a secure multi-computing technique in a distributed fashion. Apart from the (instance, witness) pair, some private/common inputs can be assigned to the participants of the protocol, e.g., by a trusted third party (TTP). A common component known by both parties is generally called by common reference string (CRS).

A zero-knowledge proof system is an interactive protocol reinforced with the **zero knowledge property**; the verifier will not learn anything from the proof procedure other than the fact that the statement is true. If in a proof system, the prover convinces the verifier by sending only a single message, then this system is said to be **non-interactive**. By its very nature, originally defined ZKP is a highly repetitive interactive protocol. On the other hand via a CRS or a random oracle model, one may achieve (computational) non-interactive zero-knowledge (NIZKP). A common method to achieve a NIZKP is to transform an interactive protocol to a non-interactive one by Fiat–Shamir heuristic.

The zero-knowledge property, the verifier will not gain any extra information from the interaction, is formulated employing a simulator (a probabilistic polynomial-time

algorithm). The simulation paradigm [78] postulates that "whatever a party can do by itself cannot be considered a gain from interaction with the outside". The simulator's output and the verifier's output are expected to be indistinguishable.

5.3.1.2 Security Analysis of EPPZKP

Lemma 5.3.1 *Assuming EPPG on a given set S is hard and, the commitment scheme $Comm$ is computationally binding and statistically hiding, described EPPZKP is a perfectly complete statistical zero-knowledge argument of knowledge with soundness error $\frac{2}{3}$.*

Proof 5.3.1 (Perfect Completeness) *In the protocol, elements b_i are constructed as $b_i = a_i g_i$ for all i. Any permutation on indices preserves the equality. i.e., $b_{\alpha(i)} = a_{\alpha(i)} g_{\alpha(i)}$.*

Similarly, for any permutation α in the index set,

$$\alpha(S) = S, \tag{5.3}$$

$$x = \prod_{i=1}^{n} g_i^{t_i} = \prod_{i=1}^{n} g_{\alpha_i}^{t_{\alpha_i}} = \prod_{i=1}^{n} g_{\alpha_i}^{t_{r_i}}, \tag{5.4}$$

$$A = \prod_{i=1}^{n} a_i^{t_i} = \prod_{i=1}^{n} a_{\alpha_i}^{t_{\alpha_i}} = \prod_{i=1}^{n} a_{\alpha_i}^{t_{r_i}}, \tag{5.5}$$

$$B = \prod_{i=1}^{n} b_i^{t_i} = \prod_{i=1}^{n} b_{\alpha_i}^{t_{\alpha_i}} = \prod_{i=1}^{n} b_{\alpha_i}^{t_{r_i}}, \tag{5.6}$$

$$B = \prod_{i=1}^{n} b_{\alpha_i}^{t_{r_i}} = \prod_{i=1}^{n} (a_{\alpha_i} g_{\alpha_i})^{t_{r_i}} = \prod_{i=1}^{n} a_{\alpha_i}^{t_{r_i}} \prod_{i=1}^{n} g_{\alpha_i}^{t_{r_i}} = Ax, \tag{5.7}$$

$$\sum_{i=1}^{n} t_{r_i} = \sum_{i=1}^{n} t_{\alpha_i} = \sum_{i=1}^{n} t_i = \frac{n}{2} = |T|. \tag{5.8}$$

Since, $\alpha \in Sym(n)$, applying the permutation to the set S changes only the ordering of the set, as shown in Equation 5.3. The values A and B can be defined as the multiplication of the values a_{α_i}, and b_{α_i} respectively, since the corresponding power of each a_{α_i} and b_{α_i} is preserved by the permutation α, as in Equation 5.4 and Equation 5.5. The sum of indices of the vector t is also preserved under the permutation α

as defined in Equation 5.7. Combining the above three equations (from Equation 5.4 to Equation 5.6), Equation 5.8 gives the equality $B = Ax$. ■

Lemma 5.3.2 (Argument of Knowledge) *Under the assumption that EPPG on the given set* S *is hard and, the commitment scheme* $Comm$ *is computationally binding, protocol EPPZKP given in Figure 4.1 is an argument of knowledge with soundness error* $\frac{2}{3}$.

Proof 5.3.2 *EPPZKP is a* 3*-move public-coin interactive proof. For our protocol, we will describe an extractor* E *which outputs a valid subset* T *on inputs public parameters and three valid transcripts having the same initial commitments. Any three accepted transcripts with different challenges are of the form* $(Comm, 0, RES_0)$, $(Comm, 1, RES_1)$, $(Comm, 2, RES_2)$. *The corresponding responses are* $RES_0 = \{\mathbf{S}^{\alpha 0}, \mathbf{a}^{\alpha 0}, \mathbf{b}^{\alpha 0}\}$, $RES_1 = \{A^1, B^1, \mathbf{a}^{\alpha 1}, \mathbf{t_r}^1\}$ *and* $RES_2 = \{A^2, B^2, \mathbf{b}^{\alpha 2}, \mathbf{t_r}^2\}$, *respectively.*

The extractor E *outputs the subset* $T = \{g^0_{\alpha_i} \mid t^1_{\alpha_i} = 1$ *for* $1 \leq i \leq n\}$.

By the binding property of the commitment scheme, the following equalities hold:

$$a^0_{\alpha_i} = a^1_{\alpha_i} \tag{5.9}$$

$$b^0_{\alpha_i} = b^2_{\alpha_i} \tag{5.10}$$

$$A^1 = A^2 \tag{5.11}$$

$$B^1 = B^2 \tag{5.12}$$

$$t^1_{\alpha_i} = t^2_{\alpha_i} \tag{5.13}$$

for all i.

Using these equalities from Equation 5.9 to Equation 5.13, we show that the output

T is valid.

$$\prod_{g\in T} g = \prod_{i=1}^{n} (g_{\alpha_i}^0)^{t_{\alpha_i}^1} \tag{5.14}$$

$$= \prod_{i=1}^{n} ([a_{\alpha_i}^0]^{-1} b_{\alpha_i}^0)^{t_{\alpha_i}^1} \tag{5.15}$$

$$= \prod_{i=1}^{n} ([a_{\alpha_i}^0]^{-1})^{t_{\alpha_i}^1} \prod_{i=1}^{n} (b_{\alpha_i}^0)^{t_{\alpha_i}^1} \tag{5.16}$$

$$= \prod_{i=1}^{n} ([a_{\alpha_i}^1]^{-1})^{t_{\alpha_i}^1} \prod_{i=1}^{n} (b_{\alpha_i}^2)^{t_{\alpha_i}^2} \tag{5.17}$$

$$= A^{-1}B = x. \tag{5.18}$$

Therefore EPPZKP is 3-out-of-3 special sound. By Lemma A.2.1, it is knowledge sound with knowledge error $\epsilon = \frac{2}{3}$.

■

To show soundness, we defined an extractor E, and this extractor exists for every possible Prover combining the binding property of the commitment scheme. Following the equations, from Equation 5.14 to Equation 5.18 results in the validity of the claimed proof, knowing the existence of a subset T of S satisfying $x = \prod_{g\in T} g = \prod_{g\in S\setminus T} g$. Therefore EPPZKP is 3-out-of-3 special sound. By Lemma A.2.1, it is knowledge sound with knowledge error $\epsilon = \frac{2}{3}$.

Lemma 5.3.3 *Under the assumption that the commitment scheme $Comm$ is statistically hiding, protocol EPPZKP given in Figure 4.1 is black-box statistical zero knowledge.*

Proof 5.3.3 (Zero Knowledge) *Assume that the commitment scheme $Comm$ is statistically hiding. For the proof, we describe a PPT simulator Sim (described in Algortihm 2), given black-box access to a (possibly) malicious verifier V^* and given public input, it outputs a simulated transcript with probability $\frac{2}{3}$. We will see by a sequence of games that owing to the statistically hiding property of $Comm$, the distribution of successful outputs of Sim, are statistically close to those of the real interaction.*

Game 1 (Real World): The algorithm G_1, described in Algorithm 1, gets inputs of Sim and additionally a valid secret position bit vector $\mathbf{t}$.

For both algorithms G_1 and Sim, at each iteration, the probability of output $\perp$ is $\frac{1}{3}$.

Suppose the algorithms gives successful outputs (i.e., $b \neq \bar{b}$). The distribution over the commitments generated by $G1$ and a real prover P are identical as they both use a valid secret $\mathbf{t}$*, random permutations over $Sym(n)$ and follow the same commitment algorithm. Therefore for a successful iteration, output generated by algorithm G_1 identically distributed to real life conversation $View\langle P(\mathbf{t}, pp), V^*(pp)\rangle$.*

Now case by case we will discuss the reason why the output (COM, b, RES) of G_1 and Sim are statistically indistinguishable. Both algorithms runs in the same way accept Stage 2. They predict the value that V^* will not choose as challenge. In each case, as $G1$ uses the secret $\mathbf{t}$ *bit-vector and Sim uses a fake secret bit-vector t^* chosen uniform randomly where both vectors are of weight $\frac{n}{2}$.*

First note that over the set of integers $\mathbf{Z}_p = \{0, 1, \ldots, p-1\}$ modulo prime number p, sum of two uniform random elements results in a uniform random element. Also if we fix an element $m \in \mathbf{Z}_p$, adding m with a uniform random element in $\mathbf{Z}_p$ will give us a uniform random element in $\mathbf{Z}_p$. As $G = \{g^0, g^1, \ldots, g^{p-1}\}$ for any non-identity element g, product of two element in G is equivalent to adding two numbers in $\mathbf{Z}_p$. Therefore product of two uniform random elements or, product of a non-identity fixed element with a uniform random element in G gives a uniform random element.

Case 1: $\bar{b} = 0$

$G1$ and Sim choose uniform random elements $a_1, \ldots, a_n$ from G and evaluate $A = \prod_{i=1}^{n} a_{\alpha_i}^{t_{r_i}}$ in the same manner using $\mathbf{t}$ *and* $\mathbf{t}^*$ *respectively. Therefore evaluation of A values have the same distribution. It is equivalent to choosing a uniform random element. Then, for all i, $G1$ computes the elements $b_i = a_i g_i$ as a product of a fixed element and a uniform random element from G. Therefore, distribution of the tuple* $\mathbf{b}$ *and also its image* $\mathbf{b}^\alpha$ *under a random permutation α evaluated by $G1$ are uniformly distributed among the group G. Sim chooses $b_1, \ldots, b_{k-1}, b_{k+1}, \ldots, b_n$ uniform randomly and evaluates $b_k = (b_1 \ldots b_{k-1} b_{k+1} \ldots b_n)^{-1} xA$ for some index k. However b_k can be treat as a uniform random element as it is a product of uniform random*

Algorithm 1 Algorithm $G1$ for EPPZKP

procedure $G_1(x, \mathbf{t})$ ▷ x: Public parameters and $\mathbf{t}$ as input
  $\bar{b} \xleftarrow{\$} \{0, 1, 2\}$ ▷ Stage 1: Predict the challenge value.
  $\mathbf{a} = (a_1, \ldots, a_n) \xleftarrow{\mathrm{R}} G^n$ ▷ Stage 2: Fix the randomness and masking.
  $\alpha \xleftarrow{\mathrm{R}} Sym(n)$
  $i \leftarrow 1$
  while $i < n + 1$ **do**
    $b_i \leftarrow a_i g_i$
    $i \leftarrow i + 1$
  end while
  $\mathbf{S}^\alpha \leftarrow (g_{\alpha_1}, \ldots, g_{\alpha_n})$
  $\mathbf{a}^\alpha \leftarrow (a_{\alpha_1}, \ldots, a_{\alpha_n})$
  $\mathbf{b}^\alpha \leftarrow (b_{\alpha_1}, \ldots, b_{\alpha_n})$
  $\mathbf{t_r} := \mathbf{t}^\alpha \leftarrow (t_{\alpha_1}, \ldots, t_{\alpha_n})$
  $A \leftarrow \prod_{i=1}^{n} {a_{\alpha_i}}^{t_{r_i}}$
  $B \leftarrow \prod_{i=1}^{n} {b_{\alpha_i}}^{t_{r_i}}$
  $C_1 \leftarrow Comm(A, B)$ ▷ Stage 3: Create commitments as in the real protocol.
  $C_2 \leftarrow Comm(S^\alpha)$
  $C_3 \leftarrow Comm(a^\alpha)$
  $C_4 \leftarrow Comm(b^\alpha)$
  $C_5 \leftarrow Comm(\mathbf{t_r})$
  $COM \leftarrow (C_1, C_2, C_3, C_4, C_5)$
    ▷ Stage 4: Obtain the challenge b via oracle query with input COM
  if $b = \bar{b}$ **then** ▷ Stage 5: Response
    return $\perp$
  end if
  if $b = 0$ **then** ▷ Stage 6: Follow the protocol response stage
    $RES \leftarrow \{\mathbf{S}^\alpha, \mathbf{a}^\alpha, \mathbf{b}^\alpha\}$
  else if $b = 1$ **then**
    $RES \leftarrow \{A, B, \mathbf{a}^\alpha, \mathbf{t_r}\}$
  else if $b = 2$ **then**
    $RES \leftarrow \{A, B, \mathbf{b}^\alpha, \mathbf{t_r}\}$
  end if
  return (COM, b, RES)
end procedure

elements with a fixed element x in G. The last element $B = \prod_{i=1}^{n} b_{\alpha_i}^{t_{r_i}}$ evaluated in the same way with elements having the same distribution. So, obtained B values have also identical distribution. For the case $\bar{b} = 0$, algorithms $G1$ and Sim output (COM, b, RES) if either $b = 1$ or $b = 2$. That is $RES = \{A, B, \mathbf{a}^{\alpha}, \mathbf{t_r}\}$ or $RES = \{A, B, \mathbf{b}^{\alpha}, \mathbf{t_r}\}$ with the same probability for both algorithms. As we discussed above, these values have identical distribution.

Case 2: $\bar{b} = 1$

In this case, the responses output by both algorithms are either $RES = \{\mathbf{S}^{\alpha}, \mathbf{a}^{\alpha}, \mathbf{b}^{\alpha}\}$ or $RES = \{A, B, \mathbf{b}^{\alpha}, \mathbf{t_r}\}$. $G1$ and Sim choose a uniform random permutation $\alpha \in Sym(n)$ and construct the tuple $\mathbf{S}^{\alpha} = (g_{\alpha_1}, \ldots, g_{\alpha_n})$. Sim randomly chooses the tuple $\mathbf{b}^{\alpha}$ then evaluates $B = \prod_{i=1}^{n} b_{\alpha_i}^{t_{r_i}}$. As we discussed above, corresponding values evaluated in $G1$ can also be treat like uniform random choices. Sim evaluates elements $A = Bx^{-1}$ and $a_{\alpha_i} = g_{\alpha_i}^{-1} b_{\alpha_i}$ (for all i) as product of a fixed element with a random element. So these elements distributed equivalently with the choices in $G1$.

Case 3: $\bar{b} = 2$

This case was designed with the same strategy as the previous case, with the a and b values changing roles.

In each case, the challenge and response parts of the outputs are statistically close. Therefore, if there is a distinguisher D of outputs of algorithms $G1$ and Sim with non-negligible advantage, then D should provide an algorithm breaking the (statistically) hiding property of Com.

■

5.3.1.3 Security Analysis of 2EPPZKP

Under the assumption that the commitment scheme Comm is computationally binding and statistically hiding 2EPPZKP is an honest verifier zero-knowledge argument of knowledge.

Algorithm 2 The algorithm of simulator Sim for EPPZKP.

procedure SIM1(x) ▷ x: Public parameter as input
  $\bar{b} \xleftarrow{\$} \{0,1,2\}$ ▷ Stage 1
  $t^* = (t_1 \dots t_n) \xleftarrow{\$} \{0,1\}^n$
  while weigth(t^*) $\neq \frac{n}{2}$ **do** ▷ Check the weight of t^*.
    $t^* \xleftarrow{\$} \{0,1\}^n$
  end while
  $r = (r_1 \dots r_n) \xleftarrow{\$} \{0,1\}^n$
  while weigth(r) $\neq \frac{n}{2}$ **do** ▷ Check the weight of r.
    $r \xleftarrow{\$} \{0,1\}^n$
  end while
  if $\bar{b} = 0$ **then** ▷ Stage 2
    $k \leftarrow 0$
    while $t_k \neq 1$ **do**
      $k \leftarrow k + 1$
    end while
    $\mathbf{a} = (a_1, \dots, a_n) \xleftarrow{\mathrm{R}} G$
    $\mathbf{b} = (b_1, \dots, b_{k-1}, b_{k+1}, \dots, b_n) \xleftarrow{\mathrm{R}} G$
    $A \leftarrow \prod_{i=1}^{n} a_i^{r_i}$
    $b_k \leftarrow (b_1, \dots b_{k-1}, b_{k+1}, \dots, b_n)^{-1} xA$
    $B \leftarrow \prod_{i=1}^{n} b_i^{r_i}$
    $C_1 \leftarrow Comm(A, B), C_2 \leftarrow Comm(g_i), C_3 \leftarrow Comm(a_i)$
    $C_4 \leftarrow Comm(b_i), C_5 \leftarrow Comm(t_i)$.
  else if $\bar{b} = 1$ **then**
    $\alpha \xleftarrow{\mathrm{R}} Sym(n)$
    $\mathbf{d} = (d_1, \dots, d_n) \xleftarrow{\mathrm{R}} G$
    $b_{\alpha_i} \leftarrow d_i$
    $t_{\alpha_i} \leftarrow r_i$
    $B \leftarrow \prod_{i=1}^{n} b_\alpha(i)^{t_\alpha(i)}$,
    $A \leftarrow Bx^{-1}$
    $a_{\alpha(i)} \leftarrow g_{\alpha(i)}^{-1} b_{\alpha(i)}$
    $C_1 \leftarrow Comm(A, B), C_2 \leftarrow Comm(g_{\alpha_i}), C_3 \leftarrow Comm(a_{\alpha_i})$
    $C_4 \leftarrow Comm(b_{\alpha_i}), C_5 \leftarrow Comm(t_{\alpha_i})$.

else if $\bar{b} = 2$ **then**

$\alpha \xleftarrow{\mathrm{R}} Sym(n)$

$\mathbf{d} = (d_1, \ldots, d_n) \xleftarrow{\mathrm{R}} G$

$a_{\alpha_i} \leftarrow d_i$

$t_{\alpha_i} \leftarrow r_i$

$A \leftarrow \prod_{i=1}^{n} a_\alpha(i)^{t_\alpha(i)}$,

$B \leftarrow xA$

$b_{\alpha(i)} \leftarrow a_{\alpha(i)} g_{\alpha(i)}$

$C_1 \leftarrow Comm(A, B)$, $C_2 \leftarrow Comm(g_{\alpha_i})$, $C_3 \leftarrow Comm(a_{\alpha_i})$

$C_4 \leftarrow Comm(b_{\alpha_i})$, $C_5 \leftarrow Comm(t_{\alpha_i})$.

end if

$COM \leftarrow (C_1, C_2, C_3, C_4, C_5)$

if $b = \bar{b}$ **then** ▷ Stage 3

return $\perp$

end if

if $b = 0$ **then** ▷ Stage 4

$M \leftarrow \{g_{\alpha_i}, a_{\alpha_i}, b_{\alpha_i}\}$

else if $b = 1$ **then**

$M \leftarrow \{A, B, a^\alpha, t^\alpha\}$

else if $b = 2$ **then**

$M \leftarrow \{A, B, b_{\alpha_i}, t_{\alpha_i}\}$

end if

return (COM, b, M)

end procedure

Proof 5.3.4 (Completeness) *The proof is straightforward. An honest prover with the knowledge of bit-vector* $\mathbf{t}$ *can give correct responses to the verifier. We give detailed equalities to convince the reader.*

$$\prod_{i=1}^{n} u_i^{t_{r_i}} = \prod_{i=1}^{n} (a_i {S_{r_i}}^d)^{t_{r_i}} = \prod_{i=1}^{n} {a_i}^{t_{r_i}} \prod_{i=1}^{n} ({S_{r_i}}^{t_{r_i}})^d \tag{5.19}$$

$$= A \prod_{i=1}^{n} ({g_i}^{t_i})^d = Ax^d \tag{5.20}$$

$$\sum_{i=1}^{n} t_{r_i} = \sum_{i=1}^{n} t_{\alpha(i)} = \sum_{i=1}^{n} t_i = \frac{n}{2} \tag{5.21}$$

■

A verifier will have the value of values u_i and $\mathbf{t_r}$, following the equations, from Equation 5.19 to Equation 5.21, helps him to check that the following three equalities hold:

1. $\prod_{i=1}^{n} u_i^{t_{r_i}} = Ax^d$,
2. $\sum_{i=1}^{n} t_{r_i} = \frac{n}{2}$
3. $C_2 = Comm(\mathbf{t_r})$

Proof 5.3.5 (Argument of Knowledge) *Under the assumption that EPPG on a given set* S *is hard and the used commitment function* $Comm$ *is computationally binding, the soundness error of 2EPPZKP at each round is* $\frac{1}{2} + \frac{1}{2p-2}$. *We discuss a strategy for a dishonest PPT prover* P^* *can develop.* P^* *uses a random bit-vector* $\mathbf{t}^*$ *of weight* $\frac{n}{2}$ *as its secret and predicts the first challenge* $d^* \in \{1, \ldots, p\}$. P^* *chooses random values* $\alpha, \mathbf{a}$ *and evaluates* $\mathbf{S_r} = \alpha(\mathbf{S}), \mathbf{t_r} = \alpha(\mathbf{t}^*), A^* = \prod_{i=1}^{n} a_i^{t_{r_i}} (x^{-1} \prod_{i=1}^{n} S_{r_i}^{t_{r_i}})^{d^*}$ *and the commitments* $c_1 = Comm(\alpha, \mathbf{a}), c_2 = Comm(\mathbf{t_r})$. *After these computations,* P^* *follows the protocol. More precisely,* P^* *sends* c_1, c_2, A^* *to Bob. Receiving the actual random* d *value, it responds with the string* $\mathbf{u}$ *entries constructed as* $u_i = a_i {S_{r_i}}^d$. *Bob sends his challenge* b. *If he asks for* $b = 0$, *regardless of what the actual* d *value is,* P^* *will be able to give an accepted response pair* $\mathbf{a}$ *and* α. *If the challenge is* $b = 1$, *the malicious prover* P^* *can succeed only if they are lucky*

enough to get $d^* = d$. They send $\mathbf{t_r}$. From the checklist, checking that $\sum_{i=1}^{n} t_{r_i} = \frac{n}{2}$ and $C_2 = Comm(\mathbf{t_r})$ are obvious to prove with the generated value $\mathbf{t_r} = \alpha(\mathbf{t}^*)$. To check $\prod_{i=1}^{n} u_i^{t_{r_i}} = Ax^d$:

$$\prod_{i=1}^{n} u_i^{t_{r_i}} = \prod_{i=1}^{n} (a_i {S_{r_i}}^d)^{t_{r_i}} \tag{5.22}$$

$$= x^d [\prod_{i=1}^{n} {a_i}^{t_{r_i}} (x^{-1} \prod_{i=1}^{n} {S_{r_i}}^{t_{r_i}})^d] \tag{5.23}$$

$$= x^d [\prod_{i=1}^{n} {a_i}^{t_{r_i}} (x^{-1} \prod_{i=1}^{n} {S_{r_i}}^{t_{r_i}})^{d^*}] \tag{5.24}$$

$$= A^* x^d \tag{5.25}$$

To sum up, with the above strategy, P^* achieves an accepted response in two cases. Either Bob sends "$b = 0$" or "$b = 1$ and $d^* = d$." So success probability of P^* is $\frac{1}{2} + \frac{1}{2(p-1)} = \frac{p}{2p-2}$.

Next, we aim to show that this value is the upper limit for any cheating strategy as long as EPPG on the given set S is hard and the used commitment function $Comm$ is computationally binding. Suppose there exists a cheating strategy that a malicious prover P^* can succeed with "non-negligible" higher probability, say $\frac{p}{2p-2} + \epsilon$. This assumption will lead us to a contradiction that there exists a polynomial time probabilistic adversary A which either finds a solution vector $\mathbf{t}$ to the given problem or finds a collision for $Comm$. Knowledge extractor rewinds the protocol for the same commitment string c_1, c_2, A and ask all possible question pairs (d, b). Then among $2p - 2$ questions, with overwhelming probability, the cheating strategy succeeds in $p + 1$ questions. By pigeonhole principle, for at least two d challenges say d_1, d_2, malicious prover P^* gives correct answers for both cases $b = 0, 1$.

Note that in each case the same randomness and knowledge are considered. So values $\alpha, \mathbf{t}, \mathbf{S_r} = \alpha(\mathbf{S}), \mathbf{t_r} = \alpha(\mathbf{t}), A = \prod_{i=1}^{n} a_i^{t_{r_i}}$ and $c_1 = Comm(\alpha, \mathbf{a}), c_2 = Comm(\mathbf{t_r})$ are fixed in all four cases.

The query pairs and answers are shown in Table 5.5.

Table 5.5: Query pairs and answers.

(d, b)	Response 1:$\mathbf{u}$	Response 2
$(d_1, 0)$	$\mathbf{u_1} = (\mathbf{u_{1i}}) = (\mathbf{a_i S_{r_i}}^{\mathbf{d_1}})$	α
$(d_1, 1)$	$\mathbf{u_1} = (\mathbf{u_{1i}}) = (\mathbf{a_i S_{r_i}}^{\mathbf{d_1}})$	$\mathbf{t_r}$
$(d_2, 0)$	$\mathbf{u_2} = (\mathbf{u_{2i}}) = (\mathbf{a_i S_{r_i}}^{\mathbf{d_2}})$	α
$(d_2, 1)$	$\mathbf{u_2} = (\mathbf{u_{2i}}) = (\mathbf{a_i S_{r_i}}^{\mathbf{d_2}})$	$\mathbf{t_r}$
$\ldots$	$\ldots$	$\ldots$
$(d_{p-1}, 0)$	$\mathbf{u_{p-1}} = (\mathbf{u_{(p-1)i}}) = (\mathbf{a_i S_{r_i}}^{\mathbf{d_{p-1}}})$	α
$(d_{p-1}, 1)$	$\mathbf{u_{p-1}} = (\mathbf{u_{(p-1)i}}) = (\mathbf{a_i S_{r_i}}^{\mathbf{d_{p-1}}})$	$\mathbf{t_r}$

Without loss of generality, let us assume that $d_1 > d_2$. From the equality:

$$Comm(\alpha, \mathbf{u_1 S_r}^{-\mathbf{d_1}}) = c_1 = Comm(\alpha, \mathbf{u_2 S_r}^{-\mathbf{d_2}}) \tag{5.26}$$

Either A gets a collision for $Comm$ or the equality $\mathbf{u_1 S_r}^{-\mathbf{d_1}} = \mathbf{u_2 S_r}^{-\mathbf{d_2}}$. Since $Comm$ is computationally binding, the latter case must be valid. That is;

$$\mathbf{u_1 (u_2)^{-1}} = \mathbf{S_r}^{d_1 - d_2}. \tag{5.27}$$

This implies that the following holds:

$$x^{d_1-d_2} = Ax^{d_1}(Ax^{d_2})^{-1} \tag{5.28}$$

$$= \prod_{i=1}^{n} u_{1i}^{t_{r_i}} (\prod_{i=1}^{n} u_{2i}^{t_{r_i}})^{-1} \tag{5.29}$$

$$= \prod_{i=1}^{n} (u_{1i} u_{2i}^{-1})^{t_{r_i}} \tag{5.30}$$

$$= \prod_{i=1}^{n} (S_{r_i}^{d_1-d_2})^{t_{r_i}} \tag{5.31}$$

$$= (\prod_{i=1}^{n} (S_{r_i})^{t_{r_i}})^{d_1-d_2}. \tag{5.32}$$

This assures that adversary A obtained the required solution $\alpha^{-1}(\mathbf{t_r})$. ■

Proof 5.3.6 (Zero Knowledge) *In Algorithm 3, we present a PPT simulator $Sim2$ On given public input, $Sim2$ outputs a simulated transcript between an honest prover and an honest verifier.*

Under the assumption that the commitment scheme $Comm$ is statistically hiding, the distribution of the outputs is statistically close to those of the real interaction.

■

5.3.2 Performance Analysis

Now, we present the performance analysis of the EPPZKP protocol. First, we explain the computation cost. Second, we present the experimental performance of the protocol.

5.3.2.1 Computational Performance

First, we analyze the computational cost of three-pass ZKP Based on the Equal Partition Problem (EPPZKP). The general protocol, the algebraic setup phase has 4 steps. In Step 1, it takes n multiplications for Alice to compute the elements $b_i = a_i g_i$ for all i values. Before step 1 ends, Alice creates commitments of 5 different values. In Step 4, when $b = 0$, Bob needs to calculate $a_{\alpha_i} g_{\alpha_i}$ to check whether the result is equal to b_{α_i}, then commitments C_2, C_3 and C_4 will be created by Bob. When $b = 1$, or $b = 2$, the number of computations is the same. It takes one multiplication to calculate Ax, and $(\frac{n}{2} - 1)$ multiplication to calculate the value of A, or B. Finally, Bob needs to do $n-1$ addition operation in group G to check whether $\sum_{i=1}^{n} t_{\alpha(i)} = \frac{n}{2}$ holds. Table 5.6 summarizes a breakdown of each round's cost.

Second, the EPPZKP on elliptic curves will be analyzed. In Step 1, it takes n elliptic curve point addition for Alice to compute the elements $b_i = P_i + a_i$ for all i values. Before step 1 ends, Alice evaluates hashes of 5 different values. In Step 4, when $b = 0$, Bob needs to calculate $a_{\alpha_i} + P_{\alpha_i}$ to check whether the result is equal to b_{α_i}, then checks the validity of the hashes H_2, H_3 and H_4. When $b = 1$, or $b = 2$, the number of computations is the same. It takes one elliptic curve point addition to calculate $Q + A$, and $(\frac{n}{2} - 1)$ elliptic curve point addition to calculate the value of A, or B. Finally, Bob needs to do $n - 1$ addition operation in group G to check whether

Algorithm 3 The algorithm of simulator $Sim2$ for 2EPPZKP

procedure SIM2(x) ▷ x: Public parameter as input
  $d \xleftarrow{\$} [1, p-1]$ ▷ Stage 1
  $b \xleftarrow{\$} \{0,1\}$
  $t^* = (t_1 \dots t_n) \xleftarrow{\$} \{0,1\}^n$
  while weigth$(t^*) \neq \frac{n}{2}$ **do** ▷ Check the weight of t^*.
    $t^* \xleftarrow{\$} \{0,1\}^n$
  end while
  $\alpha \xleftarrow{\mathrm{R}} Sym(n)$ ▷ Stage 2
  $\mathbf{a} \xleftarrow{\mathrm{R}} G^n$
  $\mathbf{S_r} \leftarrow \alpha(\mathbf{S})$
  $\mathbf{t_r} \leftarrow \alpha(\mathbf{t}^*)$
  if $b = 0$ **then** ▷ Stage 3
    $A \leftarrow \prod_{i=1}^{n} a_i^{t_{r_i}}$
  else if $b = 1$ **then**
    $A \leftarrow \prod_{i=1}^{n} a_i^{t_{r_i}} (x^{-1} \prod_{i=1}^{n} S_{r_i}^{t_{r_i}})^d$
  end if
  $C_1 \leftarrow Comm(\alpha, \mathbf{a})$ ▷ Stage 4
  $C_2 \leftarrow Comm(\mathbf{t_r})$
  $i \leftarrow 0$
  while $i < n$ **do**
    $u_i \leftarrow a_i {S_{r_i}}^d$
    $i \leftarrow i + 1$
  end while
  if $b = 0$ **then** ▷ Stage 5
    $M \leftarrow \alpha$
  else if $b = 1$ **then**
    $M \leftarrow t_r$
  end if
  return $(C_1, C_2, A, d, \mathbf{u}, b, M)$
end procedure

$\sum_{i=1}^{n} t_{\alpha(i)} = \frac{n}{2}$ holds. Table 5.7 summarizes a breakdown of each round's cost.

Finally, we analyze the computational performance of two Challenge ZKP Based on the Equal Partition Problem (2EPPZKP). This protocol has 6 steps in total. In Step 1, it takes $(\frac{n}{2} - 1)$ multiplications for Alice to compute the value of A. Before step 1 ends, Alice evaluates commitments of 3 different values. In Step 3, Alice needs to complete n exponentiations and n multiplication to evaluate the value of **u**. In step 6, when $b = 0$, Bob needs to evaluate a single commitment C_1. When $b = 1$, it takes one multiplication and one exponentiation to calculate Ax^d, and $(\frac{n}{2} - 1)$ multiplication to calculate $\prod_{i=1}^{n} {u_i}^{t_{r_i}}$. Finally, Bob needs to evaluate the commitment C_2. Table 5.8 summarizes a breakdown of each round's cost.

Table 5.6: The analysis of the computational cost of EPPZKP.

Step No	Alice (Prover)	Bob (Verifier)	
		$b = 0$	$b = 1$ or $b = 2$
Step 1	nM + 2($\frac{n}{2} - 1$)M + 5Comm		
Step 2			
Step 3			
Step 4		nM + 3Comm	1M + $(\frac{n}{2} - 1)$M + $(n - 1)$A + 3Comm
Total	$(2n - 2)$M + 5Comm	nM + 3Comm + [$-\frac{n}{2}$M + $(n - 1)$A]	

[1] A: Addition in group G

[2] M: Multiplication in group G

[3] Comm: Commitment Operation

[4] []: Required operational depending on the flow of the protocol

Table 5.7: The analysis of the computational cost of EPPZKP on elliptic curves.

Step No	Alice (Prover)	Bob (Verifier)	
		$b = 0$	$b = 1$ or $b = 2$
Step 1	nPA + 2($\frac{n}{2} - 1$)PA + 5H		
Step 2			
Step 3			
Step 4		nPA	1PA + ($\frac{n}{2} - 1$)PA + $n - 1$A + 3H
Total	$(2n - 2)$PA + 5H	nPA + 3H [$-\frac{n}{2}$PA + $n - 1$A]	

[1] A: Addition in group G

[2] PA: Elliptic Curve Point Addition

[3] H: Hashing Operation

[4] []: Required operational depending on the flow of the protocol

Table 5.8: The analysis of the computational cost of 2EPPZKP.

Step No	Alice (Prover)		Bob (Verifier)	
	$b = 0$	$b = 1$	$b = 0$	$b = 1$
Step 1	$(\frac{n}{2} - 1)$M + 3Comm			
Step 2				
Step 3	n(E + M)			
Step 4				
Step 5				
Step 6			1Comm	1E + 1M + $(\frac{n}{2} - 1)$M + $n - 1$A + 1Comm
Total	$(\frac{n}{2} - 1)$M + n(E + M) + 3Comm		1Comm + [1E + 1M + $(\frac{n}{2} - 1)$M + $\frac{n}{2}$A]	

[1] E: Exponentiation in group G

[2] M: Multiplication in group G

[3] Comm: Commitment Operation

5.3.2.2 Experimental Performance

The protocol is implemented in Go [79] and the results are collected on a PC (8 CPU Cores, 16 GB RAM) running macOS Monterey.

The implementation of the EPPZKP protocol using different curves, including P-224, P-256, P-384, and P-521 as recommended in the FIPS PUB 186-4 Digital Signature Standard (DSS) report [75]. The length of the prime field p changes with each selected curve, and the run-time of the protocol is affected as seen in Figure 5.3. The resulting values are the average of 100 experiments. The results have three components. In the commitgen phase, Alice generates commitments. In the commit phase, Bob sends a challenge to Alice and Alice replies. In the prove phase, Bob checks the validity of the information that Alice sent. The sum of time that is required for these three phases constitutes the total run-time of a protocol for a single round.

Table 5.9: Comparable security strengths, and run-time of EPPZKP for the selected curves.

Curve	Security Strength	Commitgen (ms)	Commit (ms)	Prove (ms)
P-224	112	53.342	0.233	1.392
P-256	128	22.591	0.205	1.163
P-384	192	178.050	0.314	3.125
P-521	256	481.065	0.389	6.203

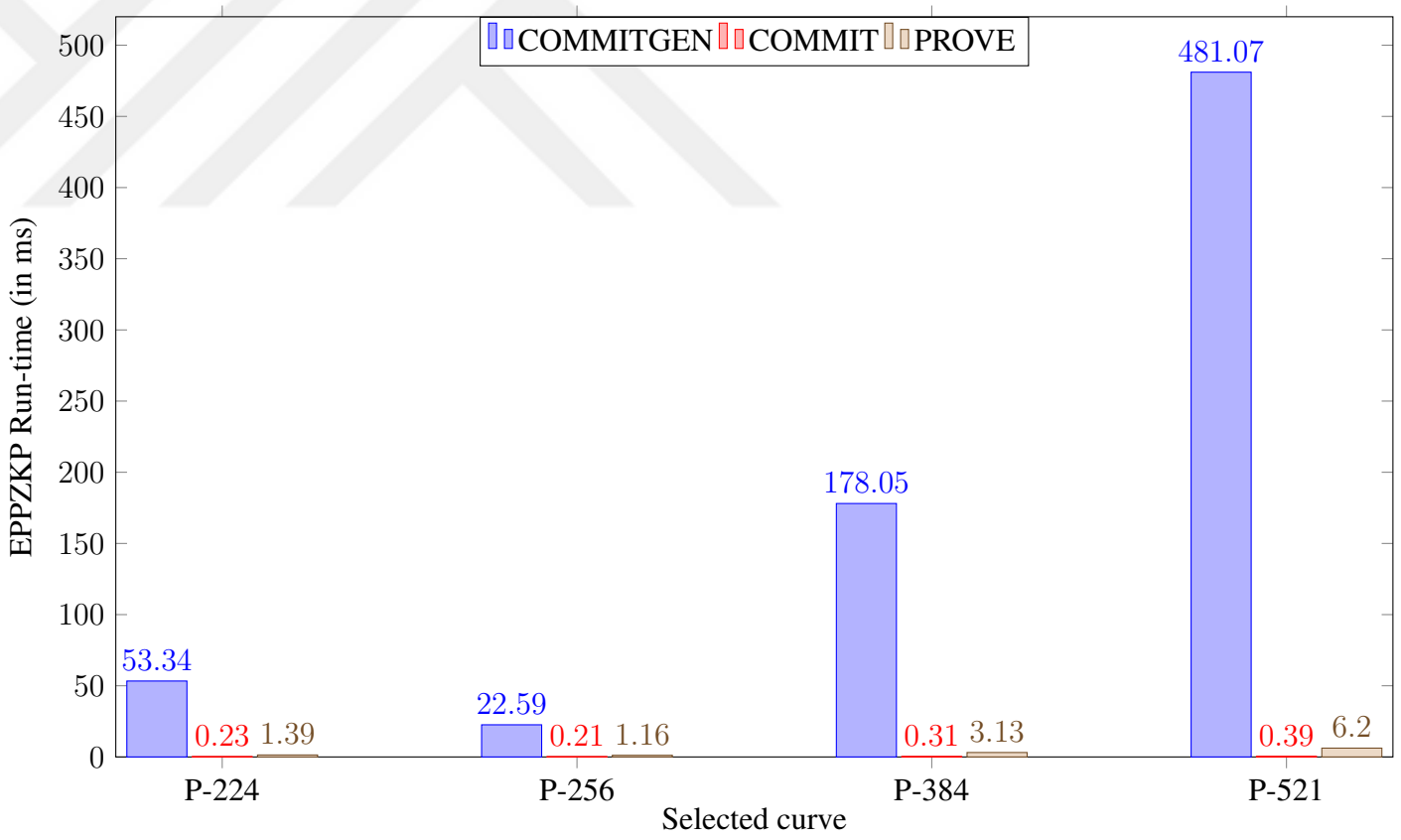


Figure 5.3: Run-time of EPPZKP using ECC.

CHAPTER 6

CONCLUSION

As one of the main contributions, this thesis proposes a novel (2, 3)-threshold group key-sharing mechanism, named PATS, which employs principles akin to secret-sharing schemes. Unlike traditional secret-sharing schemes, our approach involves the collaborative generation of shares by participating parties. A distinctive feature is the absence of prior knowledge about the secret used as the group key in this work. The parties engage in a mutual agreement procedure, generating pieces of information crucial for constructing a group key in a distributed manner over insecure channels. Importantly, parties remain unaware of each other's shares until the blending stage during key usage. Notably, PATS eliminates the need for a fully trusted third party. In summary, PATS, while resembling threshold secret-sharing schemes, demonstrates a distinct structure and approach. Unlike traditional schemes, PATS does not involve sharing prior confidential information, and it operates without a trusted setup. Unlike legacy secret sharing where a known secret is distributed to parties, PATS starts with an unknown secret (key). Parties collaboratively construct the secret (key) using shares obtained at the protocol's conclusion. This differs from the traditional approach where the secret is known in advance, and shares are constructed from it. PATS enhances security by avoiding the exposure of party shares in the event of a data breach. We propose a novel zero-knowledge (2,3)-threshold password-authenticated key exchange scheme, PATS, involving parties in group key sharing. This protocol leverages zero-knowledge proofs, providing advantages over well-established threshold key exchange schemes and accommodating semi-trusted parties. We introduced both centralized and distributed variants of PATS, extending it to a general (t, n)-threshold scheme. PATS supports distributed operation and enables key verification by a trusted third party participating in group key sharing.

As the second main contribution, we propose two zero-knowledge proofs. The first protocol, a Three-pass ZKP Based on the Equal Partition Problem (EPPZKP), is a Stern-type ZKP. The second protocol, Two Challenge ZKP Based on Equal Partition Problem (2EPPZKP), is inspired by the 5-pass identification scheme and reduces soundness error. Detailed security and performance analyses of EPPZKP and 2EPPZKP were also provided. Both protocols satisfy three properties, namely completeness, soundness, and zero-knowledge(ness) required for a protocol to be a ZKP and the formal explanation to prove these requirements is provided for each protocol. Finally, we present the implementation result of the EPPZKP protocol.

6.1 Future Work

The following approaches can be integrated into this research:

1. PATS has to be retrofitted with classical security tools to be applicable in practice. This includes the integration of nonces or timestamps to mitigate replay attacks, and using message authentication codes to safeguard against integrity attacks. We left them as future work since they are classical approaches. As another future work, we plan to develop techniques for reducing the messaging load in (t, n)-PATS.

2. PATS takes advantage of zero-knowledge proofs compared to well-known threshold key exchange schemes. The integration of EPPZKP and 2EPPZKP can be implemented to compare the protocols with other ZKPs.

3. Understanding the complexity of ZKPs requires extensive knowledge of arithmetic circuits and rank-1 constraint systems. Both protocols can be considered in such a way as to calculate the proof size, prover complexity, and verifier complexity to compare with the state-of-the-art ZKP schemes given in Table 2.1.

APPENDIX A

In this section, we present adopted notations and essential preliminary knowledge.

A.1 Notation

Security parameter is denoted by λ. Notation $b \xleftarrow{\mathrm{R}} B$ denotes that b is chosen uniform randomly from the finite set B. For integers $a \leq b$ the set of integers $\{a, a+1, \ldots, b\}$ is denoted $[a, b]$ as a shorthand. $Sym(n)$ denotes the set of all permutations on the set $\{1, \ldots, n\}$. Notations for a hash function and a commitment algorithm are H and Com respectively. For a permutation α, the value $\alpha(i)$ is also denoted by α_i.

Position bit-vectors and length-n tuples of group elements are denoted in bold lower-case letters $\mathbf{t}, \mathbf{t_r}, \mathbf{a}, \mathbf{b}, \mathbf{g}$. For a length-n tuple $\mathbf{a} = (a_1, \ldots, a_n)$, symbol $\mathbf{a}^\alpha$ denotes the image of $\mathbf{a}$ under the permutation α, that is $\mathbf{a}^\alpha = \alpha(\mathbf{a}) = (a_{\alpha_1}, \ldots, a_{\alpha_n})$. For an algorithm, the time complexity "probabilistic polynomial-time" is abbreviated as PPT.

A.2 Zero-knowledge Argument of Knowledge

Let X and Y be two finite sets and let $R \subseteq X \times Y$ be a relation defining a language L in NP. An element $w \in X$ is said to be a witness for a statement $y \in Y$ if $(w, y) \in R$. A zero knowledge argument of knowledge for L is a two party protocol between PPT interactive algorithms called prover (P) and verifier (V) satisfying three security conditions (completeness, proof of knowledge and zero knowledge). For a statement

y, while P gets a (w, y) pair as input, the other entity V only gets y. The sequence of exchanged messages at an interaction of P and V, on inputs s and t respectively, is called the transcript of the interaction. We will denote it by $View\langle P(s), V(t)\rangle$ and we will denote an execution that yields an output out by $out \leftarrow \langle P(s), V(t)\rangle$.

Definition A.2.1 *(Argument of Knowledge) A two-party protocol between PPT interactive algorithms (P, V) is said to be Argument of Knowledge with error ϵ if the following properties hold:*

- *Perfect Completeness: For any (w, y) pair in R,*

$$Pr[1 \leftarrow \langle P(w, y), V(y)\rangle] = 1.$$

- *Knowledge Soundness with error ϵ: Suppose P^* is a potentially dishonest PPT prover with success probability $\epsilon^* > \epsilon$. Then, there exists an expected polynomial time knowledge extractor algorithm E such that, given statement y and rewindable black-box oracle access to P^*, outputs a witness w for y with probability at least $\frac{\epsilon^* - \epsilon}{q(|y|)}$ for some positive polynomial q.*

In this thesis, we design a protocol on proof of knowledge of equal partition problem on groups. Within this framework, a set of group elements $S = \{g_1, \ldots, g_n\}$ is an instance and a position bit vector $\mathbf{t}$ determining the partition is a witness.

Argument of Knowledge protocols are specialized interactive proofs. Next we give the definition of k-special soundness defined for interactive proofs.

Definition A.2.2 *(adapted from [80]) Let $\pi = (P, V)$ be a 3-move public-coin (the verifier chooses its messages randomly and publish them) interactive proof for relation R with challenge set of cardinality $N \in \mathbf{N}$. The protocol π is said to be k-out-of-N special-sound for $k \leq N$, if there exists a polynomail time algorithm such that given a statement y and k-many accepting transcripts $(a, c_1, z_1), \ldots, (a, c_k, z_k)$ for distinct challanges but the same commitment, it outputs a witness w satisfying $(w, y) \in R$.*

It is well known that 2-special soundness implies knowledge soundness. As it is easier to check, 2-special soundness is commonly used to prove knowledge extraction. In [80] Theorem 1, this property generalized to k-special soundness.

Lemma A.2.1 *(adapted from [80] Theorem 1) A k-out-of-N special-sound 3-move public-coin interactive proof is knowledge sound with knowledge error $\epsilon = \frac{k-1}{N}$.*

Let us define zero-knowledge property:

Definition A.2.3 *(Zero-Knowledge) An argument of knowledge is zero-knowledge if, for every possibly cheating PPT verifier V^*, there exists an efficient simulator (a PPT algorithm) S such that, on input y, it outputs a transcript $S(y)$ which is indistinguishable from $View\langle P(w,y), V^*(y)\rangle$. The zero-knowledge property is called computational, statistical, or perfect zero-knowledge depending on whether the two distributions are computationally indistinguishable, statistically indistinguishable, or equal, respectively.*

An additional condition that the verifier behaves honestly gives Honest Verifier Zero-Knowledge.

Definition A.2.4 *(Honest Verifier Zero-Knowledge) An argument of knowledge is (computationally/statistically/perfectly) honest verifier zero-knowledge if there exists an efficient simulator (a PPT algorithm) S such that, on input y, the distributions of the simulated transcript output by S and the transcript $View\langle P(w,y), V(y)\rangle$, for honest verifier V, are (computationally/statistically/perfectly) indistinguishable.*

Definition A.2.5 *(Black-Box Zero-Knowledge) (adapted from [81]) An argument of knowledge is black-box zero-knowledge if there exists an efficient PPT oracle machine S such that for every non-uniform possibly cheating PPT verifier V^*, upon black-box access to $V^*(y)$, it outputs a transcript which is indistinguishable from $View\langle P(w,y), V^*(y)\rangle$.*

A.3 Commitment Schemes

Definition A.3.1 *A non-interactive commitment scheme Com is a sequence $(Setup, Commit, Ver)$ of algorithms such that:*

- $pp \leftarrow Setup(1^\lambda)$ *On input security parameter λ, algorithm $Setup$ outputs the public parameters pp.*
- $(c, r) \leftarrow Commit(pp, m)$ *On input public parameters pp and a message m, algorithm $Commit$ outputs a commitment c and some associated randomness r used for computing and opening (verifying) c. it specify the the message space M, randomness space R, and commitment space C.*
- $b \leftarrow Ver(pp, m, c, r)$ *On input the public parameters pp, a message m, a commitment c and the randomness r, the algorithm Ver outputs a bit $b \in \{0, 1\}$.*

A commitment scheme is required to possess three essential properties: correctness, binding and hiding. Correctness implies that if the scheme run honestly, the verification algorithm will output 1. Binding property ensures that it is practically impossible to reveal a commitment to two different messages, thereby preventing any manipulation or deceit. Hiding, on the other hand, ensures that the commitment does not disclose any information about the actual message it represents, maintaining the confidentiality and privacy of the committed data.

Formally,

Definition A.3.2 *A commitment scheme Com is correct if*

$$Pr\left[1 \leftarrow Ver(pp, m, c, r) \quad | \quad \begin{array}{c} pp \leftarrow Setup(1^\lambda) \\ (c, r) \leftarrow Commit(pp, m) \end{array}\right] = 1 \tag{A.1}$$

Definition A.3.3 *A commitment scheme Com is statistically hiding if for any adver-*

sary A there exists a negligible function ϵ such that

$$Pr\left[b' = b \quad | \quad \begin{array}{c} pp \leftarrow Setup(1^\lambda) \\ (m_o, m_1) \leftarrow A(pp) \\ b \xleftarrow{R} \{0,1\} \\ (c, r) \leftarrow Commit(pp, m_b) \\ b' \leftarrow A(c) \end{array}\right] \leq \frac{1}{2} + \epsilon(\lambda) \tag{A.2}$$

Definition A.3.4 *A commitment scheme Com is computationally binding if for any stateful PPT adversary A there exists a negligible function ϵ such that*

$$Pr\left[\begin{array}{c} 1 \leftarrow Ver(pp, m_0, c, r_0) \\ 1 \leftarrow Ver(pp, m_1, c, r_1) \\ m_0 \neq m_1 \end{array} \quad | \quad \begin{array}{c} pp \leftarrow Setup(1^\lambda) \\ (m_o, r_0, m_1, r_1, c) \leftarrow A(pp) \\ Commit(pp, m_0) = Commit(pp, m_1) \\ b' \leftarrow A(c) \end{array}\right] \leq \epsilon(\lambda) \tag{A.3}$$

REFERENCES

[1] A. Mohr, "A survey of zero-knowledge proofs with applications to cryptography," *Southern Illinois University, Carbondale*, pp. 1–12, 2007.

[2] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems," *Journal of the ACM (JACM)*, vol. 38, no. 3, pp. 690–728, 1991.

[3] O. Goldreich, "Zero-knowledge twenty years after its invention.," *IACR Cryptol. ePrint Arch.*, vol. 2002, p. 186, 2002.

[4] G. Yao, H. Wang, and D. Feng, "A group pake protocol using different passwords," in *2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*, vol. 1, pp. 270–273, 2009.

[5] X. Liu, J. Liu, and G. Chang, "npake: An improved group pake protocol," in *Proceedings of the 2012 Ninth Web Information Systems and Applications Conference*, pp. 115–118, 2012.

[6] M. Abdalla, O. Chevassut, P.-A. Fouque, and D. Pointcheval, "A simple threshold authenticated key exchange from short secrets," in *Proceedings of the Advances in Cryptology-ASIACRYPT 2005: 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005. Proceedings 11*, pp. 566–584, Springer, 2005.

[7] J. Camenisch, A. Lehmann, A. Lysyanskaya, and G. Neven, "Memento: How to reconstruct your secrets from a single password in a hostile environment," in *Proceedings of the Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II 34*, pp. 256–275, Springer, 2014.

[8] F. Hao, X. Yi, L. Chen, and S. F. Shahandashti, "The fairy-ring dance: Password authenticated key exchange in a group," in *Proceedings of the 1st ACM Work-*

shop on IoT Privacy, Trust, and Security, IoTPTS '15, (New York, NY, USA), p. 27–34, Association for Computing Machinery, 2015.

[9] K. Sakumoto, T. Shirai, and H. Hiwatari, "Public-key identification schemes based on multivariate quadratic polynomials," in *Proceedings of the Advances in Cryptology–CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings 31*, pp. 706–723, Springer, 2011.

[10] A. Kılıç, C. Betin Onur, and E. Onur, "Pats: Let parties have a say in threshold group key sharing," *IET Information Security*, vol. 2024, no. 1, p. 7557514, 2024.

[11] Q. Dai, X. Zhao, Q. Xu, and H. Jiang, "A new cross-realm group password-based authenticated key exchange protocol," in *Proceedings of the 2011 Seventh International Conference on Computational Intelligence and Security*, pp. 856–860, 2011.

[12] L. Zhu, C. Guo, Z. Zhang, W. Fu, and R. Xu, "A novel contributory cross-domain group password-based authenticated key exchange protocol with adaptive security," in *Proceedings of the 2017 IEEE Second International Conference on Data Science in Cyberspace (DSC)*, pp. 213–222, 2017.

[13] Y. Zhang, Y. Xiang, and X. Huang, "Password-authenticated group key exchange: A cross-layer design," *ACM Transactions on Internet Technology*, vol. 16, dec 2016.

[14] F. Wei, N. Kumar, D. He, and S.-S. Yeo, "A general compiler for password-authenticated group key exchange protocol in the standard model," *Discrete Applied Mathematics*, vol. 241, pp. 78–86, 2018.

[15] N. Asokan and P. Ginzboorg, "Key agreement in ad hoc networks," *Computer communications*, vol. 23, no. 17, pp. 1627–1637, 2000.

[16] E. Bresson, O. Chevassut, and D. Pointcheval, "Provably secure authenticated group diffie-hellman key exchange," *ACM Transactions on Information and System Security (TISSEC)*, vol. 10, no. 3, pp. 10–es, 2007.

[17] G. R. Blakley, "Safeguarding cryptographic keys," in *Proceedings of the Managing Requirements Knowledge, International Workshop on*, pp. 313–313, IEEE Computer Society, 1979.

[18] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[19] C. Asmuth and J. Bloom, "A modular approach to key safeguarding," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 208–210, 1983.

[20] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable secret sharing and achieving simultaneity in the presence of faults," in *Proceedings of the 26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pp. 383–395, IEEE, 1985.

[21] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *Proceedings of the 28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pp. 427–438, IEEE, 1987.

[22] J. C. Benaloh, "Secret sharing homomorphisms: Keeping shares of a secret secret," in *Proceedings of the Conference on the Theory and Application of Cryptographic Techniques*, pp. 251–260, Springer, 1986.

[23] Y. Zhang, C. Xu, H. Li, K. Yang, N. Cheng, and X. Shen, "Protect: Efficient password-based threshold single-sign-on authentication for mobile users against perpetual leakage," *IEEE Transactions on Mobile Computing*, vol. 20, no. 6, pp. 2297–2312, 2020.

[24] J. Jiang, D. Wang, G. Zhang, and Z. Chen, "Quantum-resistant password-based threshold single-sign-on authentication with updatable server private key," in *European Symposium on Research in Computer Security*, pp. 295–316, Springer, 2022.

[25] A. Bagherzandi, S. Jarecki, N. Saxena, and Y. Lu, "Password-protected secret sharing," in *18th ACM conference on Computer and Communications Security*, pp. 433–444, 2011.

[26] J. Camenisch, A. Lysyanskaya, and G. Neven, "Practical yet universally composable two-server password-authenticated secret sharing," in *2012 ACM conference on Computer and Communications Security*, pp. 525–536, 2012.

[27] Z. Zhang, X. Li, Y. Wang, Y. Miao, X. Liu, J. Weng, and R. H. Deng, "TAGKA: Threshold Authenticated Group Key Agreement protocol against member disconnect for UANET," *IEEE Transactions on Vehicular Technology*, 2023.

[28] N. Bitansky, R. Canetti, A. Chiesa, S. Goldwasser, H. Lin, A. Rubinstein, and E. Tromer, "The hunting of the SNARK," *Journal of Cryptology*, vol. 30, no. 4, pp. 989–1066, 2017.

[29] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit, "Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 327–357, Springer, 2016.

[30] D. Hopwood, S. Bowe, T. Hornby, N. Wilcox, *et al.*, "Zcash protocol specification," *GitHub: San Francisco, CA, USA*, vol. 4, no. 220, p. 32, 2016.

[31] T. Chen, H. Lu, T. Kunpittaya, and A. Luo, "A review of zk-SNARKs," *arXiv preprint arXiv:2202.06877*, 2022.

[32] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," *Cryptology ePrint Archive*, 2018.

[33] E. Ben-Sasson, L. Goldberg, and D. Levit, "Stark friendly hash–survey and recommendation," *Cryptology ePrint Archive*, 2020.

[34] A. Canteaut, T. Beyne, I. Dinur, M. Eichlseder, G. Leander, G. Leurent, M. Naya-Plasencia, L. Perrin, Y. Sasaki, Y. Todo, *et al.*, "Report on the security of stark-friendly hash functions (version 2.0)," *https://inria.hal.science/hal-02883253/document*, 2020.

[35] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward, "Aurora: Transparent succinct arguments for r1cs," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 103–128, Springer, 2019.

[36] C. Lund, L. Fortnow, H. Karloff, and N. Nisan, "Algebraic methods for interactive proof systems," *Journal of the ACM (JACM)*, vol. 39, no. 4, pp. 859–868, 1992.

[37] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish, "Doubly-efficient zkSNARKs without trusted setup," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*, pp. 926–943, IEEE, 2018.

[38] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam, "Ligero: Lightweight sublinear arguments without a trusted setup," in *2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2087–2104, 2017.

[39] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*, pp. 315–334, IEEE, 2018.

[40] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song, "Libra: Succinct zero-knowledge proofs with optimal prover computation." Cryptology ePrint Archive, Paper 2019/317, 2019. `https://eprint.iacr.org/2019/317`.

[41] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "Delegating computation: interactive proofs for muggles," *Journal of the ACM*, vol. 62, no. 4, pp. 1–64, 2015.

[42] D. Benarroch, L. Brandão, M. Maller, and E. Tromer, "ZKProof Community Reference. version 0.3.." ZKProof, 2022. `https://docs.zkproof.org/reference`.

[43] J. Groth, "On the size of pairing-based non-interactive arguments," in *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pp. 305–326, Springer, 2016.

[44] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity." Cryptology ePrint Archive, Paper 2018/046, 2018. `https://eprint.iacr.org/2018/046`.

[45] J. Stern, "A new identification scheme based on syndrome decoding," in *Proceedings of the Annual International Cryptology Conference*, pp. 13–21, Springer, 1993.

[46] P. Véron, "Improved identification schemes based on error-correcting codes," *Applicable Algebra in Engineering, Communication and Computing*, vol. 8, pp. 57–69, 1997.

[47] P. Gaborit and M. Girault, "Lightweight code-based identification and signature," in *Proceedings of the 2007 IEEE International Symposium on Information Theory*, pp. 191–195, IEEE, 2007.

[48] P.-L. Cayrel, P. Véron, and S. M. El Yousfi Alaoui, "A zero-knowledge identification scheme based on the q-ary syndrome decoding problem," in *Proceedings of the International Workshop on Selected Areas in Cryptography*, pp. 171–186, Springer, 2010.

[49] C. Aguilar, P. Gaborit, and J. Schrek, "A new zero-knowledge code based identification scheme with reduced communication," in *2011 IEEE Information Theory Workshop*, pp. 648–652, IEEE, 2011.

[50] Y. Song, X. Huang, Y. Mu, W. Wu, and H. Wang, "A code-based signature scheme from the lyubashevsky framework," *Theoretical Computer Science*, vol. 835, pp. 15–30, 2020.

[51] M. Baldi, M. Battaglioni, F. Chiaraluce, A.-L. Horlemann-Trautmann, E. Persichetti, P. Santini, and V. Weger, "A new path to code-based signatures via identification schemes with restricted errors," *https://arxiv.org/abs/2008.06403*, 2020.

[52] D. Kales and G. Zaverucha, "An attack on some signature schemes constructed from five-pass identification schemes," in *Proceedings of the International Conference on Cryptology and Network Security*, pp. 3–22, Springer, 2020.

[53] S. Bettaieb, L. Bidoux, O. Blazy, and P. Gaborit, "Zero-knowledge reparation of the Véron and AGS code-based identification schemes," in *Proceedings of the 2021 IEEE International Symposium on Information Theory (ISIT)*, pp. 55–60, IEEE, 2021.

[54] M. Baldi, S. Bitzer, A. Pavoni, P. Santini, A. Wachter-Zeh, and V. Weger, "Zero knowledge protocols and signatures from the restricted syndrome decoding problem," in *Proceedings of the IACR International Conference on Public-Key Cryptography*, pp. 243–274, Springer, 2024.

[55] J. Blocki, "Senior research thesis: Direct zero-knowledge proofs," 2009.

[56] M. Weir, S. Aggarwal, M. Collins, and H. Stern, "Testing metrics for password creation policies by attacking large sets of revealed passwords," in *17th ACM Conference on Computer and Communications Security*, pp. 162–175, 2010.

[57] F. Hao and P. Ryan, "J-PAKE: authenticated key exchange without PKI," in *Transactions on Computational Science XI*, pp. 192–206, Springer, 2010.

[58] F. Hao and P. C. van Oorschot, "SoK: Password-authenticated key exchange–theory, practice, standardization and real-world lessons," *https://arxiv.org/abs/2008.06403*, 2021.

[59] M. Abdalla, M. Barbosa, P. B. Rønne, P. Y. Ryan, and P. Šala, "Security characterization of J-PAKE and its variants." Cryptology ePrint Archive, Report 2021/824, 2021. `https://ia.cr/2021/824`.

[60] F. Hao, "J-PAKE: Password-authenticated key exchange by juggling," *RFC 8236*, 2017.

[61] F. Hao, "Prudent practices in security standardization," *IEEE Communications Standards Magazine*, vol. 5, no. 3, pp. 40–47, 2021.

[62] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipf's law in passwords," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2776–2791, 2017.

[63] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, "Targeted online password guessing: An underestimated threat," in *2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1242–1254, 2016.

[64] F. Hao, "Schnorr non-interactive zero-knowledge proof," *RFC*, vol. 8235, pp. 1–13, 2017.

[65] E. B. Barker, D. Johnson, and M. E. Smid, "SP 800-56A. recommendation for pair-wise key establishment schemes using discrete logarithm cryptography (revised)." https://csrc.nist.gov/CSRC/media/Publications/sp/800-56a/rev-3/draft/documents/sp800-56ar3-draft.pdf, 2007.

[66] M. R. Garey and D. S. Johnson, *Computers and Intractability*, vol. 174. Freeman San Francisco, 1979.

[67] N. Karmarkar and R. M. Karp, *The differencing method of set partitioning*. Computer Science Division (EECS), University of California Berkeley, 1982.

[68] M. Toorani, "Cryptanalysis of two pake protocols for body area networks and smart environments.," *International Journal of Network Security*, vol. 17, no. 5, pp. 629–636, 2015.

[69] M. Toorani, "Security analysis of J-PAKE," in *Proceedings of the 2014 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, IEEE, 2014.

[70] R. Lu and Z. Cao, "Simple three-party key exchange protocol," *Computers & Security*, vol. 26, no. 1, pp. 94–97, 2007.

[71] D. R. Stinson, *Cryptography: Theory and Practice*. Chapman and Hall/CRC, 2005.

[72] D. Boneh, "The decision Diffie-Hellman problem," in *Proceedings of the International algorithmic number theory symposium*, pp. 48–63, Springer, 1998.

[73] A. Stiglic, *Safe Prime*, pp. 541–541. Boston, MA: Springer US, 2005.

[74] S. Iftene, "Secret sharing schemes with applications in security protocols.," *Sci. Ann. Cuza Univ.*, vol. 16, pp. 63–96, 2006.

[75] C. F. Kerry and P. D. Gallagher, "Digital signature standard (dss)," *FIPS PUB*, pp. 186–4, 2013.

[76] E. Barker and Q. Dang, "Nist special publication 800-57 part 1, revision 4," *NIST, Tech. Rep*, vol. 16, 2016.

[77] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 186–208, 1989.

[78] G. Oded, *Foundations of cryptography basic tools*. Cambridge University Press, 2001.

[79] "The go programming language." `https://go.dev/`. Accessed: 2024-01-22.

[80] T. Attema, R. Cramer, and L. Kohl, "A compressed σ-protocol theory for lattices," in *Proceedings of the Annual International Cryptology Conference*, pp. 549–579, Springer, 2021.

[81] O. Goldreich and Y. Oren, "Definitions and properties of zero-knowledge proof systems," *Journal of Cryptology*, vol. 7, no. 1, pp. 1–32, 1994.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Kılıç, Adnan
Nationality: Turkish (TC)

EDUCATION

Degree	Institution	Year of Graduation
M.Sc.	Middle East Technical University (CEng)	2017
Double Major	Middle East Technical University (CEng)	2013
B.S.	Middle East Technical University (Mathematics)	2013

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2022-Present	Aselsan	Cyber Security Architect
2021-2022	Somera	Senior DevOps Engineer
2014-2021	Middle East Technical University	System Administrator
2013-2014	Labris	Network Security Engineer

PUBLICATIONS

[1] Kılıç, Adnan, Ertan Onur, and Cansu Betin Onur. "Revisiting Shamir's No-Key Protocol: Lightweight Key Transport." 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/Cy-

berSciTech). DOI: 10.1109/DASC-PICom-DataCom-CyberSciTec.2017.103. IEEE, 2017.

[2] Yıldız, Hüsnü, Adnan Kılıç, and Ertan Onur. "Lightweight Cryptography in 5G Machine-Type Communication." Networks of the Future. Chapman and Hall/CRC, 2017. DOI: 10.1201/9781315155517-7, 127-144.

[3] Kılıç, Adnan, Cansu Betin Onur, and Ertan Onur. "PATS: Let Parties Have a Say in Threshold Group Key Sharing." IET Information Security 2024.1 (2024): 7557514. DOI: 10.1049/2024/7557514