ON PASSWORD-BASED AUTHENTICATED KEY EXCHANGE (PAKE) PROTOCOLS

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

BY

MERYEM TONGA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CRYPTOGRAPHY

JUNE 2020

Approval of the thesis:

ON PASSWORD-BASED AUTHENTICATED KEY EXCHANGE (PAKE) PROTOCOLS

submitted by **MERYEM TONGA** in partial fulfillment of the requirements for the degree of **Master of Science in Cryptography Department, Middle East Technical University** by,

Prof. Dr. Ömür Uğur _____
Director, Graduate School of **Applied Mathematics**

Prof. Dr. Ferruh Özbudak _____
Head of Department, **Cryptography**

Assoc. Prof. Dr. Ali Doğanaksoy _____
Supervisor, **Mathematics, METU**

Dr. İsa Sertkaya _____
Co-supervisor, **TÜBİTAK BİLGEM**

Examining Committee Members:

Assoc. Prof. Dr. Murat Cenk
Cryptography, METU _____

Assoc. Prof. Dr. Ali Doğanaksoy
Department of Mathematics, METU _____

Assoc. Prof. Dr. Zülfükar Saygı
Department of Mathematics, TOBB ETU _____

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

Assoc. Prof. Dr. Oğuz Yayla
Department of Mathematics, Hacettepe University _____

Date: _____

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

Name, Last Name: MERYEM TONGA

Signature :

ABSTRACT

ON PASSWORD-BASED AUTHENTICATED KEY EXCHANGE (PAKE) PROTOCOLS

Tonga, Meryem
M.S., Department of Cryptography
Supervisor : Assoc. Prof. Dr. Ali Doğanaksoy
Co-Supervisor : Dr. İsa Sertkaya

June 2020, 78 pages

Authentication and key agreement protocols play an important role in today's digital world. Key agreement methods mostly mimic Diffie-Hellman key exchange protocol, but unfortunately they are susceptible to man-in-the-middle attacks. Password based authenticated key exchange (PAKE) protocols promise to handle these key agreement and authentication without requiring existence of certificate authorities or trusted third parties. More importantly, PAKE protocols enable agreement on low-entropy passwords rather than high-entropy cryptographic keys shared by only involved parties. Even if PAKE protocols are not widely used in practice, they are already included in IEFT (RFC), ISO security standards and TLS cryptographic suite. In this thesis, by following these recent developments, we first present these PAKE protocols in three forms, namely balanced PAKE protocols, augmented PAKE protocols and password authenticated key retrieval (PAKR) protocols and within both single and multi server settings. Particularly, we revisit EKE, SPEKE, PAK, PPK, J-PAKE, SPAKE, SESPAKE balanced, and SRP, AugPAKE, OPAQUE, B-SPEKE augmented protocols. Then, we summarize security attacks to these protocols. Afterwards, detailed explanations of the attacks against these protocols are given. We further present current state of the art for PAKE protocols. Finally, we draw attention to possible extensions for PAKE protocols and state currently open questions about the subject.

Keywords: Cryptography, Cryptographic protocols, Password-based Authenticated Key Exchange (PAKE) Protocols, Augmented PAKE, Balanced PAKE, PAKE Protocols analysis

ÖZ

PAROLA TABANLI KİMLİĞİ DOĞRULANMIŞ ANAHTAR DEĞİŞİM PROTOKOLLERİ ÜZERİNE

Tonga, Meryem
Yüksek Lisans, Kriptografi Bölümü
Tez Yöneticisi : Doç. Dr. Ali Doğanaksoy
Ortak Tez Yöneticisi : Dr. İsa Sertkaya

Haziran 2020, 78 sayfa

Kimlik doğrulama ve anahtar anlaşması protokolleri günümüz dijital dünyasında önemli bir rol oynamaktadır. Anahtar anlaşması yöntemleri çoğunlukla Diffie-Hellman anahtar değişim protokolünü taklit ederler fakat maalesef aradaki korsan saldırılarına duyarlıdırlar. Parola tabanlı kimliği doğrulanmış anahtar değişim (PAKE) protokolleri, bu anahtar anlaşması ve kimlik doğrulamasının üstesinden gelmeyi sertifika yetkililerinin veya güvenilir üçüncü tarafların varlığına ihtiyaç duymadan vaadeder. Daha da önemlisi PAKE protokolleri, sadece ilgili tarafların paylaştığı yüksek entropi kriptografik anahtarlardan ziyade düşük entropi şifreler üzerine anlaşmayı mümkün kılar. Uygulamada PAKE protokolleri yaygın bir şekilde kullanılmasa da halihazırda IEFT(RFC), ISO güvenlik standartları ve TLS kriptografik paketinde yer almaktadır. Bu tez çalışmasında güncel gelişmeleri takip ederek ilk olarak PAKE protokollerini; dengeli PAKE protokolleri, genişletilmiş PAKE protokolleri ve parola doğrulamalı anahtar geri alımı (PAKR) protokolleri olarak üç şekilde, tekli ve çoklu sunucu ortamlarında takdim ediyoruz. Özellikle dengeli protokollerden EKE, SPEKE, PAK, PPK, J-PAKE, SPAKE ve SESPAKE'nin; genişletilmiş protokollerden SRP, AugPAKE, OPAQUE ve B-SPEKE'nin üzerinden geçiyoruz. Sonra bu protokollere yapılan güvenlik saldırılarını özetliyoruz. Daha sonra bu protokollere karşı yapılan saldırıların detaylı açıklamaları verilir. Ayrıca PAKE protokollerinin güncel gelişmiş halini sunuyoruz. Son olarak PAKE protokolleri için olası eklemelere dikkat çekiyor

ve konuyla ilgili çözümlenmemiş soruları belirtiyoruz.

Anahtar Kelimeler: Kriptografi, Kriptografik protokoller, Parola Tabanlı Kimliği Doğrulanmış Anahtar Değişim (PAKE) Protokolleri, Genişletilmiş PAKE, Dengeli PAKE, PAKE Protokolleri Analizi

To my family

ACKNOWLEDGMENTS

I would like to thank my supervisor Assoc. Prof. Dr. Ali Doğanaksoy for his motivation and support. I am grateful to him for making this process easier. I also thank my co-supervisor Dr. İsa Sertkaya for his motivation, help and support. Thanks to his guidance and experience, I was able improve my thesis.

I also would like to thank Assoc. Prof. Dr. Murat Cenk, Assoc. Prof. Dr. Zülfükar Saygı, Assoc. Prof. Dr. Oğuz Yayla and Assoc. Prof. Dr. Fatih Sulak for their advices.

I would like to express my special thanks to my husband Muhammed Tonga for his endless support and patience. He undertook correction of my thesis and helped me in every moment of this process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

A-EKE	Augmented Encrypted Key Exchange
AKE	Authenticated Key Exchange
aPAKE	Augmented (Asymmetric) Password Authenticated Key Exchange
AMP	Authentication via Memorable Password
AugPAKE	Efficient Augmented Password-Only Authentication Key Exchange
B-SPEKE	'B' Extension of SPEKE Protocol
CA	certificate Authority
CDH	Computational Diffie-Hellman Key Exchange
CPSA	Cryptographic Protocol Shapes Analyzer
DDH	Decisional Diffie-Hellman Key Exchange
DH	Diffie-Hellman Key Exchange
DH-OPRF	Diffie-Hellman Oblivious Pseudo Random Function
ECC	Elliptic Curve Cryptography
EKE	Encrypted Key Exchange
J-PAKE	Password Authenticated Key Exchange by Juggling
KDF	Key Derivation Function
KE	Key Exchange
MAC	Message Authentication Code
OPRF	Oblivious Pseudo Random Function
PAK	Password Authenticated Key exchange
PAKE	Password Authenticated Key Exchange
PAKR	Password-Authenticated Key Retrieval
PIN	Personal Identification Numbers
PKI	Public Key Infrastructure
PPK	Password Protected Key exchange
PRF	Pseudo Random Function
RA	Registration Authority

RLWE	Ring-Learning-with-Errors
RSA	Rivest, Shamir and Addlemen
SESPAKE	Security Evaluated Standardized Password Authenticated Key Exchange
SNP	Schnorr NIZK(non-interactive zero-knowledge) Proof
SPAKE	Simple Password-Based Encrypted Key Exchange
SPEKE	Simple Password Exponential Key Exchange
S-PCCDH	Set Password Based Chosen-basis Computational Diffie-Hellman
SRP	Secure Remote Password
TLS	Transport Layer Security
TTP	Trusted Third Party
UKS	Unknown Key-Share
ZKP	Zero Knowledge Proof
3PAKE	Three-party PAKE protocol

CHAPTER 1

INTRODUCTION

In a network communication, parties try to exchange data or information over a secure channel and this is done by encrypting messages. Hence, involved parties need to agree on a common key to encrypt and decrypt these messages. *"The key exchange"*, also known as *key establishment*, is a cryptographic mechanism where the communicating parties exchange cryptographic keys by using symmetric or public key cryptography.

In many key exchange systems, one of the parties produces the key and sends it to other who does not have any effect on it. *"The key agreement"* is a form of key exchange where each of communicating parties contribute to computation of the shared key value.

First key agreement protocol, which is commonly called Diffie-Hellman (DH) key exchange, was proposed in paper [20], It provides two parties with establishment of a secret key over an insecure channel without giving information about each other. After that, consequent communications are encrypted by using this key along with a symmetric key cryptography.

DH key exchange is utilized in many authentication protocols as a basis and also in Transport Layer Security's (TLS) short-lived modes to provide perfect forward secrecy [42]. However, it does not provide authentication itself; thus, it can not prevent a man-in-the-middle attack [39]. An attacker who applies the man-in-the-middle attack can eavesdrop and re-encrypt messages in order to send them to again. Therefore, the parties should authenticate each other in order to prevent this kind of attack. For

details of DH and the man-in-the-middle-attack, please see Section 2.3.1.

The general key exchange protocols need to satisfy "forward secrecy" and "security of session" which are explained below [11].

Forward secrecy: Forward secrecy, which is also named as perfect forward secrecy, provides a protection for past sessions in order not to reveal the secret keys or passwords even if the session's private keys are compromised afterwards.

Security of session key: Security of session key means that the session key established in a key exchange protocol must be known only by related parties.

The process of proving of identity of a user is called as authentication.. This process runs at the beginning of the application with two different phases which are identification and actual authentication. In the identification phase, the user's identity (user ID) is provided to the security system so that security system searches user logs in. In the authentication stages, the user's identity is checked whether it belongs to the claimed user. The authentication runs based on three factors:

- Something users know: password, PIN, challenge response parameter.
- Something users have: ID card, security token, smartcard.
- Something users are: fingerprint, voice, biometric.

In general, the authentication is maintained by a certificate authority (CA) trusted third party (TTP). A TTP is an entity of communities that all parties in the community rely to execute a desired service. In some services, TTP should protect and store a long-term secrets, but it may cause a security flaw when these secrets are compromised. This means that all past and future communications preserved by these secrets may reveal [4]. A certificate authority (CA) is a foundation or company who assures the identities of the communicating parties by attaching them to cryptographic keys via digital certificates. CA is an important part of PKI (see Chapter 2) and it acts as a TTP.

The traditional communication networks use the concept of client-server interaction where a user sends his password or hash of the password and the value is stored in

server's database. But it gives attacker an opportunity to make an offline dictionary search and tries possible passwords against the hash value of the true password [66].

The authenticated key exchange (AKE) aims to prevent such vulnerabilities and also provides authentication of the parties in order to protect information while communicating over insecure networks. The general AKE protocols use digital signatures and public key encryption which may need higher cost for particular applications [69]. Contrarily, the password-based authentication key exchanges (PAKE) are based on passwords since they are easier and more convenient to remember rather than high entropy secret keys so PAKEs do not require any gadgets like smart cards [69]. Also, they do not require CA or TTP existence.

PAKE protocols generally involves methods such as:

- **Balanced password-authenticated key exchange (Balanced PAKE)**: In balanced PAKE protocols, the server stores the password and this password is used to establish and authenticate a common secret key by the parties. However, these protocols can not protect password if the server is compromised. Examples of balanced PAKE protocols are listed below:
  - EKE [9]
  - SPEKE [34], [29]
  - PAK and PPK [13]
  - J-PAKE [31]
  - EC-SRP or SRP5 [72]
  - SPAKE1 and SPAKE2 [3]
  - SESPAKE [54]

  EKE and SPEKE are insecure protocols against cryptographic attacks and J-PAKE, SPAKE1, SPAKE2, SESPAKE, PAK and PPK are known as secure protocols.

  EC-SRP protocol, or called as SRP5, is an implementation of SRP protocol over elliptic curve cryptography (ECC) groups and it is also a secure PAKE protocol.

- **Augmented password-authenticated key exchange (Augmented PAKE)**: In Augmented PAKE protocols, the client generates a couple of keys which consists of his public and private key by using his password. The public key is generally the password's a hashed value and the server uses it to authenticate the client. Since the server has not any information about a plaintext password and does not store it, these protocols can provide more security rather than balanced PAKE protocols. But they are also more complex and computationally expensive to implement.

  Examples of Augmented PAKE protocols are:

  - SRP [63]
  - AugPAKE [52]
  - OPAQUE [37]
  - Augmented-EKE [10]
  - B-SPEKE [12]
  - AMP [12]

  The Augmented EKE protocol is a variant of EKE protocol that is proposed by Bellovin and Merrit [10]. Similary, the B-SPEKE and AMP protocols are two variants of Diffie-Hellman based on EKE protocol. For further information about B-SPEKE and AMP protocols, reader may see [12].

- **Password-authenticated key retrieval (PAKR) protocol**: It provides distribution of the password of the client in n servers while each server holds a partial share of private keys, such as the Ford and Kaliski methods. Ford and Kaliski [23] suggested the-first multi-server PAKE protocol and later it is named as Password-Authenticated Key Retrieval (PAKR). Hence, PAKR protocols can protect passwords and static keys from compromises of the server which means that the attacker can not verify a single guessed password and learn anything about private key, even if he captures management of up to $n - 1$ servers.

In general, the PAKE protocols, classified as above, applied in two different settings, namely the single-server setting and multi-server setting.

In the single-server setting, the client and server create a common secret key by using a key exchange protocol. This key is utilized key to mutually authenticate the parties and also derive keys for symmetric encryption. There are three models of PAKE which are stated in the paper [66]:

- The password-only model is the first model. Bellovin and Merritt [9] introduced the first example of this model which is called "Encrypted Key Exchange (EKE)" protocol. In EKE protocol, the communicating parties use a password as a secret key and use this key in a key exchange process.

- The PKI-based model is the second model presented by Gong et al. [25], and defined formally by [27]. In this model, the client stores the server's public key and shares a password with the server by public key encryption. In this way, the communication can resist offline dictionary attacks.

- The ID-based model is the third PAKE model that was first given by Yi et al. [67], [68]. The client is in need of recalling a password with identity of the server and the server stores the password with a private key which has information about its identity [66].

Multi-server PAKE protocols was proposed to be a solution to the compromise of server of single-server systems. In single-server PAKE protocols, all passwords are stored in a single server and this causes disclosure of all passwords when the server is compromised. In multi-server setting, the password is distributed between more than one server, so it can be protected when one of the server is compromised.

The multi-server PAKE protocols can be categorized as follows:

Threshold PAKE: Ford and Kaliski [23] introduced the first *PKI-based* threshold PAKE protocol in which the client's password is shared with n servers and they collaborate in order to generate independent session keys with the client and procure authentication of the client. This protocol maintains security provided that compromise of $t-1$ or fewer servers [66]. Furthermore, the *password-only* threshold PAKE protocol, which given by Di Raimondo and Gennaro [19], is in need of compromise of less than one third of the servers.

Two-server PAKE: Two-server *PKI-based* PAKE protocol was first suggested by Brainard [14] in which two servers collaborate for the client's authentication and the password is protected until compromise of one of the servers. In addition, Katz et al. [40] introduced a two-server *password-only* PAKE protocol where two servers symmetrically authenticate the client. Lately, an *ID-based* two server PAKE protocol is presented by Yi et al. [65].

The general two-party PAKE protocols require a collection of enormous numbers of passwords from every client and thus they limit scope of key exchange [16]. To address this problem, Steiner et al. [55] first suggested three-party PAKE protocol. In-three party PAKE protocol, there are a trusted server and two clients sharing a human-memorable password. For authentication of themselves, two clients consult to this trusted server when they wish to construct a session key.

The cryptographic attacks that are mentioned in this thesis are summarized below [11]:

Man-in-the-middle attack: The attacker manages the communication by impersonating two parties while they believe that they make a connection directly over a private connection.

Offline dictionary attack: The attacker intercepts the information by eavesdropping the communication and tries to guess the correct password in an offline manner. The offline dictionary attack can not be detected because it does not require an client-server participation and a limitation of number of guessed.

Online dictionary attack: There are two types of online dictionary attack which are called as "undetectable" and "detectable" online dictionary attack. In the undetectable online dictionary attack, the attacker verifies the guessed password in many times in an online manner, but the parties can not notice a failed attempt. In the detectable online dictionary attack, the attacker verifies the guessed password by using an answer of the server. Thus, a failed attempt of the attacker can be noticed by the parties.

Replay attack: In the replay attack, also called as playback attack, the attacker can repeat a previous message and/or delay messages.

Impersonation attack: The attacker impersonates one of the communicating parties by assuming the identity of that party and tries to cheat other party.

1.1 Related Works

The first PAKE protocol that is named as EKE was introduced by Bellovin and Merrit [9] in 1992 where the parties carry out an encrypted version of DH. Then, Bellovin and Rogaway [8] improve their idea and proposed the first formal security model for authenticated key exchange protocol . Later, M. Abdalla and D. Pointcheval [3] and V. Boyko, P. MacKenzie, and S. Patel [13] presented their works that have proof in the random oracle model.

PAKE protocols are simple and efficient to use in the communication areas since they are in need of only a human-memorable password. However, this property makes them vulnerable to online and offline dictionary attacks because passwords are generally selected from a relatively small dictionary. But online dictionary attack can be prevented by limiting count of login failures.

Besides two-party PAKE protocols, many researchers studied in three-party PAKE (3PAKE) protocols. The first proposal of 3PAKE protocol was given by Steiner et al.[55] in 1995. Subsequently, Ding and Horster [22] indicated that this protocol can not withstand undetectable online dictionary attack in 1995.

In 2000, Lin et al. showed that Steiner et al.'s protocol is also susceptible to offline dictionary attack and proposed improved protocol [43] in which public key of the server is used. Moreover, many researchers study on three-party PAKE protocols (e.g [11], [16], [43], [44]).

In addition, Ford and Kaliski [23] presented the first multi-server PAKE which is called as Password-Authenticated Key Retrieval (PAKR) in 2000. PAKR protocol in [23] confides in prior server-authenticated secure channel like SSL and TLS to provide a protection against offline dictionary attacks, but it can be susceptible to phishing attacks and web-spoofing attack [53]. For this problem, Jablon [36] suggested a PAKR protocol in 2001 where multiple servers do not require a previous server-authenticated secure channel.

M. Bellare, D. Pointcheval, and P. Rogaway presented many alternative, secure PAKE protocols. In this area, there have been works about variations and security proofs. These techniques' present standards can be summarized as follows [60]:

- IETF RFC 2945 includes SRP protocol
- RFC 5931 includes EAP Protocol
- RFC 6124 includes EAP Protocol
- RFC 6617 includes Secure Pre-Shared Key (PSK) Authentication method
- RFC 6628 includes AugPAKE protocol
- RFC 6631 includes Password Authenticated Connection Establishment (PACE)
- IEEE Std 1363.2-2008 includes PAK and SRP protocol
- ISO-IEC 11770-4:2006 includes J-PAKE protocol and AugPAKE protocol

X.Yi et al. [66] presented ID2S PAKE protocol in 2016. They introduced two compilers which provide conversion of any two-party PAKE protocol to a two-server PAKE protocol based on identity based cryptography. In ID2S PAKE, a password of a client is respectively divided among two servers so that these servers do not know the password and the client is authenticated by collaborating servers. Moreover, X.Yi et al. prove the security of ID2S PAKE protocol without using random oracles. Later, in 2016, Lin Zhang and Zhenfeng Zhang [70] presented an existing related-key attack for ID2S PAKE protocol. They showed that if one of the servers is compromised, it is possible to reproduce the key. Moreover, they offered a solution to avoid this threat.

In most of PAKE protocols, the server needs to store a password of the client to authenticate him but it may cause disclosure of the password if the server is compromised. By addressing this problem, in 2017, N. Shafnamol and K.Simi Krishna [50] introduced a threshold version of ID2S PAKE which is named as signature-based multi-server PAKE protocol. In [50], the client's password is divided among n distinctive servers and then m out of these servers collaborate to authenticate the client by using signatures.

Known PAKE protocols can not satify security in IoT setting because they can no resist to side-channel attacks. For this reason, O.Rua et al. [49] suggested the first leakage-resilient PAKE (LR-PAKE) protocol in 2017. Their protocol consists of DH key exchange, LR storage and LR refreshing of LRS and they also prove the security of the protocol in the standard model.

Mochetti, Resende and Aranha [46] proposed an augmented PAKE protocol which is called as zkPAKE. It is believed that zkPAKE is convenient especially for banking implementations in which the server needs to store only hashed value of a password. Then, [7] introduced an offline dictionary attack against zkPAKE in 2019 and showed that the protocol is not secureto be used a PAKE protocol.

In addition, in 2019, Hoang et al. [33] proposed Password-based Signcryption Key Exchange (PSKE) which supplies efficient computation to some proposed PAKE protocols, like EKE, SRP and J-PAKE, by satisfying all necessities of security. PSKE protocol can also be used to authenticate users in IoT setting.

1.2 Our Contributions

In recent years, PAKE protocols are popular topics in cryptography. They play an important role in network communication and many researchers have been working on them.

In this thesis, we gather these protocols and examine their latest statuses. To do so, we tabulate properties of some balanced and augmented PAKE protocols with security analyses of them. That is, we examine EKE, SPEKE, SPAKE, J-PAKE, PAK, SESPAKE protocols of which form is balanced PAKE and B-SPEKE, SRP, AugPAKE, QPAQUE protocols of which form is augmented PAKE. Also, we indicate proposed cryptographic attacks that are performed on EKE namely "number theoretic attack" and on SPEKE namely "exponential-equivalence attack", "impersonation attack" and "key-malleability attack". Moreover, we emphasized which protocols are included in standards and calculated computational cost of these protocols.

Therefore, we provide accessible, easy to follow and comparable compilation of PAKE protocols. This study can be considered as an extensive survey on PAKE pro-

tocols in recent years. Moreover, by giving algorithm and analysis of each protocol, detailed examination about PAKE protocols is achieved and presented.

1.3 Organization

The rest of this thesis is organized as follows:

In Chapter 2, we give some notations and cryptographic primitives which are used in the thesis. We give necessary definitions and then state Diffie-Hellman Key Exchange.

In Chapter 3, we introduce some PAKE protocols and their properties. The PAKE protocols given in this chapter are EKE, SPEKE, B-SPEKE, SPAKE, SRP, AugPAKE, J-PAKE, OPAQUE, PAK, PPK and SESPAKE.

Augmented EKE protocol is a variant of EKE protocol that is proposed by Bellovin and Merrit [10]. Similary, the B-SPEKE and AMP protocols are two variants of DH-EKE protocol. We explain EKE and DH-EKE protocols in Chapter 3, hence we do not include Augmented EKE protocol and AMP protocol. For further information about B-SPEKE and AMP protocols, reader may see [12].

Moreover, EC-SRP protocol, or called as SRP5, is an implementation of SRP protocol over elliptic curve cryptography (ECC) groups. We give details of SRP protocol in Chapter 3 and reader may get information about EC-SRP in [72].

In Chapter 4, we analyze security of PAKE protocols that are mentioned in Chapter 3. Moreover, we show how SRP and OPAQUE protocols can be used in TLS protocol. Then, we give an overview for the protocols and compare computational costs of them.

In Chapter 5, we present a summary of the thesis and discuss some possible future work.

CHAPTER 2

PRELIMINARIES

In this chapter, we first set the notations and give necessary definitions that are going to be needed in the following chapters. For further details, reader may refer to [39].

2.1 Notation

The list of common notations in this thesis is given below. The notations are special to the protocol stated at the beginning of the protocol description.

- p : a public large prime number. Unless otherwise stated, all computations are performed modulo p.
- q : a large prime factor of $p-1$
- g : a public primitive root of modulo p (often called as generator)
- $\mathbb{Z}_p$: a multiplicative group of integers modulo p
- G : a finite cyclic group with prime order
- x, G_x : a factor of $p-1$, a subgroup of $\mathbb{Z}_p^*$
- H : a one way hash function
- H' : a one way hash function mapping arbitrary string into G
- P : a secret password
- K : a session key or common secret key established in the protocol

In this thesis, the terms *client* and *server* are denoted as C and S. The client is generally an ordinary computer user who can remember only short passwords. The server is a service or a network program which accepts requests from the client. In some protocols, we use the classical characters *Alice*, *Bob* and *Eve* and instead of these terms we denote them as A, B, E respectively.

2.2 Cryptographic Primitives

2.2.1 Some Definitions

Hash function: A cryptographic hash function is a mathematical function that converts a numeric input value (message) to another numeric value (hash output). Hash function's input has any size while hash function's output is always constant sized. The main property of hash function is being a one-way function; that is, finding inverse of the function is computationally infeasible. It also deterministic, means that hash output of same message is always the same. Moreover, finding two distinct messages with the same hash output is not possible.

Cryptographic hash functions are quite useful and used in many information security applications like digital signatures, message authentication (MACs) and authentication applications.

Throughout the thesis, we denote the hash functions as H: $\{0,1\}^* \rightarrow \{0,1\}^{(O\lambda)}$, λ being the security parameter, which is a collision resistant hash function.

Key Derivation Function: A key derivation function (KDF) is a hash function that is used to obtain secret keys from a password, a master key or some other secret value. Furthermore, KDFs can be used to extend the length of a key to enhance security.

Digital signatures: A digital signature is a mathematical method used to verify authenticity of a digital message, software or document. They are used for financial transaction, software distribution and implementations of electronic signatures.

Digital signatures work on public key cryptography to supply messages', transmitted over a insecure channel, validation and security. Moreover, they provide non-repudiation property which means that user (signer) can not claim he did not sign a

message.

Public key infrastructure: Public key infrastructure (PKI) is a system of procedures, policies, hardware and software that provides security for public key cryptography. The main purpose of PKI is to enhance the number of e-services of private and government applications and ensure some cryptographic requirements like strong authentication, non-repudiation, data confidentiality and data integrity.

PKI involves two main functions that are certification and validation. In the certification function, the value of the public key is authentically bound an entity. In the validation function, the certifications are verified whether they are still valid or not.

A complete PKI consists of the following components [5]:

1. Certificate Authority (CA): A certificate is a data model which includes public key value with corresponding private key value. Every public key certificate has a digital signature that is related to individual CA. The lifetime of the certificate is one or two years and it should be abolished when the private key is compromised.

2. Registration Authority (RA): A registration authority is used to verify demands of a user for a digital certificate and submit to CA to issue it. RA has one super administrator who has the access authority for all functions of RA.

3. Certificate generation and Key: RA constitutes a key holder identity so that the owner of key or CA can transfer the private key to the key holder.

4. Signature generation: The public key certificate is signed with the signature data by the holder of key.

5. Certificate validation: The cancellation status of the certificate must be controlled by a trusted party.

Zero Knowledge Proof: A zero-knowledge proof (ZKP), or also called as zero-knowledge protocol, is a mathematical approach which allows verifying data without revealing any information of that data. That is, the prover can prove a value x to the verifier without giving anything else about the value x. The purpose of ZKP is

the prover can demonstrate that they can establish a secure communication without giving information about input data or computational process.

To illustrate ZKP, suppose B is red-green color-blind and A is not. A has two balls of different colors, say red and green. She tries to prove to B that they are different colored without telling which ball is which color. Firstly, she gives balls to B and he puts the balls behind his back. Then, he selects one of them and displays it to A. Next, he again puts it behind his back and selects one of the balls to show it to A. He asks A, the question that "Did I switch the ball?". A can say they were the same color or not.

The probability of identification of switch or non switch situation is 50% and if they repeat this processs multiple times (e.g 100), B will be really persuaded that the balls are in different colors.

A complete ZKP needs to satisfy three features:

- Completeness: If the information is true, the honest prover should convince the verifier of truth of the information.

- Soudness: If the information is false, the honest prover can not convince the verifier that the information is true.

- Zero-knowledge: The verifier should not learn anything about the prover's information.

Oblivious Pseudo Random Function (OPRF): Oblivious Pseudorandom Function is a protocol that consists of pseudorandom function family F which is defined between a S and a C. In the protocol, the S's input is a key k for PRF function F and and in domain of F, C's input is a value x. When the protocol is completed, the client learns $F(k; x)$ (where k and x are inputs) but does not learn anything about x while the S learns nothing from the protocol execution [37].

2.3 Computational Assumptions

The PAKE protocols that are mentioned in this thesis are based on security of cryptographic hash functions and discrete logarithm problem.

A cryptographic hash function is secure if it fulfils three basic properties: collision resistance, pre-image resistance and second pre-image resistance.

- Being pre-image resistant means that it is difficult to find any message m such that $\mathsf{H} = Hash(m)$ for a given a hash function H.In other words, a cryptographic hash function should be one-way function.
- Being second-pre image resistant implies that it is difficult to find a different message m_2 for a given message m_1 such that $\mathsf{H}(m_1) = \mathsf{H}(m_2)$.
- Finally, being collision resistant means that it is difficult to find two different messages m_1 and m_2 satisfying $\mathsf{H}(m_1) = \mathsf{H}(m_2)$.

The discrete logarithm problem may be defined as follows problem's definition can be as follows: Suppose that g is a group and $< g >$ is the cyclic group of G generated by g. For a given $g \in G$ and $a \in< g >$, the problem aims to find an integer x which satisfies $g^x = a$.

The discrete logarithm problem's hardness comes from choice of groups. In cryptographic systems, $\mathbb{Z}_p^*$ is chosen as a group where p is a prime number, but if $p - 1$ is product of small prime numbers, the discrete logarithm problem can be solved by using the Pohlig-Hellman algorithm.

2.3.1 Diffie-Hellman Key Exchange

Diffie-Hellman (DH) key exchange is a cryptographic protocol that enables establishing a shared secret key between two parties and used in order to exchange messages over an insecure public channel. It was first introduced by Whitfield Diffie and Martin Hellman [20] and it is known as non-authenticated key agreement protocol but is used in TLS ephemeral modes to supply forward secrecy and a base for authenticated protocols.

As in Figure 2.1, let p and g are publicly known values where p is a large prime number and g is a generator of a cylic group G in modulo p. At the beginning, A selects a private random number $R_A \in [0, p-1]$ and calculates her public key $A = g^{R_A} \bmod p$. Likewise, B selects a private random number $R_B \in [0, p-1]$ and calculates his public key $B = g^{R_B} \bmod p$. After sending the public keys to each other, A calculates the key $K = (g^{R_B})^{R_A} \bmod p$ and B calculates the key $K = (g^{R_A})^{R_B} \bmod p$.

A		B
secret: R_A, K		secret: R_B, K
$A = g^{R_A} \bmod p$	$\xrightarrow{g,p,A}$	$B = g^{R_B} \bmod p$
$K = B^{R_A} \bmod p$	$\xleftarrow{B}$	$K = A^{R_B} \bmod p$
$(K = (g^{R_B})^{R_A} \bmod p)$		$(K = (g^{R_A})^{R_B} \bmod p)$

Figure 2.1: Diffie-Hellman Key Exchange

The security of Diffie-Hellman key exchange arises from difficulty of solving discrete logarithms if a finite cyclic group G and the g are chosen appropriately. However, since Diffie-Hellman algorithm does not include authentication of the parties, it has a vulnerability to man-in-the-middle attack.

To explain the attack, assume there is an attacker E who applies man-in-the-middle attack:

- A selects a private key R_A and sends her public key $A = g^{R_A} \bmod p$.
- E intercepts A's message A. She picks a private random number R_C and calculates $C = g^{R_B} \bmod p$ then sends her public key C to B with identity of A.
- B gets A's E identity with public key C. Thus, B stores A's client identity with E's public key C.
- E sends the public key C to A.
- A calculates a secret key $K_1 = C^{R_A} \bmod p$.

- E calculates $K_1 = A^{R_C} \bmod p$ as well. Therefore, the key K_1 is shared by A and E.
- B calculates a secret key $K_2 = C^{R_B} \bmod p$.
- E calculates $K_2 = B^{R_C} \bmod p$ as well. Therefore, the key K_2 is shared by B and E.
- Finally, A thinks that she shared the key K_1 with B and similarly B also presumes that he shared the key K_2 with A. But there is no common secret key between them and E can start to manage messages shared between them.

A		E		B
$A = g^{R_A} \bmod p$	$\xrightarrow{A, identity_A}$	$C = g^{R_C} \bmod p$	$\xrightarrow{C, identity_A}$	store $C, identity_A$
			$\xleftarrow{B}$	$B = g^{R_B} \bmod p$
$K_1 = C^{R_A} \bmod p$	$\xleftarrow{C}$	$K_1 = A^{R_C} \bmod p$		
		$K_2 = B^{R_C} \bmod p$		$K_2 = C^{R_B} \bmod p$

Figure 2.2: Man-in-the-middle attack on DH

Figure 2.2 shows man-in-the-middle attack on DH. Moreover, DH is sensitive to the replay attack and the impersonation attack [39].

CHAPTER 3

PAKE PROTOCOLS

In this chapter, we introduce some PAKE Protocols and their properties.

3.1 EKE : Encrypted Key Exchange Protocol

3.1.1 EKE

Encrypted Key Exhange (EKE) protocol is the first PAKE protocol introduced by Bellovin and Merrits [9]. It is a balanced PAKE protocol which is merging of symmetric key and public key cryptography. Two parties use a mutual password to exchange secret messages which is authenticated with the help of this protocol.

Bellovin and Merrits [9] claims that EKE protects password from offline dictionary attacks. Here, we show general algorithm of EKE and variants of EKE which are RSA-EKE, Diffie-Hellman EKE and ElGamal EKE. Then, we describe attacks on them.

Firstly, we introduce classical key agreement:

- A and B share a secret which is called password P.
- A generates a random key K and computes $\mathsf{P}(K)$. Then, she sends the result $\mathsf{P}(K)$ to B.
- B decrypt $\mathsf{P}(K)$ and get K, then he sends $K(challenge)$ to A.

Here $\mathsf{P}(K)$ a message such that the key K is encrypted by P by using a symmetric-key algorithm which the parties agreed on.

Note that, if there is an eavesdropper who can record these messages, he can apply a dictionary attack on $\mathsf{P}(K)$ with candidate password P' and compute $K' = \mathsf{P}'^{-1}(\mathsf{P}(K))$ in order to obtain $K(challenge)$. Therefore, this classical negotiation is sensitive to offline dictionary attack and replay attack. In addition, it shows that all classical two-party key agreement protocols are impressible to these attacks.

EKE uses public keys to strengthen the classical key agreement. The extended version of protocol is explained below:

- A generates a public key E_A and a private key D_A randomly. Then, she encrypts E_A in a symmetric cryptosystem by using P, generates $\mathsf{P}(E_A)$ and sends the result to B.
- Since they shared password P at the beginning of the protocol, B can obtain E_A from $\mathsf{P}^{-1}(\mathsf{P}(E_A)) = E_A$. After that, he randomly generates a secret key K and it is encrypted in the symmetric cryptosystem with public key E_A so that he can produce $E_A(K)$. Subsequently, he encrypts $E_A(K)$ with password P and sends the result $\mathsf{P}(E_A(K))$ to A.
- A knows P and private key D_A, she uses them to obtain K from the equation $D_A(\mathsf{P}^{-1}(\mathsf{P}(E_A(K)))) = K$.
- Now, A and B both know public key E_A and secret key K, B sends $R(challenge)$ to A.

However, if there is an eavesdropper who can acquire the values $\mathsf{P}(E_A)$, $\mathsf{P}(E_A(K))$ and $K(challenge)$, he can decrypt $\mathsf{P}(E_A)$ by using a candidate password P' and derive a candidate public key E'_A from $E'_A = \mathsf{P}'^{-1}(\mathsf{P}(E_A))$. However, deciding whether E'_A is the correct public key makes sense provided that a secret key K' is present and $E'_A(K') = E_A(K)$ and $K'^{-1}(K(challenge))$ are satisfied. Consider that E_A and K are randomly selected from wide key spaces, so even if the space of password is small, such attacks are expensive to apply [9].

The basic algorithms that are mentioned above are not secure against attacks; thus, they are enhanced by adding random challenges.

The complete protocol of EKE is described below and indicated in Figure 3.1:

- A randomly generates a public key E_A and uses it to produce $\mathsf{P}(E_A)$ in a symmetric cryptosystem then she sends $\mathsf{P}(E_A)$ along with her identity (*name*) to B. Here, A sends her name in clear form.

- Since they shared password before, B decrypts $\mathsf{P}(E_A)$ and gets E_A. Then, he produces a secret key K randomly and it is encrypted by using E_A then that results in encryption of $\mathsf{P}(E_A(K))$. Afterwards, he sends the result $\mathsf{P}(E_A(K))$ to A.

- A decrypts $\mathsf{P}(E_A(K))$ and obtains K. She chooses a matchless challenge $challenge_A$ and encrypts it with K to get $K(challenge_A)$. Then, she transmits $K(challenge_A)$ to B.

- B decrypts message and gets $challenge_A$. He chooses a matchless $challenge_B$ and encrypts two challenges with K to produce $K(challenge_A, challenge_B)$. Then, he transmits $K(challenge_A, challenge_B)$ to A.

- A decrypts message and gets challenges then she compares $challenge_A$ with her own challenge. If there is a matching, she encrypts $challenge_B$ and produces $K(challenge_B)$. Then, she sends $K(challenge_B)$ to B.

- B decrypts $K(challenge_B)$ and compares $challenge_B$ with his own challenge. If it matches, the login process is called successful and login session continues in the symmetric cryptosystem under the protection of session key K [9].

	A		B
1.	generate E_A, $\mathsf{P}(E_A)$	$\xrightarrow{\mathsf{P}(E_A),\mathsf{A}}$	
2.			decrypt $\mathsf{P}(E_A)$
		$\xleftarrow{\mathsf{P}(E_A(K))}$	generate K, $\mathsf{P}(E_A(K))$
3.	decrypt $\mathsf{P}(E_A(K))$		
	generate $challenge_A$		
	compute $K(challenge_A)$	$\xrightarrow{K(challenge_A)}$	
4.			decrypt $K(challenge_A)$
			generate $challenge_B$
		$\xleftarrow{K(challenge_A,challenge_B)}$	$K(challenge_A, challenge_B)$
5.	decrypt $K(challenge_A, challenge_B)$		
	generate $K(challenge_B)$	$\xrightarrow{K(challenge_B)}$	
6.			decrypt $K(challenge_B)$

Figure 3.1: The Complete EKE Protocol

Now, we briefly present three variants of EKE that are presented in [9] and attacks on them.

3.1.2 RSA-EKE

RSA cryptosystem is composed of a couple of numbers e and n where e is called as public key while d is private value. n is the multiplication of two large primes p and q and e is relatively prime to $\varphi(n)$ where

$$\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1)$$

The number d is calculated from $ed = 1 \pmod{(p-1)(q-1)}$. The equations for message m and ciphertext c are described respectively as $c = m^e \bmod n$ and $m = c^d \bmod n$.

The design of RSA-EKE is explained below and indicated in Figure 3.2:

- A randomly picks a public value e and a private value d. Since she has RSA's public key (e, n), she encrypts e with a common shared password P and sends her name A, n and $\mathsf{P}(e)$ to B.
- B decrypts $\mathsf{P}(e)$ by using P and obtains e. Then, a random session key K is chosen and $\mathsf{P}(K^e) \bmod n$ is calculated using the public key. He transmits result $\mathsf{P}(K^e)$ to A.
- A obtains K^e from the equation $\mathsf{P}^{-1}(\mathsf{P}(K^e))) = K^e$ and then by using her private key e, she gets the session key K.

	A		B
1.	public: (e, n)		
	compute $\mathsf{P}(e)$	$\xrightarrow{\mathsf{P}(e),n,\mathsf{A}}$	
2.			decrypt $\mathsf{P}(e)$, choose K
		$\xleftarrow{\mathsf{P}(K^e)}$	compute $\mathsf{P}(K^e) \bmod n$
3.	compute $K^e = \mathsf{P}^{-1}(K^e))$		
4.	use K for challenge-response		use K for challenge-response

Figure 3.2: RSA-EKE

Now both A and B has the session key K and they use it to start general challenge-response process so that they authenticate K.

3.1.3 DH-EKE

The implementation of Diffie-Hellman key exchange is given in Section 2.3.1. It is a powerful method because solving $g^{R_A R_B}$ mod p is substantially difficult without knowing the values R_A and R_B. However, DH can not resist man-in-the-middle attack. The EKE version of DH can solve this vulnerability.

The implementation of DH-EKE is described below and shown in Figure 3.3:

- A computes (g^{R_A} mod p) and encrypts it by password P. Then, she sends P(g^{R_A} mod p) to B with her name A along with the parameters the p and g. Consider, she transmits her name in clear form.

- Similarly, B computes (g^{R_B} mod p) then decrypts P(g^{R_A} mod p) by using the shared password P and produces the session key K from $K = (g^{R_A})^{R_B}$ mod p. At this time, B generates a random challenge $challenge_B$ and sends the parameters P(g^{R_B} (mod p)), $K(challenge_B)$ to A.

- A decrypts P(g^{R_B} (mod p)) and retrieves the value g^{R_B}. She also derived the session key K from $K = (g^{R_B})^{R_A}$ mod p and uses it to decrypt $K(challenge_B)$. Next, she randomly generates a challenge $challenge_A$ and produces $K(challenge_A, challenge_B)$ then sends it to B.

- B decrypts $K(challenge_A, challenge_B)$ and checks that if $challenge_B$ matches with his own challenge. If there is a matching, he sends $K(challenge_A)$ to A.

- A decrypts $K(challenge_A)$ and similarly verifies $challenge_A$.

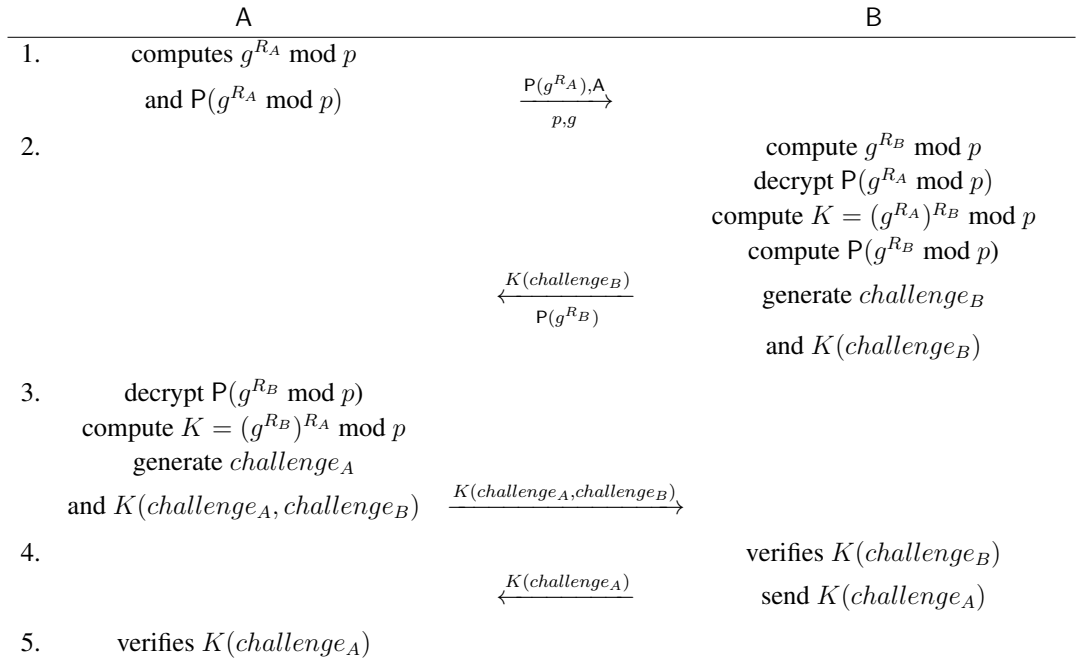

Figure 3.3: DH-EKE

If there is an eavesdropper who can obtain g^{R_A} and g^{R_B}, he can not approve password because R_A and R_B are generated randomly. That is, the values (g^{R_A} mod p) and (g^{R_B} mod p) also show up randomly. Moreover, it is not feasible to calculate the session key ($K = g^{R_A}g^{R_B}$ mod p) even though g^{R_A} and g^{R_B} are correctly guessed. Thanks to that reason, DH-EKE can avoid man-in-the-middle attack.

3.1.4 ElGamal-EKE

The ElGamal cryptosystem is a public key cryptosystem which is on the basis of Diffie-Hellman key exchange. The major property of the system is providing complication of finding discrete logarithm in a cyclic group.

At first, A and B settle on the parameters p and g. Then, A randomly picks a number R_A, computes $\mathsf{H} = g^{R_A}$ mod p and she publishes the public key parameters (G, p, g, h) while keeping R_A as a private key.

B randomly picks a number k from $1, ..., g-1$ and generates $y_1 \equiv g^k$ mod p and $y_2 \equiv m(g^{R_A})^k$ mod p where m is an encrypted message. Next, A gets the ciphertext message $(y_1 y_2)$ and obtains the message m from $m = y_2(y_1^{R_A})^{-1}$ mod p.

Figure 3.4 shows ElGamal-EKE.

	A		B
1.	compute $\mathsf{P}(g^{R_A} \bmod p)$	$\xrightarrow[\mathsf{P}(g^{R_A}\ modp),\mathsf{A}]{p,g}$	
2.			compute $(g^k \bmod p)$ compute $(Kg^{R_A k} \bmod p)$
		$\xleftarrow{X}$	$X = \mathsf{P}(g^k \bmod p, Kg^{R_A k} \bmod p)$
3.	decrypt X obtain K		
4.	continue with classical EKE		continue with classical EKE

Figure 3.4: ElGamal-EKE

Now, we present the description of ElGamal-EKE:

- A and B agreed on the parameters which are mentioned above. Initially, A generates $\mathsf{P}(g^{R_A} \bmod p)$ by using the password P and sends it with her name E, the prime p and the generator g.
- B randomly picks a number k then generates $(g^k \bmod p)$ and $(Kg^{R_A k} \bmod p)$ where K is the session key. Subsequently, he encrypts the parameters by P and sends the result $\mathsf{P}(g^k \bmod p, Kg^{R_A k} \bmod p)$ to B.
- A decrypts the message $\mathsf{P}(g^k \bmod p, Kg^{R_A k} \bmod p)$ and obtains the session key K. Now, she sends her challenge message and they continue with authentication of the session key K as described in the classical EKE protocol.

If there is an eavesdropper during the operation of the protocol, he can not find the password since the values R_A, K and k are selected randomly; thus, $(g_A^R \bmod p)$, $(g^k \bmod p)$ and $Kg^{R_A k}$ are also generated randomly.

3.2 SPEKE : Simple Password Exponential Key Exchange

Simple Password Exponential Key Exchange (SPEKE) is a balanced PAKE protocol that was proposed by D.Jablon [34] and it is recovered version of EKE protocol based on its weaknesses. It has been currently used in Entrust's TruePass end-to-end web

products and Blackberry phones [29]. Moreover, it was included in standard of ISO 11770-4 [1] and IEEE P1363 [26].

We discuss limitations, benefits and security of SPEKE and also present attacks on the protocol. There are so many attacks on SPEKE, but we especially focus on three attacks that are called "the exponential-equivalence attack" [71], "the impersonation attack" and "the key-malleability attack" [29].

The SPEKE protocol uses the following extra notations:

- $f(\mathsf{P})$: computes a group generator such that $g = f(\mathsf{P}) = \mathsf{P}^2 \bmod p$
- $E_K()$: a symmetric encryption function uses key K

The SPEKE protocol is composed of two stages which are called the key exchange stage and the key confirmation stage. The protocol is shown in Figure 3.5.

The key exchange stage is based on DH key exchange [34]:

- A randomly chooses a uniform secret value R_A from $\mathbb{Z}_p^* = \{1, ..., p-1\}$, computes $Q_A = f(\mathsf{P})^{R_A} \bmod p$ and sends it to B.
- Likewise, B randomly chooses a uniform secret value R_B from $\mathbb{Z}_p^* = \{1, ..., p-1\}$, computes $Q_B = f(\mathsf{P})^{R_B} \bmod p$ and sends it to A.
- A generates the session key $K = H(Q_B^{R_A} \bmod p)$ and B also generates the same session key $K = H(Q_A^{R_B} \bmod p)$.

Note that, the key exchange stage is symmetric and it simplifies security of the stage.

The key confirmation stage is performed to supply parties with the same session key K [34] :

- A selects a number C_A randomly, computes $E_K(C_A)$ and transmits it to B.
- B selects a number C_B randomly, computes $E_K(C_B, C_A)$ and transmits it to A.
- A checks whether C_A is the same with her own number, computes $E_K(C_B)$ and transmits it to B.

A		B
key exchange stage		
select $R_A \in \mathbb{Z}_p^*$		select $R_B \in \mathbb{Z}_p^*$
compute $Q_A = f(\mathsf{P})^{R_A} \bmod p$	$\xrightarrow{Q_A}$	compute $Q_B = f(\mathsf{P})^{R_B} \bmod p$
generate $K = \mathsf{H}(Q_B^{R_A} \bmod p)$	$\xleftarrow{Q_B}$	generate $K = \mathsf{H}(Q_A^{R_B} \bmod p)$
key confirmation (optional)		
select random C_A		
compute $E_K(C_A)$	$\xrightarrow{E_K(C_A)}$	
		select random C_B
	$\xleftarrow{E_K(C_A,C_B)}$	compute $E_K(C_A, C_B)$
verify C_A		
compute $E_K(C_B)$	$\xrightarrow{E_K(C_B)}$	
		verify C_B
alternative		
compute $\mathsf{H}(\mathsf{H}(K))$	$\xrightarrow[proof\ of\ K]{\mathsf{H}(\mathsf{H}(K))}$	
		verify $\mathsf{H}(\mathsf{H}(K))$
		if verification fails
	$\xleftarrow{\mathsf{H}(K)}$	abort the stage
verify $\mathsf{H}(K)$		
if verification fails		
abort the stage		

Figure 3.5: SPEKE

- Similarly, B checks C_B whether it is the correct number.

The key confirmation stage may be optional and it can be applied by using the session key K for both encryption and decryption of messages:

- A computes $\mathsf{H}(\mathsf{H}(K))$ and transmits proof of K.
- B checks whether $\mathsf{H}(\mathsf{H}(K))$ is correct and if it is not, he should aborts the confirmation stage. Otherwise, he sends $\mathsf{H}(K)$.
- A checks whether $\mathsf{H}(K)$ is correct and if it is not, she should aborts the confirmation stage.

In the SPEKE protocol, the $p-1$ must satisfy the property of having a large prime factor q in order to protect the protocol from discrete logarithm computations.

In addition, the function f creates a primitive base which may be dangerous to use. Because the values Q_A and Q_B are not encrypted, the attacker can try to test all candidates in smaller subgroups if $f(\mathsf{P})$ is chosen arbitrary from G_{p-1} [34].

3.3 B-SPEKE : "B "Extension of SPEKE Protocol

B-SPEKE protocol is an augmented PAKE protocol that is introduced by Jablon [35]. Hablon creates "B "extension of his own SPEKE protocol [34] by using Bellovin and Merritt's A-EKE protocol [10] as a basis. In his extended protocol, DH is used to demonstrate knowledge of the password instead of digital signature. Moreover, the purpose of the protocol is to prevent an eavesdropper and the man-in-the-middle attack.

The algorithm of B-SPEKE protocol is given below [12] and Figure 3.6:

	A		B
1.	$\mathsf{P}_1 = H(\mathsf{P})^2$ select $R_A \in \{1, ..., 2^L\}$ compute $A = \mathsf{P}_1^{R_A} \bmod p$	$\xrightarrow{A}$	
2.			select R_B, $R'_B \in \{1, ..., 2^L\}$ compute $Z = A^{2R_B} \bmod p$ $B = \mathsf{P}_1^{R_B}$, $B' = g^{R'_B} \bmod p$ $V_B = H_1(A, B, Z, \mathsf{P}_1)$ $Z_{AB} = \mathsf{P}_2^{R'_B}$
		$\xleftarrow{(B,B',V_B)}$	
3.	compute $Z = B^{2R_A} \bmod p$ verify V_B $Z_{AB} = B'^{\mathsf{P}}$ $V_A = H_2(A, B, Z, Z_{AB}, \mathsf{P}_1)$	$\xrightarrow{v_A}$	
4.	compute $K = H_0(Z)$		verify V_A compute $K = H_0(Z)$

Figure 3.6: B-SPEKE Protocol

1. At the beginning of the protocol, B stores two password images such that $\mathsf{P}_1 = H(\mathsf{P})^2$ and $\mathsf{P}_2 = g^{\mathsf{P}}$. A begins with selecting a random number $R_A \in \{1, ..., 2^L\}$ where L is a security parameter and computes a public parameter $A = \mathsf{P}_1^{R_A} \bmod p$. Then, she sends A to B.

2. B picks numbers R_B, $R'_B \in \{1, ..., 2^L\}$ randomly and computes $Z = A^{2R_B}$ such that $Z \not\equiv -1, 0$ or 1 mod p. Then, he computes $B = \mathsf{P}_1^{R_B} \bmod p$, $B' = g^{R'_B} \bmod p$, also a verifier $V_B = H_1(A, B, Z, \mathsf{P}_1)$ and $Z_{AB} = \mathsf{P}_2^{R'_B}$. The parameters B, B' and V_B are sent.

3. A generates $Z = B^{2R_A} \bmod p$ such that $Z \not\equiv -1, 0$ or 1 mod p and verifies V_B. To respond, she must send a knowledge of the DH key $Z_{AB} = B'^{\mathsf{P}}$, but it may cause a dictionary attack on P when the value Z_{AB} is used with a challenge value B'. Thus, A sends a hash result $V_A = H_2(A, B, Z, Z_{AB}, \mathsf{P}_1)$ that makes possible to check knowledge of P and Z so SPEKE protocol is extended to B-SPEKE protocol with this method.

4. B verifies V_A and if the verification does not fail, he generates $K = H_0(Z)$.

5. Finally, A also generates $K = H_0(Z)$.

The security of B-SPEKE protocol depends on DH and the discrete logarithm problem. However, Jablon [35] does not state formal security proof for the protocol.

3.4 SPAKE : Simple Password-Based Encrypted Key Exchange Protocol

Simple Password-based Key Exchange (SPAKE) is a balanced PAKE protocol that was introduced by Abdalla and Pointcheval [3]. The purpose of this two-party PAKE protocol is decreasing the usage of random oracles in the security proof. Abdalla and Pointcheval expressed two variants of SPAKE which are SPAKE1 and SPAKE2 whose security can be demonstrated under assumption of the difficulty of the computational Diffie-Hellman (CDH) problem in the random oracle model. Also, they introduce new variant of Diffie-Hellman assumptions and use them to prove security of their protocols.

Furthermore, since SPAKE1 and SPAKE2 are not in need of ideal ciphers onto a

group or full domain hash functions, they are easier to implement than other two-party PAKE protocols.

SPAKE1 is a non-concurrent PAKE protocol that is subject to multi-dimensional version of set password-based chosen basis CDH problem (S-PCCDH) in [3]. In the protocol, the values M_A, $M_B \in G$ are fixed public values such that they ensure the values of m and m' are calculated only once. Moreover, ID_A and ID_B denote the identities of parties.

The protocol runs with an only key agreement stage after a secret password $\mathsf{P} \in \mathbb{Z}_q$ is shared between the parties:

- A picks a $R_A \in \mathbb{Z}_q$ randomly and calculates $A = g_A^R \bmod p$. Then, she produces her public value $m = M_A^{\mathsf{P}} A \bmod p$ and sends it to B. Likewise, B picks a $R_B \in \mathbb{Z}_q$ randomly, calculates $B = g_B^R \bmod p$ and his public value $m' = M_B^{\mathsf{P}} B \bmod p$. Next, he sends message m'.

- A computes $K_A = (m'/M_B^{\mathsf{P}})^{R_A}$ and uses it to produce a session key $K_s = H(ID_A, ID_B, m, m', K_A)$. Similarly, B computes $K_B = (m/M_A^{\mathsf{P}})^{R_B}$ and the session key $K_s = H(ID_A, ID_B, m, m', K_B)$. Note that, $K_A = K_B = g^{R_A R_B}$.

A	shared password P	B
select $R_A \in \mathbb{Z}_q$		select $R_B \in \mathbb{Z}_q$
compute $A = g_A^R \bmod p$	$\xrightarrow{m}$	compute $B = g_B^R \bmod p$
$m = M_A^{\mathsf{P}} A \bmod p$	$\xleftarrow{m'}$	$m' = M_B^{\mathsf{P}} B \bmod p$
compute $K_A = (m'/M_B^{\mathsf{P}})^{R_A}$		$K_B = (m/M_A^{\mathsf{P}})^{R_B}$
$K_s = H(ID_A, ID_B, m, m', K_A)$		$K_s = H(ID_A, ID_B, m, m', K_B)$

Figure 3.7: SPAKE1 Protocol

Figure 3.7 shows SPAKE1 protocol. SPAKE2 is a concurrent PAKE protocol which is based on the CDH problem. SPAKE2 is exactly the same with SPAKE1 protocol. The only difference is that the production of session key K includes the password P, that is $K = H(ID_A, ID_B, m, m', \mathsf{P}, K)$.

3.5 SRP : Secure Remote Password Protocol

The SRP protocol is augmented PAKE protocol that was proposed by Thomas Wu [63]. It is known as a zero-knowledge proof protocol where S does not have to preserve hashed version of the password and C can authenticate S securely. Especially, C does not send a password itself or any other information, that can be derived from the password, while authenticating S.

The SRP protocol allows C to authenticate herself to a S without need of transmitting the password and without trusting third party. Also, it provides to generating a secure session key so that the parties can transmit encrypted data providing higher security on top of TLS. Additionally, SRP withstands online and offline dictionary attacks. Moreover, it ensures forward secrecy [63].

In the protocol, S stores the password in a form {*clientname*, v, s} where v is a password verifier and s is a random string utilized as salt of C. The password entries (x, v) are generated as follows:

$$x = \mathsf{H}(s, \mathsf{P})$$

$$v = g^x \mod p$$

Note that, the parameter P is C's password and parties use SHA hash function to generate the session key K.

The SRP protocol is described below and given in Figure 3.8:

1. C sends his clientname *clientname*.

2. S gets salt s from password file and transmits s to C. Also, C computes a long term private key x using password file.

3. After identifying himself to S, C randomly generates a number $a \in (1, p)$ and calculates an ephemeral public key $A = g^a \bmod p$ then transmits it to S.

4. S randomly generates a number b and u, calculates an ephemeral public key $B = v + g^b \bmod p$ and transmits it to C with the parameter u. The parameter u is a 32-bit unsigned integer and it is first 32-bits of $\mathsf{H}(B)$.

5. C computes the common $S = (B - g^x)^{a+ux} \bmod p$ and S computes $S = (Av^u)^b$. Actually, the common value S is $S = g^{ab+bux} \bmod p$.

6. The parties produce the session key K by using $K = \mathsf{H}(S)$.

7. Finally, they have to prove that their keys are identical. To ensure this, C sends $M_1 = \mathsf{H}(A, B, K)$. Then, S computes his own M_1 and compares it with C's M_1.

8. If there is a matching between M_1s, S sends $M_2 = \mathsf{H}(A, M, K)$ to C. Then, C verifies M_2s and ıf the verification fails, C aborts the authentication [63].

	C		S
1.		$\xrightarrow{clientname}$	{*clientname*, v, s}
2.	compute private $x = \mathsf{H}(s, \mathsf{P})$	$\xleftarrow{s}$	
3.	select $a \in (1, p)$		
	compute public key $A = g^a \bmod p$	$\xrightarrow{A}$	
4.		$\xleftarrow{(B,u)}$	select b and u
			compute public key $B = v + g^b \bmod p$
5.	compute $S = (B - g^x)^{a+ux}$		compute $S = (Av^u)^b$
	$(S = g^{ab+bux})$		$(S = g^{ab+bux})$
6.	compute $K = \mathsf{H}(S)$		compute $K = \mathsf{H}(S)$
7.	compute $M_1 = \mathsf{H}(A, B, K)$	$\xrightarrow{M_1}$	verify M_1
8.	verify M_2	$\xleftarrow{M_2}$	$M_2 = \mathsf{H}(A, M_1, K)$

Figure 3.8: SRP

The authentication is aborted by C if $B \equiv 0 \bmod n$ and is aborted from S if $A \equiv 0 \bmod p$. The parameters p and g can be set at first or C can provide them for C by sending them in the first message along with the salt s. Moreover, the parameter p should be a safe prime such that $p = 2q + 1$ where q is a prime number.

The value B $(B = g^b + v)$ plays an important key role in the protocol against an active dictionary attack. Suppose that there is an attacker who masquerades S and persuades C to make an authentication attempt and also suppose the value B is just calculated as $B = g^b \bmod p$.

- C sends his clientname to the attacker.

- The attacker has the salt s which was snooped earlier and sends it to C.
- C sends his public exponential residue A.
- The attacker selects random values b and u, computes his own exponential residue B and sends B to C along with u.
- C generates his own session key K from the $S = B^{a+ux}$ mod p and sends a proof of that K to the attacker.
- The attacker imitates network failure or inform C that the password is not correct.
- The attacker has the value A, b and a proof of K. Now, he can guess a password P$'$ and compute x' and v'. Then, he can produce S' from $S' = (Av'^u)^b$ and compute K' using the formula $K' = H(S')$. Finally, the attacker checks whether K' is equal to C's proof of K or not. If they match, the guessed password P$'$ is valid [63].

In addition, the value u plays an important role against attacks. Assume, the attacker has attained value v and masquerades as a fake C and presume that, the value u is somehow captured.

- The attacker sends C's $clientname$ to S.
- S sends C's salt s to the attacker.
- The attacker generates $A = g^a v^{-u}$ mod p and sends it to S.
- S computes $B = v + g^b$ mod p and sends it to the attacker.
- The attacker generates the session key K by using the formula

  $$K = H((B - v)^a \ mod \ p)$$

  Then, he sends a proof of this K to C.
- S computes his session key :

  $$S = (Av^u)^b = (g^a v^{-u} v^u)^b = g^{ab}$$

This attack works because the value S does not depend on the long term keys. Hence, the attacker can calculate it and when the attacker has the same session key, he can easily imitate C. For this reason, if the value u is publicly known, the protocol becomes vulnerable to this attack [63].

SRP protocol is also known as SRP-3 and is included in IEEE P1363 standard [26]. Later, Wu [62] proposed a variant of SRP-3 which is SRP-6 protocol and it is also standardised in IEEE P1363 standard [26] and ISO standard [59].

For the security analysis of SRP protocol, please see Section 4.4

3.6 AugPAKE : Efficient Augmented Password-Only Authentication Key Exchange

AugPAKE protocol is an augmented PAKE protocol that was given by Shin and Kobara [52]. In the general PAKE protocols, the password is chosen from a relatively small dictionary and it makes them susceptible to offline dictionary attacks. Specifically, the attacker can obtain the password by applying exhaustive searches. The Augmented PAKE protocol (AugPAKE) is a two-party password authenticated key exchange protocol such that C keeps low-entropy password secret and the verifier of the password is registered in S. It provides "resistance to server compromise" which means that the attacker can not masquerade as C without applying offline dictionary attack on the verifier of password even if he obtains the verifier from S. Besides protection for server compromise, it provides security against passive, active and offline dictionary attacks [52]. Furthermore, it was standardised in the ISO 11770-4 [1].

Before introducing the protocol description, we give notations for the protocol:

- CI : the identity of client such that $U \in \{0,1\}^*$ where $\{0,1\}^*$ denotes a set of finite binary strings.
- SI : the identity of S such that $S \in \{0,1\}^*$.
- $0x$: a hexadecimal value.
- P : the password of C. It can be used by way of $\mathsf{H}'(0x00\|CI\|SI\|\mathsf{P})$.

- W : the password' verifier in S. The value W is produced in two ways : $W = g^{\mathsf{P}} \bmod p$ where w is the clear form of password or $W = g^{\mathsf{P}'} \bmod p$ where $\mathsf{P}' = \mathsf{H}'(0x00 \| CI \| SI \| \mathsf{P})$.
- $M(x)$: the function which converts integer x to binary string.

The AugPake protocol comprises of two stages which are called initialization and protocol execution.

The parties should carry out initialization part securely at the beginning of the protocol. Firstly, C computes ($W = g^{\mathsf{P}'} \bmod p$) and sends W to S. S registers W as a verifier of the password w'.

The protocol execution works as follow and it is shown in Figure 3.9:

- C begins with selecting $x \in \mathbb{Z}_q^*$. Then, C computes ($X = g^x \bmod p$) and transmits the 1^{st} message (CI, X) to S. Note that, the public value X is a DH value of C.
- S checks that $CI \not\equiv 0, 1$ or -1 in modulo p. If it is so, the protocol is terminated by S. Otherwise, S randomly selects a number $y \in \mathbb{Z}_q^*$ and calculates ($Y = (X.W^r)^y \bmod p$) where the number r is produced from $r = \mathsf{H}'(0x01 \| CI \| SI \| M(X))$. Next, S transmits the 2^{nd} message (SI, Y) to C.
- C verifies that $SI \not\equiv 0, 1$ or -1 in modulo p then generates ($K = Y^z \bmod p$) where $z = \frac{1}{(x+(Pr))} \bmod q$ and $r = \mathsf{H}'(0x01 \| CI \| SI \| M(X))$. Moreover, C produces an "authenticator" V_u where

  $$V_u = \mathsf{H}(0x02 \| CI \| SI \| M(X) \| M(Y) \| M(K))$$

  and sends the 3^{rd} message V_u.
- Now, S verifies that $CI \not\equiv \mathsf{H}(0x02 \| CI \| SI \| M(X) \| M(Y) \| M(K))$ where ($K = g^y \bmod p$) and produces an "authenticator" V_s where

  $$V_s = \mathsf{H}(0x03 \| CI \| SI \| M(X) \| M(Y) \| M(K))$$

  and sends the 4^{th} message V_s to C. Also, the session key

  $$S_k = \mathsf{H}(0x04 \| CI \| SI \| M(X) \| M(Y) \| M(K))$$

is produced in this stage.

- Finally, C verifies that $V_s \not\equiv \mathsf{H}(0x03\|CI\|SI\|M(X)\|M(Y)\|M(K))$ and produces the session key

$$S_k = \mathsf{H}(0x04\|CI\|SI\|M(X)\|M(Y)\|M(K)).$$

When the generating of S_k is completed, the parties should delete all the internal states from memory.

Note that, if verifications can not result in success at all stages, the protocol execution is terminated by related party. Also, the session key S_k can be produced by using a key derivation functions.

	C		S
1.	select $x \in \mathbb{Z}_q^*$ compute $X = g^x \bmod p$	$\xrightarrow{(CI,X)}$	
2.			check $CI \not\equiv 0, 1, -1 \bmod p$ select $y \in \mathbb{Z}_q^* \bmod p$
		$\xleftarrow{(S,Y)}$	compute $Y = (XW^r)^y \bmod p$
3.	check $SI \not\equiv 0, 1, -1 \bmod p$ compute $K = Y^z \bmod p$, and V_u	$\xrightarrow{V_u}$	
4.			verify V_u
		$\xleftarrow{V_s}$	compute V_s and S_k
5.	verify V_s compute S_k		

Figure 3.9: AugPAKE

It is important that the messages should be sent sequentially in the protocol execution in order to prevent possible attacks. For instance, if the 2^{nd} message (CI, Y) and the 4^{th} message V_s are sent together by S, it becomes possible to derive the valid password P by applying offline dictionary attacks.

In the protocol, the parameters p and q should be chosen as large primes which satisfy that q is a denominator of $((p-1)/2)$ and every multiplier of $((p-1)/2)$ are primes to q comparably in length. The prime p is named as secure prime and it provides

computational gains for the protocol. That is, the parties do not need to check order of the parameters they received [52].

3.7 J-PAKE : Password Authenticated Key Exchange by Juggling

J-PAKE is a balanced PAKE protocol that was introduced by F.Hao and P.Ryan [31] and it is used over a finite field and elliptic curves. It was standardised in the ISO-11770-4 [1].

3.7.1 J-PAKE Protocol Over a Finite Field

We describe the two rounds of the protocol at first [30]:
Before starting to key exchange, A and B close on the parameters p, q and g by using the method in NIST FIPS 186-4 [32].

Let $s \in [1, q-1]$ be a secret key that is obtained from mutual low-entropy password of A and B. In J-PAKE s, $s+q$, $s+2q$,... are conceived equivalent values; therefore, the range of secret key s is a necessary condition for the protocol setup [31].

In the first round of J-PAKE protocol, A selects uniformly two ephemeral random keys x_1 and x_2 such that $x_1 \in [0, q-1]$ and $x_2 \in [1, q-1]$. B also randomly picks two ephemeral random private keys x_3 and x_4 such that $x_3 \in [0, q-1]$ and $x_4 \in [1, q-1]$ as well.

A computes the parameters g_1 and g_2 where $g_1 = g^{x_1} \bmod p$ and $g_2 = g^{x_2} \bmod p$; then, she sends the parameters with zero knowledge proofs for x_1 and x_2. Likewise, B computes the g_3 and g_4 where $g_3 = g^{x_3} \bmod p$ and $g_4 = g^{x_4} \bmod p$; then, he sends the parameters with zero knowledge proofs for x_3 and x_4.

One of the methods to use the zero knowledge proof is Schnorr NIZK proof [28]. For example, A aims to send zero knowledge proof for $D = g^d \bmod p$. The parameters of Schnorr NIZK proof for D are client id, which is unique identity of A, a uniform random number V such that $V \in [0, q-1]$ and output of hash function $\mathsf{H}(g \| V \| D \| ClientID)$. The uniqueness of C id means that A should use a unique identity when communicating with more than one party. After receiving a Schnorr NIZK proof, A checks whether B's C id is a valid identity and it is not the same with

her own identiy. In the J-PAKE protocol, each party should guarantee that identity of other party has remained the same until the end of the protocol.

At the end of first round, A verifies zero knowledge proof of the parameters x_3 and x_4 and also checks that $g_4 \neq 1 \bmod p$. B verifies zero knowledge proof of the parameters x_1 and x_2 and checks that $g_2 \neq 1 \bmod p$ as well [28].

In the second round of protocol, A computes $A = (g_1 g_3 g_4)^{x_2 s} \bmod p$ and transmits A to B along with zero knowledge proof for $x_2 s$. Similarly, B computes $B = (g_1 g_2 g_3)^{x_4 s} \bmod p$ and transmits B with zero knowledge proof for $x_4 s$.

The zero knowledge proofs of the parameters $x_2 s$ and $x_4 s$ can be computed in the same way but consider that the generators $(g_1 g_3 g_4)$ and $(g_1 g_2 g_3)$ are used instead of g respectively by A and B. At this point, A and B only need to guarantee whether $g_1 g_3 g_4 \neq 1 \bmod p$ and $g_1 g_2 g_3 \neq 1 \bmod p$. There is a small possibility for these inequalities to occur, even if C is in communication with an opposer.

To complete the second round, A and B verify zero knowledge proofs of the parameters which they received. Then, A calculates $K_A = (B/g_4^{x_2 s})^{x_2} \bmod p$ and B computes $K_B = (A/g_2^{x_4 s})^{x_4} \bmod p$ where $K_A = K_B = g^{((x_1+x_3)x_2 x_4 s)} \bmod p$. After agreeing on the the same key $K = K_A = K_B$, they compute the common session key from K by applying a KDF.

	C		S
	first round		
1.	select random x_1 and x_2		select random x_3 and x_4
2.	compute $g_1 = g^{x_1} \bmod p$	$\xrightarrow{(x_1, x_2, ZKPs)}$	compute $g_3 = g^{x_3} \bmod p$
	and $g_2 = g^{x_2} \bmod p$	$\xleftarrow{(x_3, x_4, ZKPs)}$	and $g_4 = g^{x_4} \bmod p$
3.	verify ZKPs of x_3 and x_4 check $g_4 \neq 1 \bmod p$		verify ZKPs of x_1 and x_2 check $g_2 \neq 1 \bmod p$
	second round		
4.	compute $A = (g_1 g_3 g_4)^{x_2 s} \bmod p$	$\xrightarrow{(A, ZKP)}$ $\xleftarrow{(B, ZKP)}$	compute $B = (g_1 g_2 g_3)^{x_4 s} \bmod p$
5.	verify ZKP of $x_4 s$		verify ZKP of $x_2 s$
6.	compute $K_A = (B/g_4^{x_2 s})^{x_2} \bmod p$ $(K_A = K_B = g^{((x_1+x_3)x_2 x_4 s)} \bmod p)$		$K_B = (A/g_2^{x_4 s})^{x_4} \bmod p$ $(K_A = K_B = g^{((x_1+x_3)x_2 x_4 s)} \bmod p)$

Figure 3.10: J-PAKE Protocol over a finite field

Figure 3.10 describes J-PAKE protocol over a finite field.

3.7.2 J-PAKE Protocol up over an Elliptic Curve

The J-PAKE protocol also works over an elliptic curve and the only difference is that an additive group is used instead of multiplicative group over a finite field.

Consider an elliptic curve $E(F_p)$ such that it is identified over a finite field F_p and Gr is a generator for the subgroup over $E(F_p)$ of a prime order n. We denote scalar multiplication of an elliptic curve point P with a scalar b over $E(F_p)$ as $[b]P$; that is, $\underbrace{P + P + \cdots + P}_{b \text{ times}}$. The curves which are used in the protocol can be chosen from the NIST curves. As in the protocol over a finite field, let $s \in [1, n-1]$ be a secret key between A and B.

A selects uniformly two ephemeral private keys x_1 and x_2 such that $x_1, x_2 \in [1, n-1]$ and B also selects two ephemeral private keys x_3 and x_4 such that $x_3, x_4 \in [1, n-1]$. Then, A computes $Gr_1 = [x_1]Gr$ and $Gr_2 = [x_2]Gr$ and sends Gr_1 and Gr_2 to B with zero knowledge proofs of x_1 and x_2. Similarly, B computes $Gr_3 = [x_3]Gr$ and $Gr_4 = [x_4]Gr$ and sends them to A with zero knowledge proofs of x_3 and x_4. To complete first round, A and B verify the zero knowledge proofs that they received [28]. Moreover, each party needs to check that other's C id is different from its own id and is valid. In the second round of protocol over an elliptic curve, A sends the value A and $x_2 s$ to B where $A = ([x_2 s](Gr_1 + Gr_3 + Gr_4)$. Likewise, B sends to A the value B and $x_4 s$ where $B = ([x_4 s](Gr_1 + Gr_2 + Gr_3))$. When the second round is finished, the zero knowledge proofs of the parameters are verified by A and B. Note that, verification of the parameters is done as in the former round but this time, the generator of A is $(Gr_1 + Gr_3 + Gr_4)$ and the generator of B is $(Gr_1 + Gr_2 + Gr_3)$. If the verification is completed successfully, A calculates $K_a = ((B - ([x_2]Gr_4)[x_2])$ and B calculates $K_b = ((A - ([x_4]Gr_4)[x_4])$. Consequently, they settle over the common key K where $K = K_a = K_b$ and use it to create a session key by applying a KDF.

3.7.3 Three-Pass Variant of J-PAKE Protocol

The two round of J-PAKE protocol is a symmetric protocol; thus, security analysis is significantly simple. In the protocol, one party sends parameters and the other

responds in the same way. Hence, the protocol can be expanded to three passes with keeping it secure.

Let the protocol work over an finite field, A selects uniformly two ephemeral keys x_1, x_2 and computes $g_1 = (g_1^x \bmod p)$ and $g_2 = (g_2^x \bmod p)$. Afterwards,, she sends the parameters g_1 and g_2 to B with zero knowledge proofs of x_1 and x_2. Similarly, B selects uniformly two ephemeral keys x_3, x_4 and computes $g_3 = (g_3^x \bmod p)$, $g_4 = (g_4^x \bmod p)$ and $B = ((g_1 g_2 g_4)^{x_4 s} \bmod p)$. Then, he sends the parameters g_3, g_4 and B to A with zero knowledge proofs of x_3, x_4 and $x_4 s$.

After obtaining the parameters, A computes $A = ((g_1 g_3 g_4)^{x_2 s} \bmod p)$ and sends it to B A with zero knowledge proof of $x_2 s$. Finally, the verification and agreement of session key is the same as previous process.

3.8 OPAQUE Protocol

In an augmented (asymmetric) PAKE protocol, S stores C's password under a hash function without storing plaintext form of the password and without using Public Key Infrastructure (PKI) protocol. Although many of aPAKE protocols are called PKI-free, they are not secure against pre-computation attacks. The reason for that is these protocols generally do not use a secret salt. The salt is sent from S to C in cleartext when they are used. Hence, secrecy of the salt can not be ensured against pre-computation attacks.

OPAQUE protocol's design comes from a work of Ford and Kaliski [23] and it is described in [37]. It is the first PKI-free aPAKE protocol which supports mutual authentication with the ability of being secure against pre-computation attack while using a secret salt.

OPAQUE consists of two functional properties that are an Oblivious Pseudo Random Function (OPRF) and a key-exchange protocol. For OPRF, please see Section 2.2.

OPAQUE protocol uses a specific function of OPRF which is called DH-OPRF. Firstly, we introduce protocol setup for DH-OPRF with notations.

- DH-OPRF domain: any random string into G

- DH-OPRF operation: $F(k; x) = H(x, v, H'(x)^k)$

The process of DH-OPRF computation is given below:

- C chooses random $r \in [0, q-1]$ and sends $a = H'(x).g^r$ to S.
- S responds with $v = g^k$ and $b = a^k$.
- After obtaining the values b and v, C generates PRF output $\mathsf{H}(x, v, bv^{-1})$.
- Finally, the parameters (a, b, v) are checked whether they are non-unit elements in G [37].

Now, we introduce the protocol setup for OPAQUE protocol. OPAQUE consists of OPRF protocol and a key exchange (KE) protocol. Before execution of the protocol, the parties agree on which KE protocol to be used [41].

Let C have private and public keys that are denoted as u_1 and u_2 respectively. Similarly, S has private key and public key s_1 and s_2 respectively.

Initially, the password registration process works between C and S. It is considered that S can be authenticated by C throughout the password registration process [41] :

- C chooses a password p_u with the parameters u_1 and u_2 before coming to an agreement on a key exchange protocol.
- S randomly chooses a OPRF key k_u with the parameters s_1 and s_2. Also, S computes $v = g^{k_u}$ and sends the parameter s_2 to C. Note that the parameter k_u is randomly chosen such that it is independent for each C.
- C and S run OPRF function $F(k_u, p_u)$.
- Only C learns the result which is called "randomized password" and denoted by r_u.
- C generates an envelope function Env_u such that $Env_u = AutEnc(r_u,$ $u_1, u_2, s_2, v)$ where $AutEnc$ is an authenticated encryption function in [37]. In this equation, all the parameters except v require authentication and only

u_1 is encrypted. Also, there are some inferences about the parameters in this equation. For example, if u_2 is rearranged from u_1, it can be omitted with v from the equation. Note that, S needs to send v with its OPRF response in this situation.

- C sends Env_u with u_2 to S and deletes p_u, r_u and all keys.
- S stores $(Env_u, s_1, s_2, u_2, k_u, v)$ in a C-specific record. The parameters s_1 and s_2 can be kept individually and removed from the record if they are assigned to distinguished Cs.

	C		S
	password registration		
1.	private u_1, public u_2 select password p_u		private s_1, public s_2
2.		$\xleftarrow{s_2}$	select a OPRF key ku compute $V = g^{k_u}$
3.	run OPRF function $F(k_u, p_u)$		run OPRF function $F(k_u, p_u)$
4.	generate Env_u	$\xleftarrow{(v, OPRF\ response)}$	
	delete p_u, r_u, all keys	$\xrightarrow{(Env_u, u_2)}$	
5.			store $(Env_u, s_1, s_2, u_2, k_u, v)$
	protocol execution		
6.		$\xrightarrow{account\ information}$	
7.	run OPRF function		OPRF function and get r_u
8.		$\xleftarrow{Env_u}$	
9.	decrypt Env_u authenticate s_2		
10.	run key-exchange protocol		run key-exchange protocol

Figure 3.11: OPAQUE Protocol

At the end of this step, registration part is completed and the parties can operate the protocol [41]:

- C sends account information to S in order for S to take C's information back.
- C and S execute OPRF function and S obtains the value r_u.

- S sends Env_u to C.
- C decrypts Env_u by using r_u and obtains C's private and public keys while authenticating S's public key.
- The parties run the key exchange protocol by using the public and private keys of each party.

The complete protocol of OPAQUE is given in Figure 3.11.

3.9 PAK and PPK Protocols

The PAK (Password Authenticated Key exchange) protocol is a balanced PAKE protocol that was proposed by V.Boyko et al. [13]. It provides mutual explicit authentication while using the Diffie-Hellman exchange and security against all passive and active attacks. Moreover, it consists of a key agreement stage and a key authentication stage. The most known variants of PAK protocols are PPK and PAK-X that are introduced in [13] and we will mention the PPK (Password Protected Key exchange) variant in this section. There are more variants of this protocol like PAK-EC, PAK-R, PAK-Y and they are given in [45].

Assume that κ is a main security parameter of hash functions such that it consists of 128 or 160 bits. $\ell > \kappa$ is the security parameter for discrete-log-based public keys such that it consists of 1024 or 2048 bits.

Also, the set of finite binary strings and the set of binary strings of length n are denoted by $\{0,1\}^*$ and $\{0,1\}^n$ respectively. In the protocol, $\mathsf{H}_1 : \{0,1\}^* \rightarrow \{0,1\}^n$ where $n \geq \ell + \kappa$ is the derivation function and $\mathsf{H}_{2a}, \mathsf{H}_{2b}, \mathsf{H}_3 : \{0,1\}^\kappa \rightarrow \{0,1\}^*$ are random hash functions.

The PAK protocol is presented as follows and it is given Figure 3.12:

- A and B share a secret password P that ensures the equation $\mathsf{H}_1(A\|B\|\mathsf{P}) \neq 0$, $\mathsf{H}_{2a}(A\|B\|\mathsf{P}) \neq 0$ and $\mathsf{H}_{2b} \neq 0$ where the pair (A, B) denotes identities of Cs.
- A randomly picks a secret value R_A and computes $X = g^{R_A} \bmod p$. Similarly, B randomly picks a secret value R_B and computes $Y = g^{R_B} \bmod p$. Note that,

g^{R_A} and g^{R_B} are Diffie-Hellman values.

- A computes the message $m = g^{R_A}(\mathsf{H}_1(A\|B\|\mathsf{P}))^r$ and sends it to B.
- B verifies that $m \not\equiv 0 \bmod p$ and he aborts the procedure if $m = 0$. After verification, he computes $\sigma = (\frac{m}{(\mathsf{H}_1(A\|B\|\mathsf{P}))^r})^{R_B}$ and $k = H_{2a}(A\|B\|m\|Y\|\sigma\|\mathsf{P})$ then sends k with the value Y.
- A computes $\sigma = Y^{R_A}$ and verifies that $k = \mathsf{H}_{2a}(A\|B\|m\|Y\|\sigma\|\mathsf{P})$. Then, she also computes $k' = \mathsf{H}_{2b}(A\|B\|m\|Y\|\sigma\|\mathsf{P})$ with the session key $K = \mathsf{H}_3(A\|B\|m\|Y\|\sigma\|\mathsf{P})$ and send k' to B.
- B verifies that $k' = \mathsf{H}_{2b}(A\|B\|m\|Y\|\sigma\|\mathsf{P})$ and computes $K = \mathsf{H}_3(A\|B\|m\|Y\|\sigma\|\mathsf{P})$.

If all of the verifications are completed successfully, the parties authenticate each other and produce the session key. Otherwise, the protocol fails.

	A	share password P	B
1.	select random R_A compute $X = g^{R_A} \bmod p$		select random R_B compute $Y = g^{R_B} \bmod p$
2.	$m = g^{R_A}(\mathsf{H}_1(A\|B\|\mathsf{P}))^r$	$\xrightarrow{m}$	
3.			verify $m \not\equiv 0 \bmod p$ $\sigma = (\frac{m}{(\mathsf{H}_1(A\|B\|\mathsf{P}))^r})^{R_B}$ $k = \mathsf{H}_{2a}(A\|B\|m\|Y\|\sigma\|\mathsf{P})$
		$\xleftarrow{(Y,k)}$	
4.	compute $\sigma = Y^{R_A}$ verify $k = \mathsf{H}_{2a}(A\|B\|m\|Y\|\sigma\|\mathsf{P})$ $k' = \mathsf{H}_{2b}(A\|B\|m\|Y\|\sigma\|\mathsf{P})$ $K = \mathsf{H}_3(A\|B\|m\|Y\|\sigma\|\mathsf{P})$	$\xrightarrow{k'}$	
5.			verify $k' = \mathsf{H}_{2b}(A\|B\|m\|Y\|\sigma\|\mathsf{P})$ $K = \mathsf{H}_3(A\|B\|m\|Y\|\sigma\|\mathsf{P})$

Figure 3.12: PAK Protocol

Moreover, R_a and R_b which are Diffie-Hellman parameters should be minimum 384 bits, the outputs of hash functions H_1 and H_{2a} should have 1152 bits while H_{2b} and H_3 should have 128 bits. In addition, the prime p and the session key K should be 1024 and 128 bits long respectively [15].

In the PAK protocol, the parties share the secret key and other values; thus, the protection and managing these values are important. If there is a potential attacker such

that he can represent himself as one of the parties, he can apply discrete logarithm attack on the multiplicative group of congruences mod p. Then, he can produce a discrete logarithms' table in order to use it as a dictionary. Hence, the parameters p and g should be chosen sufficiently large to prevent such attacks.

The PPK Protocol consists of implicit authentication which ensures the key is obtained by only one of the parties who claims to be. The PPK protocol is more useful than the PAK protocol because it does not need explicit authentication and it has only two rounds. The protocol description [13] is given below and is shown in Figure 3.13:

- Initially, two parties share a secret password P then A selects a secret value R_A randomly, calculates $X = g^{R_A} \bmod p$ and $m = g^{R_A}(\mathsf{H}_1(A\|B\|\mathsf{P}))^r$ and sends m.
- B verifies whether $m \not\equiv 0 \bmod p$ and after selection of a random secret R_B, he calculates $Y = g^{R_B}(H_1(A\|B\|\mathsf{P}))^r$ and sends Y. Also, he calculates $\sigma = (\frac{m}{(\mathsf{H}_1(A\|B\|\mathsf{P}))^r})^{R_B}$. Next, he produces the session key $K = \mathsf{H}_3(A\|B\|m\|Y\|\sigma\|\mathsf{P})$.
- A verifies $Y \not\equiv 0 \bmod p$ and calculates $\sigma = (\frac{Y}{(\mathsf{H}_1(A\|B\|\mathsf{P}))^r})^{R_A}$ then she produces the session key $K = \mathsf{H}_3(A\|B\|m\|Y\|\sigma\|\mathsf{P})$.

	A	share a password P	B
1.	select random R_A compute $X = g^{R_A} \bmod p$ $m = g^{R_A}(\mathsf{H}_1(A\|B\|\mathsf{P}))^r$	$\xrightarrow{m}$	
2.			verify $m \not\equiv 0 \bmod p$ select random R_B $Y = g^{R_B}(\mathsf{H}_1(A\|B\|\mathsf{P}))^r$ $\sigma = (\frac{m}{(\mathsf{H}_1(A\|B\|\mathsf{P}))^r})^{R_B}$ $K = \mathsf{H}_3(A\|B\|m\|Y\|\sigma\|\mathsf{P})$
		$\xleftarrow{Y}$	
3.	verify $Y \not\equiv 0 \bmod p$ compute $\sigma = (\frac{Y}{(\mathsf{H}_1(A\|B\|\mathsf{P}))^r})^{R_A}$ $K = \mathsf{H}_3(A\|B\|m\|Y\|\sigma\|\mathsf{P})$		

Figure 3.13: PPK Protocol

PAK protocol' security is based on DH assumptions and V.Boyko et al. [13] proved

the protocol' security in Shoup's simulation model in which the hash functions are used as random oracles. Furthermore, the PAK protocol is included in some standards such as ITU [2], IETF [15] and IEEE P1363 [26].

3.10 SESPAKE : Security Evaluated Standardized Password Authenticated Key Exchange

The SESPAKE protocol is a balanced PAKE protocol that was introduced by V.Smyshlyaev et al. [54]. It consists of a key agreement step and a key confirmation step. It is confirmed in the standardization system of the Russian Federation.

In SESPAKE protocol, the parties use DH to exchange keys in the key agreement step while they exchange strings that are subject to the generated key in the key confirmation step. The notatations of the protocol are listed below [54]:

- V_n : the set of all strings of length n and with elements from $GF(2)$
- E : a subgroup of prime order q
- P : a generator of E
- m : the order of the group
- F : a key derivation function (PBKDF2) is given in [38]
- τ : a run time for computing multiple points in the group of elliptic curve points
- ID : an identity of the party with length N ($ID = V_8^N$)
- P : a password of C such that $\mathsf{P} \in V_8^k$
- l : a random positive integer
- ind : a number such that $ind \in \{1, \ldots l\}$
- s : a salt such that $s \in V_{64}$
- T_A, T_B : open constant strings of the parties
- src : denotes scalar point multiplication

C keeps his password P secret and S stores a point Q_{P} as a secret where $Q_{\mathsf{P}} = F(\mathsf{P}, \mathsf{salt}, 2000) \cdot \mathsf{Q_{ind}}$. Additionally, they publicly stores $l+1$ provable pseudorandom points P and $Q_1 \dots Q_l$.

The protocol's description is shown in Figure 3.14 [54].

	A		B
1.	A_{ID}, P	$\xrightarrow{A_{ID}}$	$B_{ID}, Q_{\mathsf{P}}, ind, s$
		$\xleftarrow{B_{ID},ind,s}$	
2.	flag $z_A = 0$ compute $Q_{\mathsf{P}}^A = F(\mathsf{P}, \mathsf{s}, 2000) \cdot \mathsf{Q_{ind}}$ select random $\alpha \in \{1, \dots, q-1\}$ compute $u_1 = [\alpha]P - Q_{\mathsf{P}}^A$	$\xrightarrow{u_1}$	
3.			if $u_1 \notin E \rightarrow$ finish flag $z_B = 0$ compute $Q_B = u_1 + Q_{\mathsf{P}}$ select $\beta \in \{1, \dots, q-1\}$ if $[\frac{m}{q}]Q_B = 0_E \rightarrow Q_B = P, z_B = 1$ $src = [\frac{m}{q} \cdot \beta \bmod q]Q_B$ $K_B = H_{256}(src)$
		$\xleftarrow{u_2}$	$u_2 = [\beta]P + Q_{\mathsf{P}}$
4.	if $u_2 \notin E \rightarrow$ finish compute $Q_A = u_2 - Q_{\mathsf{P}}^A$ if $[\frac{m}{q}]Q_A = 0_E \rightarrow Q_A = P, z_A = 1$ $src = [\frac{m}{q} \cdot \alpha \bmod q]Q_A$ $K_A = H_{256}(src)$ $tag_A = (T_A \| A_{ID} \| ind \| s \| u_1 \| u_2)$ $M_A = HMAC_{K_A}(tag_A)$	$\xrightarrow{M_A}$	
5.			set $tag = (T_A \| A_{ID} \| ind \| s \| u_1 \| u_2)$ compute $M = HMAC_{KB}(tag)$ if $M \neq M_A$ or $z_B \neq 0 \rightarrow$ finish $tag_B = (T_B \| B_{ID} \| ind \| s \| u_1 \| u_2)$
		$\xleftarrow{M_B}$	$M_B = HMAC_{K_B}(tag_B)$
6.	set $tag = (T_B \| B_{ID} \| ind \| s \| u_1 \| u_2)$ compute $M = HMAC_{KA}(tag)$ if $M \neq M_B$ or $z_A \neq 0 \rightarrow$ finish		

Figure 3.14: SESPAKE Protocol

The identities A_{ID}, B_{ID} are used against an impersonation attack; the flags (z_A, z_B) and assignments ($Q_A = P,\ Q_B = P$) are implemented in constant time to avoid side channel attack. Finally the operations $Q_A = u_2 - Q_{\mathsf{P}}^A$ and $Q_B = u_1 + Q_{\mathsf{P}}$ are performed to avoid impersonation attack.

SESPAKE protocol's security relies on the ROM. V.Smyshlyaev et al. proved the protocol' security under two subjects defined in [54]: a threat of distinguishing a session key from a random string and the decisional version of the false authentication threat.

Moreover, hash functions used in the protocol should be chosen such that $hashlen \leq log_2(q) + 4$ where $hashlen$ denotes the length of output of the hash function.

The points P and $(Q_1, Q_2, \ldots, Q_n)$ should be chosen in the manner that they satisfy provable pseudorandomness. That is, each point Q_i should be generated such that factor of any point under any other point is unknown [48].

CHAPTER 4

ANALYSIS OF PAKE PROTOCOLS

In this chapter, we analyse PAKE Protocols that are mentioned in Chapter 3.

4.1 Analysis of EKE

4.1.1 Number Theoretic Attack on RSA-EKE

Suppose there is an attacker who is called querying attacker in [47] and impersonated A. He generates RSA modulus n and the password encrypted public key X instead of $\mathsf{P}(e)$ without knowing the password P. B receives e from $\mathsf{P}^{-1}(X)$. There is an assumption [47] that we somehow know e is in the form of $e = 3k$ for some value k.

B randomly selects a R and computes $\mathsf{P}(R^e)$ which corresponds to $\mathsf{P}(R^{3k})$. Then, the attacker applies a dictionary attack on $\mathsf{P}'^{-1}(\mathsf{P}(R^{3k})))$ with candidate password P' and tries to obtain $(R^{3k})'$ in order to decide whether this number is a cubic residue. He continues with different candidate password P' unless it is a cubic residue.

Approximately one ninth of the passwords on average gives a matching for cubic residue mod $(n = pq)$, the rest can be eliminated in another sessions [47]. Consequently, it is impossible to reduce valid passwords' space to one at a logarithmic rate.

If e is not in the form of $e = 3k$, the process can be repeated as described above with several random numbers e and different random number X. For example, if the selected e does not have 3 as a factor, it is considered as there is no candidate that creates a cubic residue. Therefore, it is known that e does not have 3 as a factor.

It is repeatedly tried to have a unique password that always creates cubic residues. Average three e to obtain the desired e is needed because one third random e will possess factors of 3.

At this stage, we need to verify that $(R^e)'$ is a cubic residue mod n and it is a cubic residue mod p and mod q. It can be done by selecting p such that $p-1$ has factors of 3 and selected q where $q-1$ has factor of 3. Then, if e possesses a 3 as a factor, the number $\frac{p-1}{3}$ mod p will equal to 1 according to Fermat's theorem that is $(R^e)'^{\frac{p-1}{3}}$ mod $p \equiv (R^{3k})'^{\frac{p-1}{3}}$ mod $p \equiv (R^k)^{p-1}$ mod p [47] .

This number theoretic attack on RSA-EKE can find the password in only tens of sessions. After each $1/9$ of the password in the dictionary, there is still possible valid candidates. Assume that the number W is the length of the dictionary and the number i is an average query such that $(\frac{1}{9})^i W = 1$. [47] claims that three random e can be found in about six sessions where the dictionary size is one million and the password is broken after trying eighteen sessions [47].

For further details about the number theoretic attack and facts of Number Theory, see the paper [47].

4.1.2 Number Theoretic Attack on DH-EKE

Theorem 4.1.1. *[47] Let p be a prime such that $p-1$ has integer d as a factor. The congruence $x^d \equiv a$ mod p has a solution and a is a dth power residue when $a^{\frac{p-1}{d}} \equiv 1$ mod p, else the congruence does not have a solution and a is a dth power non-residue when $a^{\frac{p-1}{d}} \neq 1$ mod p.*

Corollary 4.1.1.1. *[47] The number of dth power residues mod p is equal to $\frac{p-1}{d}$.*

An attacker who does not have the password P can impersonate A by sending his g, p and a random value X instead of sending P(g^{R_A} mod p). After B receives message, he randomly generates a number R_B and computes (g^{R_B} mod p). Finally, he encrypts it with password P and then sends the result P(g^{R_B} mod p) to A.

Although the parameters R_B, g^{R_B} mod p and P(g^{R_B} mod p) are random values, [47] claims the attacker can achieve the password under the condition that the numbers g and p are reasonable choices.

To illustrate, the attacker sends g^d and p in which d is a low prime and a factor of $p-1$. Now, B computes $a = (g^d)^{R_B} \bmod p$ which equals to $g^{R_B d} \bmod p$. It is clear that, the number a is a dth power residue and according to Theorem 4.1.1, we have $a^{\frac{p-1}{d}} \equiv 1 \bmod p$ if d is a factor of $p-1$. Also, $(g^{R_B d})^{\frac{p-1}{d}} \equiv (g^{R_B})^{p-1} \bmod p$ and $(g^{R_B})^{p-1} \equiv 1 \bmod p$ by Fermat's theorem.

In the number theoretic attack, the attacker gets encrypted password $\mathsf{P}(g^{R_B d} \bmod p)$, struggles to decrypt it with different candidate passwords and enhances the decrypted number to $\frac{p-1}{d} \bmod p$. If it does not result in 1, the candidate password is refused.

Corollary 4.1.1.1 implies that $\frac{p-1}{d}$ numbers out of $p-1$ numbers will be dth power residues so $\frac{1}{d}$ numbers will be equivalent to 1 mod p if they are power to $\frac{p-1}{d}$. Note that, the candidate password space is decreased to $\frac{1}{d}$ and the valid password space narrows to one logarithmically [47].

4.1.3 Number Theoretic Attack on the ElGamal-EKE

An attacker, who pretends A, sends the values g^d, p and X instead of the actual message $\mathsf{P}(g_A^R \bmod p)$. Consider that $p-1$ has a factor d like in the theoretic attack on the DH-EKE. When B receives the values, he computes $((g^d)^k \bmod p)$ and decrypts X from $Y = \mathsf{P}^{-1}(X)$. Moreover, he computes $(RY^k \bmod p)$ and produces $\mathsf{P}(g^{kd} \bmod p,\ RY^k \bmod p)$ then sends it to the attacker. He uses different candidate passwords P' in order to decrypt the message received from B. If the candidate password P' is the valid password then the equality $(g^{kd})^{\frac{p-1}{d}} \equiv 1 \bmod p$ is always true. Hence, the candidate P' which does not satisfy this equality can be refused. In particular, this process continues with logarithmic rate until the valid password space is narrowed to one [47].

4.2 Analysis of SPEKE

Firstly, we present "the exponential-equivalence attack" proposed in [71]. This attack is based on exponential relation of two passwords. To illustrate, assume there are two distinct passwords P and P' such that $\mathsf{P}' = \mathsf{P}^r \bmod p$ for an arbitrary integer $r \neq 1$. If there is an exponential relation among all the passwords, the attacker can exclude two passwords. However, if the password is in PIN form, this attack may be disputable. In

order to avoid this problem, Zhang [71] suggested to using the hash of the password instead of applying square operation on it. That is, the password mapping function can be used as $f(\mathsf{P}) = (H(\mathsf{P}))^2 \bmod p$. Hence, it becomes hard for an attacker to find passwords that are exponentially related to hashed passwords.

There are three proposals for attack on SPEKE illustrated by Tang and Mitchell [57]. The first attack is known as Unknown Key-Share (UKS) attack and it is based on sharing the password with multiple servers. To apply this attack, the computation of g needs to include identity of the server, but this requirement damages the symmetry of the protocol. The second attack is about swapping of two sessions. The attacker swaps two messages of two sessions and causes the parties to confuse which message belongs to which session at the end of the protocol. The last attack is actually the same with Zhang's attack [71]. The only difference is that they offer to use the generator g as a hash of the identities with the password i.e $g = H(\mathsf{P}\|A\|B)$ where the values A and B are identities of the parties. Moreover, this attack does not provide the symmetrical feature of the protocol.

Secondly, we present a new attack which is described in [29]: "the impersonation attack". The attacker E impersonates B. Here, instead of B, E communicates with A in parallel sessions in order to convince A that B is the one who she is in communication with even though B is not in the communication at all. The steps of attack are given below and in Figure 4.1:

1. A selects a secret exponent value x and calculates $X = g^x \bmod p$, then sends X to B with identity value A.

2. E receives all messages instead of B. She selects an exponent value z where $X^z \in \{2, ..., p-2\}^2$ then starts the parallel session by transmitting her identity value B and X^z to A.

3. A continues with the second session, selects an exponent value y and calculates $Y = g^y \bmod p$. Then, she transmits Y with identity value A.

4. E gets the message and calculates $Y^z = (g^y)^z \bmod p$. Afterwards, E sends Y^z to A with the value B in the first session.

5. A calculates the key $K = H((Y^z)^x) = H(g^{xyz})$ and the key confirmation parameter $\mathsf{H}(\mathsf{H}(K))$ then transmits it to E.

6. From the first session, E receives the key confirmation parameter of A and sends it back to A in the second session.

7. A sends the challenge value $\mathsf{H}(K)$.

8. At the last stage, E receives A's message from the second session and responds it in the first session in order to complete key confirmation process.

A		E
session 1		
select $x \in \mathbb{Z}_q^*$		select z
compute $X = g^x \bmod p$	$\xrightarrow{1.(A,X)}$	
compute $K = (Y^z)^x)$	$\xleftarrow{4.(B,Y^z)}$	
start key confirmation: $\mathsf{H}(\mathsf{H}H(K))$	$\xrightarrow{5.(\mathsf{H}(\mathsf{H}(K))}$	
verify key confirmation	$\xleftarrow{8.\mathsf{H}(K)}$	
session 2		
select $y \in \mathbb{Z}_q^*$	$\xleftarrow{2.(B,X^z)}$	
compute $Y = g^y \bmod p$	$\xrightarrow{3.(A,Y)}$	compute $Y^z = (g^y)^z \bmod p$
verify key confirmation	$\xleftarrow{6.H(H(K))}$	receive key confirmation parameter from first session and send back
reply key confirmation	$\xrightarrow{7.H(K)}$	

Figure 4.1: Impersonation attack on SPEKE [29]

"The impersonation attack" indicates that SPEKE protocol has a serious vulnerability in the authentication process. The protocol needs to include the key confirmation message in the first session in order to protect itself from this attack. However, the standards ISO 11770-4 [1] and IEEE P1363 [26] allow parties to initiate the key confirmation process in any order [29].

Finally, the paper [29] presents a new attack "the key malleability attack" which is based on the man-in-the-middle:

- A selects a secret exponent value $x \in Z_q^*$, computes $X = g^x \bmod p$ and sends

it with the identity A. Likewise, B selects a secret exponent value $y \in Z_q^*$, computes $Y = g^y \bmod p$ and sends (B, Y).

- E, who intercepts the message pairs (A, X) and (B, Y), selects an arbitrary number z and generates $Y^z \bmod p$ and $X^z \bmod p$. Then, he respectively sends the messages Y^z and X^z to A and B.

- Hence, the parties generates the common session key $K = H(g^{xyz})$ without realizing that the messages are changed.

A		E		B
select $x \in \mathbb{Z}_q^*$				$y \in \mathbb{Z}_q^*$
compute $X = g^x \bmod p$	$\xrightarrow{(A,X)}$		$\xleftarrow{(B,Y)}$	compute $Y = g^y \bmod p$
		select z		
		compute $Y^z \bmod p$		
	$\xleftarrow{(B,Y^z)}$	compute $X^z \bmod p$	$\xrightarrow{(A,X^z)}$	
generate $K = H(g^{xyz})$				generate $K = H(g^{xyz})$

Figure 4.2: Key-malleability attack on SPEKE [29]

Figure 4.2 indicates key malleability attack on SPEKE. The original paper of SPEKE protocol claims that security of the protocol is subject to Decisional Diffie-Hellman (DDH) or Computational Diffie-Hellman (CDH). However, "key malleability attack" shows that DDH and CDH are not sufficient to satisfy the security of authentication. These assumptions require independent secret values on the exponent. For instance, suppose the output of arbitrary function $f(.)$ can be in the form of $z = f(g^x, g^y)$; therefore, a possible correlation between the exponents makes DDH and CHD assumption inapplicable.

4.3 Analysis of SPAKE

The security of SPAKE protocol is based on new variants of DH assumption which are introduced by Abdalla and Pointcheval in [3]. Abdalla and Pointcheval explained security of variants SPAKE1 and SPAKE2 through theorems. For details of the variants of DH assumption and relations between them, please see the paper [3].

Before the theorems, we need to state definition *advantage function* and Lemma 4.3.1.

Suppose there is an adversary $\mathcal{A}$ in the key exchange protocol P that can access to queries of $Reveal$, $Execute$, $Send$ and $Test$ defined in [3]. Let $SUCC$ be an event where the adversary $\mathcal{A}$ is said to be successful. The *ake-advantage* of an $\mathcal{A}$ in violating semantic security of the protocol P is

$$Adv^{ake}_{P,D}(\mathcal{A}) = 2 \cdot Pr[SUCC] - 1$$

and the *advantage function* of the protocol P when passwords are received from dictionary D with time complexity at most t and using resources at most R is

$$Adv^{ake}_{P,D}(t, R) = max_{\mathcal{A}}\{Adv^{ake}_{P,D}(\mathcal{A})\}$$

Lemma 4.3.1. *[3] Assume $\mathbb{G} = (G, g, p)$ is a represented group, n and s are integers, and also assume $\mathcal{P}$ is a public injective map from $\{1, \ldots, n\}$ into Z_p.*

$$Adv^{s-pccdh}_{\mathbb{G},n,s}(t, \mathcal{P}) \geq \frac{1}{n} + \epsilon \Longrightarrow Adv^{cdh}_{\mathbb{G}}(t') \geq \frac{n^2\epsilon^6}{2^{14}} - \frac{2s^4}{p},$$

in which $t' = 4t + (18 + 2s)\tau$ and τ expresses the time for an exponentiation in $\mathbb{G}$. Apparently,

$$Adv^{s-pccdh}_{\mathbb{G},n,s}(t, \mathcal{P}) \geq \frac{1}{n} + \epsilon \geq \frac{1}{n} \times \left(1 + \frac{8(ns)^{2/3}}{p^{1/6}}\right) \Longrightarrow Adv^{cdh}_{\mathbb{G}}(t') \geq \frac{n^2\epsilon^6}{2^{15}}.$$

Theorem 4.3.2. *[3] Assume $\mathbb{G}$ is a represent group and D is a uniformly distributed dictionary of size $|D|$. Then, for any numbers t, q_{start}, q^A_{send}, q^B_{send}, q_H, q_{exe}, with SPAKE1 as given in Figure 3.7*

$$Adv^{ake}_{SPAKE,D}(t, q_{start}, q^A_{send}, q^B_{send}, q_H, q_{exe})$$

$$\leq 2.(q^A_{send} + q^B_{send}.\ Adv^{s-pccdh}_{\mathbb{G},|D|,qH}(t', \mathcal{P}) +$$

$$2.\left(\frac{(q_{exe} + q_{send})^2}{2p} + q_H\ Adv^{cdh}_{\mathbb{G}}(t + 2q_{exe}\tau + 3\tau)\right)$$

where q_H represents the number of queries to the H oracle; q_{exe} represents the number of queries to the Execute oracle; q_{start} and q^A_{send} represent the number of queries to the Send oracle with respect to the initiator A; q^B_{send} represents the number of queries to the Send oracle with respect to the responder B; $q_{send} = q^A_{send} + q^B_{send}+, q_{start}$; $t' = t + O(q_{start}\tau)$; and τ is the time to compute one exponentiation in $\mathbb{G}$.

According to Theorem 4.3.2, the SPAKE1 protocol is secure in the random oracle model relying on S-PCCDH problem which is hard in G.

Corollary 4.3.2.1. *[3] Assume $\mathbb{G}$ is a represent group and $\mathcal{D}$ be is uniformly distributed dictionary of size $|\mathcal{D}|$. Then, for any numbers t, q_{start}, q^A_{send}, q^B_{send}, q_H, q_{exe}, with SPAKE1 as given in Figure 3.7*

$$Adv^{ake}_{SPAKE,\mathcal{D}}(t, q_{start}, q^A_{send}, q^B_{send}, q_H, q_{exe})$$
$$\leq 2 \cdot \left(\frac{q^A_{send} + q^B_{send}}{|\mathcal{D}|} + \sqrt[6]{\frac{2^{14}}{|\mathcal{D}|^2} Adv^{cdh}_{\mathbb{G}}(t') + \frac{2^{15} q_H^4}{|\mathcal{D}|^2 p}} \right) +$$
$$2 \cdot \left(\frac{(q_{exe} + q_{send})^2}{2p} + q_H Adv^{cdh}_{\mathbb{G}}(t + 2q_{exe}\tau + 3\tau) \right),$$

where $t' = 4t + O((q_{start} + q_H\tau)$ and the other parameters are defined as in Theorem 4.3.2.

Corollary 4.3.2.1 indicates a relation between S-PCCDH and CDH problems. Therefore, [3] also says SPAKE1 is a secure PAKE protocol under assumption that CDH problem is hard in G.

Theorem 4.3.3. *[3] Assume $\mathbb{G}$ is a represent group and $\mathcal{D}$ is a uniformly distributed dictionary of size $|\mathcal{D}|$. Then, for any numbers t, q_{start}, q^A_{send}, q^B_{send}, q_H, q_{exe}, with SPAKE2 as given in Section 3.4.*

$$Adv^{ake}_{SPAK2E,\mathcal{D}}(t, q_{start}, q^A_{send}, q^B_{send}, q_H, q_{exe})$$
$$\leq 2 \cdot \left(\frac{q^A_{send} + q^B_{send}}{n} + \frac{(q_{exe} + q_{send})^2}{2p} \right) +$$
$$2 \cdot \left(q_H Adv^{cdh}_{\mathbb{G}}(t + 2q_{exe}\tau + 3\tau) + q_H^2 Adv^{cdh}_{\mathbb{G}}(t + 3\tau) \right),$$

where the parameters are defined in Theorem 4.3.2.

In addition, Theorem 4.3.3 says that the SPAKE2 protocol is secure in the random oracle model provided that CDH problem is hard in G.

4.4 Analysis of SRP

The SRP protocol eliminates need of sending password in the clear text form and provides security against passive and active attacks. In the protocol, S stores the

password as a password file; therefore, attacker can not easily find the password as a password even if he can discover password database. Since he still needs to perform an expensive dictionary attack in order to find out passwords. The attacker can not carry out a man-in-the-middle attack unless he has C's password. The attacker who does not know the values x (private key) or v (verifer) can not fool S or C, so man-in-the-middle attack fails.

The paper [63] also claims that the protocol resists Denning-Sacro Attack [18] in which an eavesdropper captures the session key K and tries to masquerade as C or perform a brute force attack to password of C. The values M_1 and M_2 can be generated from publicly known data and K; therefore, the eavesdropper can not discover new information by using the values M_1, M_2 and K.

The SRP protocol does not have a proof model for security. Recently, a formal analysis of SRP has been introduced by T.Sherman et al. [51]. In [51], SRP is analysed by using Cryptographic Protocol Shapes Analyzer (CPSA) and this analysis found that the structure of protocol does not have a major weakness such as leakage of secrets.

However, T.Sherman also emphasize that SRP is vulnerable to pre-computation dictionary attack since the salt value s is transmitted publicly.

Moreover, T.Sherman suggested a first attack on SRP which depends on leakage of verifier. In this suggestion, attacker who has the values v and b can masquerade as client since being comprised of the server makes attacker learn these values.

4.4.1 Use of SRP Protocol for TLS

In TLS protocol, public key certificates Kerberos or preshared keys are used to provide authentication. However, these methods remain incapable when using in the TLS. In this section, we briefly describe how SRP authentication method is used for TLS. Especially, we mention a new handshake message process which is described in [58] using the SRP authentication without going into detail.

Let the clientname, the salt, public key of C and public key of S be expressed as C, s, A and B respectively.

Handshake Message Flow for SRP authentication

	C		S
1.	"client hello (clientname)"	$\longrightarrow$	
2.			"server hello"
			certificate*
			server key exchange (N, g, s, B)
		$\longleftarrow$	"server hello done"
3.	client key exchange (A)	$\longrightarrow$	
4.	finished	$\longrightarrow$	
5.		$\longleftarrow$	finished
6.	application data	$\longleftrightarrow$	application data

Figure 4.3: Handshake Message Flow for SRP authentication

Figure 4.3 indicates the handshake message flow for SRP authentication. Certificate part of flow is indicated by (*) that means that it is an optional or situation dependent message.

Before proceeding to handshake message flow, we give details about the client name extension which is referred as "SRP extension".

4.4.2 Content of Handshake Message

4.4.2.1 SRP extension

Firstly, if a C decides resumption of session which utilizes SRP authentication, the SRP extension must be contained in the client hello message. If current session is agreed to continue, the information the client hello message's SRP extension must be disregarded by S, but exception is that if it is involved in the finished message hashes.

4.4.2.2 Server Certificate and Key Exchange

When C agrees with S to use an SRP cipher suite, S must transmit a certificate which supplies a further authentication with of a digital signature. Which cipher suites are allowed are described in [58].

The server key exchange message consists of parameters (N, g, s, B) where N and g are group parameters such that N is a safe prime of the form $N = 2q + 1$ (q is also a prime) and q is a generator of group. Public value of S B is generated from the equation $B = kv + g^b$ mod N and the salt s came from SRP password file that is received in the "client hello message".

The valid SRP group parameters are given [58] and C should accept valid parameters to ensure security. If the parameters which are sent from S are not valid, C must abort handshake process.

4.4.2.3 Client Key Exchange

The client key exchange message contains public key A of C that is calculated from $A = g^a \bmod N$ and it must be transmitted before the server key exchange message.

4.4.3 The Pre-master Secret

The pre-master secret is result of the shared secret S from the calculations.

C generates the pre-master secret as follows [58]:

$clientname$, P (read from C)
p, g, s, B (read from S)
a = random()
$A = g^a \bmod p$
u = H (PAD(A) || PAD(B))
k = H (p || PAD(g))
x = H (S || H($clientname$ || ":" || P))
premaster secret = $(B - (kg^x)^{(a+(ux))}) \bmod p$

Figure 4.4: Pre-master key from server [58]

The S generates the pre-master secret as follows:

p, g, s, v (read from password files)
b = random()
k = H (p || PAD(g))
$B = k^v + g^b \bmod p$
A = read from C u = H (PAD(A) || PAD(B))

Figure 4.5: Pre-master key from client [58]

The finished messages carry out the same function with messages M_1 and M_2 given in [61]. The parties abort the process if one of them calculates an incorrect pre-master secret.

4.4.4 New Message Contents

To structure the new message transmitted during a handshake, the value "srp" is defined as the extension number for the extensions in both "the server hello message" and "the client hello message" [58].

In the client hello message, the clientname C is encoded by using an extension of "srp" as described in Figure 4.6.

Similarly, in the extended server hello message, the parameters (N, g, s) are encoded by using an extension of type "srp".

```
enum {client, server} ClientOrServerExtension;

struct {
  select(ClientOrServerExtension) {
    case client :
      opaque srp_U < 1..2^8 − 1 >;
    case server :
      opaque srp_S < 1..2^8 − 1 >;
      opaque srp_N < 1..2^16 − 1 >;
      opaque srp_g < 1..2^16 − 1 >;
  }
} SRPExtension;
```

Figure 4.6: Client hello message [58]

4.4.4.1 Client Key Exchange

C sends the ephemeral key A in the key exchange message that is encoded in a $ClientSRPPublic$ structure if the value of $KeyExchangeAlgorithm$ is adjusted to "srp" and it is shown in Figure 4.7:

```
struct {
  select (KeyExchangeAlgorithm) {
     case rsa: EncryptedPreMasterSecret ;
     case diffie_hellman: ClientDiffieHellmanPublic;
     case srp: ClientSRPPublic;    /* new entry */
   } exchange_keys;
} ClientKeyExchange;
struct {
    opaque srp_A < 1..2^16 − 1 >;
} ClientSRPublic;
```

Figure 4.7: Client key exhange message [58]

4.4.4.2 Server Key Exchange

S sends the ephemeral public key B in the key exchange message that is encoded in a $ServerSRPPublic$ structure if the value of $KeyExchangeAlgorithm$ is adjusted to "srp". The encoded message is given in Figure 4.8.

To provide security, C's private key a and S's private key b should be chosen to be at least 256-bit numbers. Also, the parties should make sure that the prime parameter N is large enough to supply infeasibility of discrete logarithms. The server key exchange is described in Figure 4.8.

```
enum  {rsa, diffie_hellman, srp} KeyExchangeAlgorithm;

struct {
   select (KeyExchangeAlgorithm) {
       case diffie_hellman:
           ServerDHParams params;
           Signature signed_params;
        case rsa:
           ServerRSAParams params;
           Signature signed_params;
        case srp:     /* new entry */
           ServerSRPParams params;
           Signature signed_params;
   };
} ServerKeyExchange;

 struct {
    opaque srp_N < 1..2^16 - 1 >;
    opaque srp_g < 1..2^16 - 1 >;
    opaque srp_s < 1..2^8 - 1 >;
    opaque srp_B < 1..2^16 - 1 >;
} ServerSRPParams;    /* SRP paramaters */
```

Figure 4.8: Server key exhange message [58]

Because the attacker may try to contact to S and to find out a C's password, the general network communications restrict the authentication enterprise from a specific C or a specific IP address.

Additionally, the "client hello message" contains the clientname of C in the clear form. To prevent this, C should open a server-authenticated connection and then perform SRP-authenticated connection.

4.5 Analysis of AugPake

As stated in the Section 3.6, AugPAKE protocol promises security against passive attacks, active attacks and offline dictionary attacks and also resists server compromise. In this section, we explain that how the AugPake prevents these attacks [52].

The AugPake protocol provides security against "passive attacks" since an eavesdropper can not generate an authenticated session key even if he obtains the exchanged messages. Suppose that the eavesdropper can attain the messages (CI, X), (SI, Y), V_u and V_s and he tries to compute the session key S_k. That is, he tries to find the accurate session key S_k from the values X and Y as the hash functions are secure. Consider the following two equations where $t = w'r \bmod q$

$$X = g^x \ mod \ p$$

$$Y = (XW^r)^y = X^y \, W^{ry} = X^y \, (g^y)^t = X^y K^t$$

The number t is specified from possible passwords with the value X and the attacker can only derive the session key K from X and Y by computing X^y. But the parameters x and y are selected randomly from $\mathbb{Z}_q^*$, so it is assumed that the probability of computing X^y has a negligible rate. For this reason, we can say that AugPAKE protects itself from passive attacks [52].

The AugPAKE protocol provides security against "active attacks" because the attacker can not generate an authenticated session key even if he obtains the exchanged messages. That is, the probability of computing the session key by an active attack is bounded by the online dictionary attacks and it depends on number of interactions with the honest party linearly. Note that, C (or S) calculates the session key S_k if the authenticator value V_s (or V_u) is valid. Three situations where active attacks can be performed in are described below [52]:

- The attacker can masquerade as C by computing the session key S_k under the condition that V_u is valid. Suppose that the authenticator V_u is a valid value, then the attacker needs to calculate the correct K from the numbers X and Y due to of secure hash functions. The attacker has the discrete logarithm x of X and from the password dictionary, he finds a password w''. However, he can

obtain the correct K only if $w = w''$. Hence, the AugPAKE protocol prevents impersonation attacks on C because of restriction of impersonation attack from the online dictionary attack.

- The attacker can also masquerade as S by computing the same session key S_k if the authenticator V_s is the correct value. As in the previous attack, the attacker needs to calculate valid K from the parameters X and Y due to secure hash functions. Firstly, he selects a random number y and from the password dictionary, he predicts a password w'' then computes Y from the equality

$$Y = (X.W'^{r})^{y} = X^{y}.W'^{r} = X^{y}.(g^{y})^{t}$$

  where ($t = w''.r$ mod q). Computation of the correct K's probability is again restricted by the probability of $w = w''$. Moreover, the attacker can check if w'' equals to w' by using the authenticator V_u. Yet, the probability of computing correct K is negligible because the attacker needs to find the discrete logarithm x. Thus, this attack is bounded by the online dictionary attacks and it implies that the AugPAKE protocol provides security against impersonation attacks on S.

- The attacker can perform the man-in-the-middle attack and produce the same session key S_k, if the authenticator V_u or V_s is valid, by transmitting the exchanged messages. He needs to calculate the correct K from the values X and Y in order to produce a valid authenticator V_u or V_s so the conditions that are described above are also valid for the man-in-the-middle attack.

The AugPAKE protocol provides security against "offline dictionary attacks" because the attacker who can control the exchanged messages can not narrow possible password candidates as well as in online dictionary attacks. As mentioned before, the derivation of K is independent from the guessed password; thus, a passive attacker can not compute X^{y} and $K = g^{y}$ mod p from the received messages X and Y even if he can guess a password. Additionally, suppose that there is an active attacker and he performs the man-in-the-middle attack is previously described. However, the important point of AugPAKE protocol shows that if authentication fails, execution of protocol is terminated and no further message is sent. In this way, the protocol pre-

vents testing more than one password. Consequently, both active and passive attacks can not reduce the possible password candidates easier than online dictionary attacks.

The paper [52] presents two limitations of AugPAKE protocol: Firstly, the password and its verifier have the same entropy, thus the attacker can obtain the correct password from the verifier by performing offline dictionary attacks. Secondly, the attacker can obtain sufficient information in order to impersonate S so the impersonation of S with the verifier is a an efficient and possible attack. On the other hand, the attacker can not masquerade as C without applying offline dictionary attacks on the password verifier W in the protocol.

Suppose that there is an attacker with the W and struggles to masquerade as C without applying offline dictionary attacks on W. At first, he selects two random numbers c and d such that $c \in \mathbb{Z}_q^*$ and $d \in \mathbb{Z}_q^*$ then computes ($X = g^c W^d \bmod p$) and sends the first message (CI, X) to S. The attacker needs to generate correct ($K = g^y \bmod p$) in order to impersonate C. That is, he tries to get a number e which yields ($K = Y^e \bmod p$).

$$log_g Y^e = log_g K \ \ mod\ q$$

$$(c + (w'd) + (w'r))\, ye = y \ \ mod\ q$$

$$(c + w'(d + r))\, e = 1 \ \ mod\ q$$

From above equation ($e = \dfrac{1}{c} \bmod q$) and ($d = -r \bmod q$). Note that, the attacker does not apply offline dictionary attacks on W. Moreover, he can not find out the value r from the equation $r = \mathsf{H}'(0x01 \| CI \| SI \| M(X))$ because of a secure hash function. Hence, AugPAKE resists the server compromise [52].

4.6 Analysis of J-PAKE

The authentication and key confirmation in two round J-PAKE protocol (or three-pass variant) are implicit [56]. The two parties can utilize derived key to authenticate each other at the beginning of communication. Moreover, they can decrypt and read messages if both have the same derived session key.

To make an explicit authenticaiton, two parties should perform an additional key confirmation that provides certain assurance about having derived the same key [56] .

There are many methods to attain explicit key confirmation. We explain two examples of these methods as follow:

The first method is based on the key confirmation of *SPEKE Protocol* [34] (See Section 3.2). Suppose that, A starts the key confirmation by sending $\mathsf{H}(\mathsf{H}(k'))$ to B and then B verifies $\mathsf{H}(\mathsf{H}(k'))$. If the verification is completed successfully, B sends $\mathsf{H}(k')$ to A and then A verifies $\mathsf{H}(k')$.

Key confirmation scheme which is given in NIST SP 800-56A Revision 1 is base for the second method. [6]. In this method, A and B send MAC tag to each other a and verify it accordingly. Let k' = KDF (K $\|$ "JPAKE - KC").

The method works in the finite field as follow :

- A sends MacTagA = MAC (k', "KC-1-U " $\|$ A $\|$ B $\|$ g_1 $\|$ g_2 $\|$ g_3 $\|$ g_4) to B .
- B sends MacTagB = MAC (k', "KC-1-U " $\|$ B $\|$ A $\|$ g_3 $\|$ g_4 $\|$ g_1 $\|$ g_2) to A.

The method works in the elliptic curve as follow :

- MacTagA = MAC (k', "KC-1-U " $\|$ A $\|$ B $\|$ Gr_1 $\|$ Gr_2 $\|$ Gr_3 $\|$ Gr_4).
- MacTagA = MAC (k', "KC-1-U " $\|$ A $\|$ B $\|$ Gr_1 $\|$ Gr_2 $\|$ Gr_3 $\|$ Gr_4).

k' is the recommended key that is different from the session key and it provides staying indistinctive from random for the session key after the key confirmation process [6].

The explicit key confirmation ensures an explicit and immediate confirmation. As mentioned in Chapter 1, a secure PAKE protocol should satisfy properties such as online and offline dictionary attack resistance, security of session key and forward secrecy. It has been proven in [30] that all these security requirements are satisfied by J-PAKE protocol based on assumptions of difficulty of the Decisional Diffie-Hellman problem and security of the Schnorr NIZK proof [28].

To sum up, J-PAKE protocol may be considered more useful PAKE protocol in comparison with EKE, SPEKE or SRP-6 (See Section 3.1, 3.2, 3.5 respectively.). Because

besides having security proofs, it has quite different design approach from other protocols [30] . Also, J-PAKE protocol works in both finite field and elliptic curve. It is not required that usage of additional base to hash passwords to an assigned elliptic curve. Moreoever, real-world applications like Firefox, Pale Moon and Google Nest products have been used J-PAKE protocol. Some open source libraries such as OpenSS, Network Security Services (NSS) and the Boundary Castle include J-PAKE protocol.

4.7 Analysis of OPAQUE

As mentioned before, OPAQUE protocol does not rely on PKI and does not use plaintext password in the clear form at S where the other aPAKE protocols do not satisfy these properties.

The attacker can learn the password by only observing a session at S and applying an exhaustive offline dictionary attack. Thus, OPAQUE can resist pre-computation attacks.

In additon to these fundamental properties, OPAQUE can utilize different key-exchange protocols in practice and provide different performance tradeoffs. It also supports a unique functionality so that it allows to store and retrieve C's secrets [37]. OPAQUE further promotes password hardening so that the cost of offline dictionary attacks is increased.

Additionally, [37] states the security proof of the protocol in random oracles and demonstrates how OPAQUE protocol can be used with TLS 1.3:

The general mechanism for password authentication of TLS relies on Public Key Infrastructure and it causes disclosure of passwords to S while TLS decryption. The integrating OPAQUE with TLS supplies aims to eliminate this vulnerability and protects TLS from PKI failures. In particular, the integrating OPAQUE with TLS 1.3 is straightforward if we ignore the protection of C account information [37].

In TLS mechanism, C's private key u_1 is attained from S as a signature key for TLS authentication and the Diffie-Hellman key exhange is reused.

- The first message of the integrated protocol is TLS's "$client\ hello$" message which is enhanced with account information of C and the first message of DH-OPRF that is a.
- S replies to the second regular TLS 1.3 message with "$server\ hello$" and the second message of DH-OPRF that consists of the values b, v and Env_u. Also, S sends the private key s_1 along with public key s_2 for TLS signature.
- Finally, C sends a general "$client\ finished$" message and authenticates himself by using C signature which is generated from private key s_1 and verified by public key s_2.

To protect C's account information, the account information is attached to the first handshake message and its encryption is done under the pre-shared key. Then, the protocol continues as described above.

If there is no continuous session or pre-shared key, C account protection needs to use server certificate. Then, the TLS 1.3 handshake is enhanced by using the two OPAQUE messages that are operated alternately between second and third part of the ordinary TLS handshake.

4.8 Overview

In this section, we review PAKE protocols which are stated in this thesis. We emphasize important points of them and give an overview in tables. Also, we include computational costs of them.

Firstly, we summarize balanced PAKE protocols and present their major properties in Table 4.1 and Table 4.2.

EKE is a first PAKE protocol which uses public key and symmetric key cryptography together. It aims to prevent offline dictionary attacks and ensure forward secrecy. Protocol's security arises from discrete logarithm problem of DH in a cyclic group but there is no formal security proof model for it. EKE has three broken variants which are RSA-EKE, DH-EKE and ElGamal EKE. In Table 4.1, we give an overview for DH-EKE because it has been basis for other protocols like SPEKE and B-SPEKE.

SPEKE is a balanced PAKE protocol that depends on computational assumptions of DDH and CDH. SPEKE's security proof was given in random oracle model but last known attacks which are "the exponential-equivalence attack" [71], "the impersonation attack" and "the key-malleability attack" [29] were performed in 2018 and they show that SPEKE can not satisfy forward secrecy. Furthermore, The SPEKE protocol has been currently used in Entrust's TruePass end-to-end web products and Blackberry phones [29].

SPAKE aims to decrease usage of random oracle in the security proof. It is easier to implement than other two-party PAKE protocols since it does not require ideal ciphers onto a group or full domain hash functions. However, it does not satisfy forward secrecy. It has two variants named as SPAKE1 and SPAKE2 and we include SPAKE1 in Table 4.1.

J-PAKE can work over a finite field and elliptic curves. The difference between J-PAKE and other PAKE protocols is that the parties send it to each other zero knowledge proofs of Diffie-Hellman parameters. It ensures security for online and offline dictionary attacks and satisfies forward secrecy. Furthermore, it was included in standard of ISO 11770-4 [1].

4.1: Balanced PAKE Proposals Overview

	Computational assumption	Proof model	Known attack	Standard
DH-EKE	DH	✗	[47]	✗
SPEKE	DDH, CDH	ROM	[29], [71]	ISO 11770-4 [1] IEEE P1363 [26]
SPAKE1	CDH	ROM	✗	✗
J-PAKE	DDH SNP [28]	ROM	✗	ISO 11770-4 [1]
PAK	DDH	ROM	✗	ITU [2], IETF [15] IEEE P1363 [26]
SESPAKE	CDH	ROM	✗	Russian Federation

♠ SNP* : Schnorr NIZK Proof

PAK is an improved version of DH-EKE protocol which provides explicit authentication while resisting passive and active attacks. It was standardized in ITU [2], IETF [15] and IEEE P1363 [26]. There are many variants of PAK protocols such as PPK, PAK-X [13] and PAK-EC, PAK-R, PAK-Y [45]. We give PPK protocol that is more efficient than PAK protocol since it supplies implicit authentication in two rounds.

SESPAKE consists of a key agreement and a key confirmation stage and works over elliptic curves. The purpose of SESPAKE is to avoid impersonation attacks. Moreover, it is confirmed in the standardization system of the Russian Federation.

4.2: Balanced PAKE Proposals Computational Costs

	Exponentiation (C)	Exponentiation (S)	Total
DH-EKE	2	2	4
SPEKE	2	2	4
SPAKE1	4	4	8
J-PAKE	4	4	8
PAK	3	2	5

Lastly, we give a summary for augmented PAKE protocols and state major properties of them in Table 4.3 and Table 4.4.

B-SPEKE is an extension of SPEKE protocol so that DH is used to demonstrate knowledge of the password instead of digital signature. It aims to provide security against an eavesdropper and a man-in-the-middle attack. The security of B-SPEKE depends on difficulty of discrete logarithm problem of DH.

SRP is known as zero-knowledge proof protocol in which S does not need to store hashed version of the password. It ensures forward secrecy and offers resistance to dictionary attacks, a man-in-the-middle attack and Denning-Sacro Attack [18]. It has no security proof model but there is a formal analysis of SRP which is presented by [51] in 2020. Additionally, it was standardized as a TLS cipher suit and included in IEEE P1363 standard [26]. AugPAKE relies on difficulty of DH assumptions and

4.3: Augmented PAKE Proposals Overview

	Computational assumption	Proof model	Known attack	Standard
B-SPEKE	DH	✗	✗	✗
SRP	✗	✗	✗	IEEE P1363 [26]
AugPAKE	DH	ROM	✗	ISO 11770-4 [1]
OPAQUE	DH-OPRF	ROM	✗	✗

its security is proven in random oracles. It resists server compromise and dictionary attacks. Also, the standard ISO 11770-4 [1] includes AugPake protocol.

OPAQUE is the first PKI-free augmented protocol that remains secure against pre-computation attacks while being able to use a secret salt and it satisfies forward secrecy with explicit authentication. The protocol consists of OPRF protocol in which DH-OPRF is used as a function and as key exchange protocol. Also, [37] states OPAQUE and TLS 1.3 can be combined and shows examples of possible schemes.

4.4: Augmented PAKE Proposals Computational Costs

	Exponentiation (C)	Exponentiation (S)	Total
B-SPEKE	3	4	7
SRP	3	3	6
AugPAKE	2	2	4
OPAQUE	*	*	*

*OPAQUE's computational cost depends on the the OPRF function's cost, a regular DH exchange's cost and cost of key exchange protocol which is determined to use.

CHAPTER 5

SUMMARY

5.1 Current Directions

Most of PAKE protocols use Diffie-Hellman key exchange as a basis in their algorithms, therefore they also rely on discrete logarithm problem. However, the security of these PAKE protocols has become controversial recently with the development of quantum computers.

To illustrate, X.Gao et al. [24] presented a post-quantum version of SRP protocol which is based on Ring-Learning-with-Errors (RLWE) problem and called this protocol as RLWE-SRP. They showed that 209-bit RLWE-SRP's implementation is three times faster than 112-bit original SRP protocol and also five and half times faster than 80-bit J-PAKE protocol.

J.Ding et al. [21] presented two-lattice-based PAKE protocols which are lattice-based version of PAK and PPK protocols. It is believed that PAK and PPK can be conveniently replaced by these protocols, which are based on RLWE problem, in post-quantum world. The new lattice-based RLWE-PAK protocol that is similar with PAK consists of three phase and supplies mutual explicit authentication, whereas RLWE-PPK protocol that is similar with PPK consists of two phase and ensures implicit authentication.

D.Xu et al. [64] introduced the first lattice based three-party PAKE protocol and called this new protocol as RLWE-3PAKE. They use RLWE-PAK protocol as a basis in their protocol and implemented it by using LatticeCrypto. They analyzed performance of RLWE-3PAKE protocol and also proved the security of the protocol in

random oracle model.

Additionally, there is another three-party PAKE protocol proposed by R.Choi et al. [17] which is named as *AtLast*. They combined J.Ding et al.'s RLWE-PPK protocol with Abdalla's method and compared their protocol with RLWE-3PAKE protocol.

5.2 Conclusion

In this thesis, we compiled some proposal PAKE protocols and examined their properties with security analysis of them.

Firstly, we gave information about a key exchange and a key agreement protocol. We explained weakness of Diffie-Hellman key exchange and the importance of authentication in key exchange systems. Then, we introduced PAKE protocols and presented main features of them such as methods and type of settings.

Next, we stated algorithms of some proposal PAKE protocols and mentioned their security. We presented attacks on variants of EKE which is "number theoretic attack" and attacks on SPEKE which are "exponential-equivalence attack", "impersonation attack" and "key-malleability attack". Furthermore, we mentioned the usage of SRP and OPAQUE protocols in TLS.

Finally, we summarized features of these protocols by mentioning their important points and gave an overview for balanced and augmented PAKE protocols separately and also showed their computational costs. Current state of the art was included as well.

As already noted, there is formal methods analysis of SRP protocol [51]. As a future work, OPAQUE and J-PAKE protocols may be investigated in terms of formal methods analysis.

REFERENCES

[1] ISO: Information technology — Security techniques — Key management — Part 4: Mechanisms based on weak secrets, ISO/IEC 11770-4, 2nd edn.(2017). International Standard.

[2] ITU: Password-authenticated key exchange (PAK) protocol, ITU-T Rec.X.1035, (2007). ITU-T Recommendation.

[3] M. Abdalla and D. Pointcheval, Simple password-based encrypted key exchange protocols, in *Cryptographers' track at the RSA conference*, pp. 191–208, Springer, 2005.

[4] C. Adams, A. Barg, F. L. Bauer, O. Benoit, E. Biham, A. Biryukov, J. Black, R. Blakley, G. Bleumer, S. Boeyen, et al., *Encyclopedia of Cryptography and Security: A Springer Live Reference*, Springer-Verlag Berlin Heidelberg, 2011.

[5] A. Albarqi, E. Alzaid, F. Al Ghamdi, S. Asiri, J. Kar, et al., Public key infrastructure: a survey, Journal of Information Security, 6(01), p. 31, 2014.

[6] E. Barker, L. Chen, S. Keller, A. Roginsky, A. Vassilev, and R. Davis, Recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography, Technical report, National Institute of Standards and Technology, 2017.

[7] J. Becerra, P. Y. A. Ryan, P. Šala, and M. Škrobot, An offline dictionary attack against zkpake protocol, in G. Dhillon, F. Karlsson, K. Hedström, and A. Zúquete, editors, *ICT Systems Security and Privacy Protection*, pp. 81–90, Springer International Publishing, Cham, 2019, ISBN 978-3-030-22312-0.

[8] M. Bellare, D. Pointcheval, and P. Rogaway, Authenticated key exchange secure against dictionary attacks, in *International conference on the theory and applications of cryptographic techniques*, pp. 139–155, Springer, 2000.

[9] S. M. Bellovin and M. Merritt, Encrypted key exchange: Password-based protocols secure against dictionary attacks, in *Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 72–84, IEEE, 1992.

[10] S. M. Bellovin and M. Merritt, Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise, in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pp. 244–250, 1993.

[11] S. Bhatia and R. Saraswat, A survey on three-party password-based authenticated key exchange (3-pake) protocols.

[12] C. Boyd, A. Mathuria, and D. Stebila, *Protocols for authentication and key establishment*, volume 1, Springer, 2003.

[13] V. Boyko, P. MacKenzie, and S. Patel, Provably secure password-authenticated key exchange using diffie-hellman, in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 156–171, Springer, 2000.

[14] J. G. Brainard, A. Juels, B. Kaliski, and M. Szydlo, A new two-server approach for authentication with short secrets., in *USENIX Security Symposium*, pp. 201–214, 2003.

[15] A. Brusilovsky, I. Faynberg, Z. Zeltsan, and S. Patel, Password-authenticated key (pak) diffie-hellman exchange, RFC 5683, 2010.

[16] C.-C. Chang and Y.-F. Chang, A novel three-party encrypted key exchange protocol, Computer Standards & Interfaces, 26(5), pp. 471–476, 2004.

[17] R. Choi, H. An, and K. Kim, Atlast: Another three-party lattice-based pake scheme, in *2018 Symposium on Cryptography and Information Security (SCIS 2018)*, IEICE Technical Committee on Information Security, 2018.

[18] D. E. Denning and G. M. Sacco, Timestamps in key distribution protocols, Communications of the ACM, 24(8), pp. 533–536, 1981.

[19] M. Di Raimondo and R. Gennaro, Provably secure threshold password-authenticated key exchange, in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 507–523, Springer, 2003.

[20] W. Diffie and M. Hellman, New directions in cryptography, IEEE transactions on Information Theory, 22(6), pp. 644–654, 1976.

[21] J. Ding, S. Alsayigh, J. Lancrenon, R. Saraswathy, and M. Snook, Provably secure password authenticated key exchange based on rlwe for the post-quantum world, in *Cryptographers' Track at the RSA Conference*, pp. 183–204, Springer, 2017.

[22] Y. Ding and P. Horster, Undetectable on-line password guessing attacks, ACM SIGOPS Operating Systems Review, 29(4), pp. 77–86, 1995.

[23] W. Ford and B. S. Kaliski, Server-assisted generation of a strong secret from a password, in *Proceedings IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2000)*, pp. 176–180, IEEE, 2000.

[24] X. Gao, J. Ding, J. Liu, and L. Li, Post-quantum secure remote password protocol from rlwe problem, in *International Conference on Information Security and Cryptology*, pp. 99–116, Springer, 2017.

[25] L. Gong, M. A. Lomas, R. M. Needham, and J. H. Saltzer, Protecting poorly chosen secrets from guessing attacks, IEEE journal on Selected Areas in Communications, 11(5), pp. 648–656, 1993.

[26] I. P. W. Group et al., Ieee p1363-2: Standard specifications for passwordbased public key cryptographic techniques.

[27] S. Halevi and H. Krawczyk, Public-key cryptography and password protocols, ACM Transactions on Information and System Security (TISSEC), 2(3), pp. 230–268, 1999.

[28] F. Hao, Schnorr non-interactive zero-knowledge proof, 2017.

[29] F. Hao, R. Metere, S. F. Shahandashti, and C. Dong, Analyzing and patching speke in iso/iec, IEEE Transactions on Information Forensics and Security, 13(11), pp. 2844–2855, 2018.

[30] F. Hao and P. Ryan, J-pake: authenticated key exchange without pki, in *Transactions on computational science XI*, pp. 192–206, Springer, 2010.

[31] F. Hao and P. Y. Ryan, Password authenticated key exchange by juggling, in *International Workshop on Security Protocols*, pp. 159–171, Springer, 2008.

[32] L. Harn, M. Mehta, and W.-J. Hsin, Integrating diffie-hellman key exchange into the digital signature algorithm (dsa), IEEE Communications Letters, 8(3), pp. 198–200, 2004.

[33] V.-H. Hoang, E. Lehtihet, and Y. Ghamri-Doudane, Password-based authenticated key exchange based on signcryption for the internet of things, in *2019 Wireless Days (WD)*, pp. 1–8, IEEE, 2019.

[34] D. P. Jablon, Strong password-only authenticated key exchange, ACM SIGCOMM Computer Communication Review, 26(5), pp. 5–26, 1996.

[35] D. P. Jablon, Extended password key exchange protocols immune to dictionary attack, in *Proceedings of IEEE 6th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 248–255, IEEE, 1997.

[36] D. P. Jablon, Password authentication using multiple servers, in *Cryptographers' Track at the RSA Conference*, pp. 344–360, Springer, 2001.

[37] S. Jarecki, H. Krawczyk, and J. Xu, Opaque: an asymmetric pake protocol secure against pre-computation attacks, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 456–486, Springer, 2018.

[38] B. Kaliski, Pkcs# 5: Password-based cryptography specification version 2.0, Technical report, RFC 2898, september, 2000.

[39] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, Chapman & Hall/CRC, 2nd edition, 2014, ISBN 1466570261, 9781466570269.

[40] J. Katz, P. MacKenzie, G. Taban, and V. Gligor, Two-server password-only authenticated key exchange, in *International Conference on Applied Cryptography and Network Security*, pp. 1–16, Springer, 2005.

[41] D. H. Krawczyk, The opaque asymmetric pake protocol, Technical report, Internet-Draft draft-krawczyk-cfrg-opaque-01. Work in Progress. Internet ..., 2018.

[42] N. Li, Research on diffie-hellman key exchange protocol, in *2010 2nd International Conference on Computer Engineering and Technology*, volume 4, pp. V4–634, IEEE, 2010.

[43] C.-L. Lin, H.-M. Sun, and T. Hwang, Three-party encrypted key exchange: attacks and a solution, ACM SIGOPS Operating Systems Review, 34(4), pp. 12–20, 2000.

[44] R. Lu and Z. Cao, Simple three-party key exchange protocol, Computers & Security, 26(1), pp. 94–97, 2007.

[45] P. Mackenzie, More efficient password-authenticated key exchange, in *Cryptographers' Track at the RSA Conference*, pp. 361–377, Springer, 2001.

[46] K. Mochetti, A. Resende, and D. Aranha, zkpake: a simple augmented pake protocol, in *Brazilian Symposium on Information and Computational Systems Security (SBSeg)*, volume 6, 2015.

[47] S. Patel, Number theoretic attacks on secure password schemes, in *Proceedings. 1997 IEEE Symposium on Security and Privacy (Cat. No. 97CB36097)*, pp. 236–247, IEEE, 1997.

[48] A. K. E. S. Protocol, The security evaluated standardized password authenticated key exchange (sespake) protocol, 2016.

[49] O. Ruan, J. Chen, and M. Zhang, Provably leakage-resilient password-based authenticated key exchange in the standard model, IEEE Access, 5, pp. 26832–26841, 2017.

[50] N. Shafnamol and K. S. Krishna, Signature-based multi-server pake protocol, in *2017 International Conference on Networks & Advances in Computational Technologies (NetACT)*, pp. 310–313, IEEE, 2017.

[51] A. T. Sherman, E. Lanus, M. Liskov, E. Zieglar, R. Chang, E. Golaszewski, R. Wnuk-Fink, C. J. Bonyadi, M. Yaksetig, and I. Blumenfeld, Formal methods analysis of the secure remote password protocol, arXiv preprint arXiv:2003.07421, 2020.

[52] S. Shin and K. Kobara, Efficient augmented password-only authentication and key exchange for ikev2, IETF RFC 6628, Experimental, 2012.

[53] S. Shin and K. Kobara, Security analysis of password-authenticated key retrieval, IEEE Transactions on dependable and secure computing, 14(5), pp. 573–576, 2015.

[54] S. Smyshlyaev, I. B. Oshkin, E. K. Alekseev, and L. R. Ahmetzyanova, On the security of one password authenticated key exchange protocol., IACR Cryptology ePrint Archive, 2015, p. 1237, 2015.

[55] M. Steiner, G. Tsudik, and M. Waidner, Refinement and extension of encrypted key exchange, ACM SIGOPS Operating Systems Review, 29(3), pp. 22–30, 1995.

[56] D. R. Stinson, *Cryptography: theory and practice*, Chapman and Hall/CRC, 2005.

[57] Q. Tang and C. J. Mitchell, On the security of some password-based key agreement schemes, in *International Conference on Computational and Information Science*, pp. 149–154, Springer, 2005.

[58] D. Taylor, T. Wu, N. Mavrogiannopoulos, and T. Perrin, Using the secure remote password (srp) protocol for tls authentication, Request for Comments, 5054, 2007.

[59] A. Tsohou, S. Kokolakis, C. Lambrinoudakis, and S. Gritzalis, Information systems security management: a review and a classification of the iso standards, in *International Conference on e-Democracy*, pp. 220–235, Springer, 2009.

[60] Wikipedia, Password-authenticated key agreement, `https://en.wikipedia.org/wiki/Password-authenticated_key_agreement`, (Accessed: 2020-05-01).

[61] T. Wu, The srp authentication and key exchange system, Technical report, RFC 2945, September, 2000.

[62] T. D. Wu, A real-world analysis of kerberos password security., in *Ndss*, 1999.

[63] T. D. Wu et al., The secure remote password protocol., in *NDSS*, volume 98, pp. 97–111, Citeseer, 1998.

[64] D. Xu, D. He, K.-K. R. Choo, and J. Chen, Provably secure three-party password authenticated key exchange protocol based on ring learning with error., IACR Cryptology ePrint Archive, 2017, p. 360, 2017.

[65] X. Yi, F. Hao, and E. Bertino, Id-based two-server password-authenticated key exchange, in *European Symposium on Research in Computer Security*, pp. 257–276, Springer, 2014.

[66] X. Yi, F.-Y. Rao, Z. Tari, F. Hao, E. Bertino, I. Khalil, and A. Y. Zomaya, Id2s password-authenticated key exchange protocols, IEEE Transactions on Computers, 65(12), pp. 3687–3701, 2016.

[67] X. Yi, R. Tso, and E. Okamoto, Id-based group password-authenticated key exchange, in *International Workshop on Security*, pp. 192–211, Springer, 2009.

[68] X. Yi, R. Tso, and E. Okamoto, Identity-based password-authenticated key exchange for client/server model., SECRYPT, 12, pp. 45–54, 2012.

[69] S. Yuanyuan and L. Wengang, Password-based authenticated key exchange protocols, in *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, volume 2, pp. V2–212, IEEE, 2010.

[70] L. Zhang and Z. Zhang, Security analysis of an id-based two-server password-authenticated key exchange, IEEE Communications Letters, 21(2), pp. 302–305, 2016.

[71] M. Zhang, Analysis of the speke password-authenticated key exchange protocol, IEEE Communications Letters, 8(1), pp. 63–65, 2004.

[72] Z. Zhao, Z. Dong, and Y. Wang, Security analysis of a password-based authentication protocol proposed to ieee 1363, Theoretical Computer Science, 352(1-3), pp. 280–287, 2006.