AN ABSTRACT OF THE DISSERTATION OF

Minho Kim for the degree of Doctor of Philosophy in Electrical and Computer Engineering presented on August 4, 2006.

Title: Cryptanalysis and Enhancement of Authentication Protocols .

Abstract approved: _____

Çetin Kaya Koç

Authentication protocols play important roles in network security. A variety of authentication protocols ranging from complex public-key cryptosystems to simple password-based authentication schemes have been proposed. However, currently there is no fully secure authentication scheme that can resist all known attacks. When a user authentication is performed over an insecure network, additional problems arise due to the fact that the communication may be intercepted, or even altered, by an attacker. In general, one cannot assume that there is a secure channel between the client and the server. In this dissertation, we present specific cryptanalytic attacks on existing protocols and show their vulnerabilities in order to design more secure protocols. In particular, we propose improved security schemes to overcome certain security defects with registration, login, and password/identifier-change schemes. We also propose new authentications schemes which are more secure against guessing, stolen-verifier, replay, denial-of-service, and impersonation attacks than the existing protocols.

Cryptanalysis and Enhancement of Authentication Protocols

by

Minho Kim

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented August 4, 2006
Commencement June 2007

Doctor of Philosophy dissertation of Minho Kim presented on August 4, 2006.

APPROVED:

Major Professor, representing Electrical and Computer Engineering

---

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Minho Kim, Author

ACKNOWLEDGMENTS

Thank God for guiding me to get through this journey.

I wish to express my gratitude to Professor Çetin Kaya Koç, my advisor, for his encouragement, insightful guidance and patience throughout my Ph.D. degree at OSU. I would like to thank Professor Ben Lee for his valuable feedback and professional teaching. I also want to extend great appreciation to Professor Zhongfeng Wang, Traylor Roger, Thomas Schmidt, and Greg Baker for their serving in my graduate committee.

I would like to thanks my colleagues in Information Security Laboratory, Dr. Gokay Saldamli, Dr. Lo'ai Tawalbeh, Onur Aciicmez, Weetit Wanalertlak, and Wen-Chun Yang that they were always good mentors, general supports and being such good friends. In particular, I would like to thank Gerald Lai and Phuc Vo for helping me to understand English and fruitful discussions. Warm thanks go to Electrical and Computer Engineering staff especially Ferne Simendinger for her generous helps whenever I needed.

I want to thank Korean Air Force and Air Force Academy for giving chance to study and fully supporting me in USA. In special, I thank to go Faculty Board in KAFA and Dept. of Computer Science and Statistics for patience about my studies for the last three years.

Finally, I express my deepest appreciation and love to my deceased father, SeongYun Kim, and my mother, HyunIm Jeong, who have been encouraging, caring, and praying for my whole life. I dedicate my dissertation to my mother. I would like to acknowledge my parents-in-law ByungGyun Yoo and SoonRye Hwang for their care and patience. I also express my thanks to my brothers BongHo and KyungHo, a younger sister and brother of my wife BokHee and HeeJeong in Korea

for their love, care, and prayers. In special, I would like to express my deepest gratitude to my wife, BokRye Yoo who has shown endless patience and love. She supported me fully with victimizing her important period in her life. I thank very much my daughter, SeoHee Kim and my son, Kijin Kim for well-breeding with health and happiness.

I Love God.

Kim, Minho

Corvallis, Oregon, August 2006

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

To my parents SeongYun Kim and HyunIm Jeong,

my parents-in-law ByungGyun Yoo and SoonRye Hwang,

my wife BokRye Yoo,

my daughter SeoHee, and my son KiJin,

my brothers BongHo and KyungHo,

a younger sister and brother of my wife BokHee and HeeJeong,

and the Almighty God.

CRYPTANALYSIS AND ENHANCEMENT OF AUTHENTICATION PROTOCOLS

CHAPTER 1. INTRODUCTION

As society has developed, the need for more sophisticated methods of protecting data has been increased. History is filled with examples where people tried to keep information secret from their adversaries. In the battlefield, a commander communicated with their troops using simple cryptographic methods to prevent the enemy from learning sensitive military information. For various reasons, people have always tried to keep information away from others. As the world becomes more convenient, the demand for information and the need for good service are greater than ever.

Using a password, a user can create and send a valid login message to a remote system in order to gain access. When a user receives a message, he wants to be sure that it has been created recently and for a particular purpose by the principal who claims to have sent it. The user must be able to detect when a message has been created or modified by a malicious principal or attacker with access to the network or when a message was issued for a different purpose and is currently being replayed on the network. A variety of authentication protocols ranging from complex public-key cryptosystems to simple password-based authentication schemes have been proposed. An authentication protocol is a sequence of message exchanges between principals that either distributes secrets to some of those principals or allows the

use of some secrets to be recognized. Due to its simplicity, the most common form of user authentication is password-based. That is, a user must be able to provide a password in order to be authenticated.

Password-based authentication mechanisms are the simplest and most convenient way to authenticate a user in order to provide services of a computing or communication system to a pre-selected group of authorized users. These mechanisms are less costly than biometric methods of authentication, such as fingerprint, iris scan, voice signature, etc. In conventional password authentication schemes, a network user not only needs to log into various remote servers with repetitive registration, but also needs to remember the various user identities and passwords. Large passwords are more cryptographically secure but are also more cumbersome to remember. This problem can be circumvented by using passwords that are small and easy to remember with keys that are large and can be recorded. However, users often choose weak passwords in order to be able to remember them easily. These weak passwords are potentially susceptible to dictionary attacks. Dictionary attacks are brute force attacks on passwords that are performed by testing a large number of likely passwords against some publicly available information about the desired password. Strong symmetric keys need 60 bits or more, and nobody talks about memorizing public-keys. It is also fair to assume that a memorizable password belongs to a brute-force searchable space. To counter the threat of attack, we assign the user painfully large passwords and force frequent password changes. Another problem in using the traditional password authentication method is that a server must maintain a password table that stores each users ID and password. Therefore, the server requires extra memory space to store the password table. Many researchers have attempted to create a remote authentication scheme without using passwords. With ever-increasing computing power, there is a growing

gap between the size of the smallest safe key and the size of the largest easily remembered password. Unfortunately, most commonly-known remote password methods require a large password.

When user authentication is performed over an insecure network, additional problems arise due to the fact that the communication may be intercepted, or even altered, by an attacker. In general, one cannot assume that there is a secure channel between the client and the server. A considerable number of authentication protocols have been specified and implemented. The area is, however, remarkably subtle and many protocols have been shown to be flawed a long time after they were published. Recently, several user authentication protocols have been introduced and subsequently attacked [1–39]. Many of them can be classified into three types: ElGamal-based, RSA-based, and Hash-based password authentications. A generic password-based authentication system usually hashes the password of the user with the help of a hash function derived from a secret-key cryptographic function, such as MISTY, DES, or FEAL [40–42]. The hashed password is stored on the server in order to prevent the adversary from stealing the password. Unfortunately, there are two known limitations in password-based authentication systems: 1) the user must submit the bare password at every authentication, and 2) the transmitted password could be stolen by wiretapping or sniffing. Because of these limitations, password-based authentication must deal with password guessing, replay, impersonation (forgery), stolen-verifier, denial-of-service (DoS), and man-in-the-middle attacks, etc.

Another area that needs improvement is the payment protocol authentication. One of our works discusses weakness in the payment protocol authentication. There exist several payment protocols for electronic commerce [43–49]. The most commonly used protocols for Internet e-commerce are based on credit card charges

over the Secure Sockets Layer (SSL). Such schemes require the merchant to perform an online credit check through a process hidden from the user. The cost of a check is around 10 cents, which is expensive for low-value transactions. A secure and user-friendly solution for micropayments can be built to provide a cost-efficient and effective infrastructure for creating a network of interoperable payment service providers. Such a solution offers the following: 1) facilitation of independent operators of the system for interoperation, thus enabling the operators to reach out rapidly to a critical mass of consumers and merchants, 2) multi-channel access to different devices, 3) an open and extendible platform for developing multiple payment applications, and 4) lower operational costs and automated dispute resolution. Generally, micropayment systems collect the accumulated amount of money as one regular payment either before or after the transactions. It is well suited as a charging mechanism for public transportation systems, access control to sites and services, content sales (music, video, software, etc.), subscriptions, and "pay-per-view or -click" Web services for small amounts of money called "microcents". Since it is not practical for individual users to charge small amounts of money, such as a penny or a fraction of a penny, to a major charge card, a different method of payment is needed for sites that wish to go "micro".

Although many micropayment systems already exist, none has obtained a dominant market position. This is because customers and providers would have to install and use multiple systems; something that is undesirable to both parties. Several methods of micropayment collection are being examined, many of which involve the encoding of per-fee-links inside HTML pages and an Internet wallet account. Through this account individuals would establish a cash balance with a third-party application that would monitor, collect, and distribute micropayments. A central requirement for any electronic payment system is that a compromise or

failure should not have tragic consequences. For example, it should not be possible to double spend in a digital cash system. Nor should the compromise of a client's authorization secret entail unlimited client liability or uncollectible transactions.

The purpose of this dissertation is to describe specific cryptanalytic attacks on existing protocols and show their vulnerabilities in order to design more secure protocols. Particularly, we propose improved security schemes to overcome certain security defects with registration, login, and password/identifier-change schemes. We then propose new authentications schemes which are more secure against guessing, stolen-verifier, replay, denial-of-service, and impersonation attacks than the existing protocols.

The organization of this dissertation is as follows: we first introduce the necessary background, such as authentication schemes, attacks against protocols, and related works. In chapter 2-5, we then show our works that are published and accepted in journals: A Simple Attack on a Recently Introduced Hash-Based Strong-Password Authentication Scheme [50]; A Simple Attack on a Recently Introduced Hash-Based Secure User Authentication Scheme [37]; Vulnerabilities in the Adachi-Aoki-Komano-Ohta Micropayment Scheme [51]; and Enhanced Security for the Modified Authenticated Key Agreement Scheme [52]. In chapter 6-9, we present our works which are in processing at various journals: Two Simple Attacks on the ID-based Password Authentication Scheme Using Smart Cards and Fingerprints [53]; Security Improvements on a New User Remote Authentication Scheme [54]; Improving the Novikov and Kiselev User Authentication Scheme [55]; and A Secure Hash-Based Strong-Password Authentication Protocol Using One-Time Public-Key Cryptography [56]. Finally, we conclude our work with final comments in chapter 10.

1.1. Authentication Scheme

The whole point of cryptography is to keep the plaintext (or the key, or both) secret from the adversary (also called eavesdropper, attacker, intruder, enemy, etc.). The adversary is assumed to have complete access to the communications between the sender and the receiver. While real-world cryptanalysts do not always have such detailed information, it is a good assumption to make. If others cannot break an algorithm, even with knowledge of how it works, then they certainly will not be able to break it without that knowledge. *Cryptanalysis* is the art and science of recovering the plaintext of a message without access to the key. Successful cryptanalysis may recover the plaintext or the key. It also may find weaknesses in a cryptosystem that eventually leads to the previous results. The loss of a key through noncryptanalytic means is called a *compromise* and an attempted cryptanalysis is called an *attack*.

1.1.1. Basic Authentication Scheme

Authentication is the process of reliably verifying the identity of someone (or something). There are lots of examples of authentication in human interaction. A mail order company might accept as authentication the fact that we know the expiration date on our credit card. When A logs into a host computer (ATM, telephone banking system, or other type of database management systems, etc.), how does the host know who he is? How does the host know he is not malicious attacker trying to falsify A's identity?

In the context of an HTTP transaction, the basic authentication scheme is

a method designed to allow a web browser, or other client program, to provide credentials - in the form of a user name and password - when making a request. Although the scheme is easily implemented, it relies on the assumption that the connection between the client and server computers is secure and can be trusted. Specifically, the credentials are passed as plaintext and could be intercepted easily. The scheme also provides no protection for the information passed back from the server.

One advantage of the basic authentication scheme is that it is supported by almost all popular web browsers. It is used on normal Internet web sites but may sometimes be used by small, private systems. A later mechanism, digest access authentication, was developed in order to replace the basic authentication scheme and enable credentials to be passed in a relatively secure manner over an otherwise insecure channel.

A typical transaction between an HTTP client and an HTTP server running on the local machine is comprised of the following steps:

- The client asks for a page that requires authentication but does not provide a user name and password. Typically this is because the user simply entered the address or followed a link to the page.

- The server responds with the response code and provides the authentication realm. At this point, the client will present the authentication realm (typically a description of the computer or system being accessed) to the user and prompt for a user name and password.

- The user may decide to cancel at this point. However, once a user name and password have been supplied, the client re-sends the same request but includes the authentication header. In this example, the server accepts the

authentication, and the page is returned. If the user name is invalid or the password incorrect, the server might return the response code and the client would prompt the user again.

- A client may pre-emptively send the authentication header in its first request, with no user interaction required.

The basic authentication scheme was originally defined by RFC 1945 (Hypertext Transfer Protocol HTTP/1.0) [57] although further information regarding security issues may be found in RFC 2616 (Hypertext Transfer Protocol HTTP/1.1) [58] and RFC 2617 (HTTP Authentication: Basic and Digest Access Authentication) [59].

1.1.2. Authentication in Secret Key Cryptography

Authentication means that someone can prove knowledge of a secret without revealing it. Strong authentication is possible with cryptography and is particularly useful when two computers are trying to communicate over an insecure network. Suppose, user A and user B share a key K_{AB}, and they want to verify that they are talking to each other. They each choose a random number, called a *challenge*. A chooses R_A and B chooses R_B. The value X encrypted with K_{AB} is known as the *response* to the challenge X. If the adversary impersonates A, he could get B's information to encrypt a value for him, however, this information would not be useful later in impersonating B to the real A because the real A would choose a different challenge. If A and B complete this exchange, they have each proven to the other party that they know K_{AB} without revealing it to the attacker. In this protocol, there is an opportunity for the adversary to obtain some pairs

(chosen plaintext, ciphertext), since he can insist to be B and request A to encrypt a challenge for him. This is the general idea of a cryptographic authentication algorithm, though this algorithm has a subtle problem that would prevent it from being useful in most computer-to-computer cases.

1.1.3. Authentication in Public Key Cryptography

In secret key cryptography, if people want to talk secretly, they have to share a secret key. If A wants to be able to prove his ID to lots of places, then he should remember a secret key for each place to which he would like to prove his ID. He could use the same shared secret with B as with the third party, but that has the disadvantage that the third party and B could impersonate A to each other. Public key technology is more convenient than secret key technology. A only needs to remember his own private key. If A wants to be able to verify his identity in many places, he does not need to remember a lot of keys. He only needs to know the public keys. For example, A chooses a random number R and encrypts it using B's public key $PubK_B$, and sends the result to B. B proves he knows B's private key, $PriK_B$, by decrypting the message and sending R back to A. Thus, in public key cryptography, no one needs to keep any secret information for verification purposes.

1.1.4. User Authentication Protocol

Authentication is done somewhat differently depending on the capabilities of the thing being authenticated. The two most important capabilities are the ability

to store a high-quality cryptographic key and the ability to perform cryptographic operations. A computer has both of these capabilities; a person has neither of them. Humans are incapable of securely storing high-quality cryptographic keys, and have unacceptable speed and accuracy when performing cryptographic operations.

User authentication consists of a computer verifying that we are who we claim to be. There are three main techniques: *"what we know", "what we have"*, and *"what we are"*. Passwords provide a method of reassuring someone that we are who we claim to be, and fit into the "what we know" category. Physical keys or ATM cards fit into the "what we have" category, and biometric devices are fit into the "what we are" category.

- Password Authentication

  In order to establish communications over a PPP (Point-to-Point Protocol), each end of the PPP link must first send LCP (Link Control Protocol) packets to configure the data link during Link Establishment phase. After the link has been established, PPP provides for an optional Authentication phase before proceeding to the network-layer protocol phase. By default, authentication is not mandatory. If authentication of the link is desired, an implementation must specify the authentication-protocol configuration option during link establishment phase. These authentication protocols are intended for use primarily by hosts and routers that connect to a PPP network server via switched circuits or dial-up lines, but might be applied to dedicated links as well. The server can use the identification of the connecting host or router in the selection of options for network layer negotiations.

  PAP(Password Authentication Protocol) is an authentication protocol that requires users to enter a password before accessing a secure system. The

user's name and password are sent over the wire to a server, where they are compared with a database of user account names and passwords. This technique is vulnerable to wiretapping (sniffing) because the password can be captured and used by someone to log on to the system. PAP provides a simple method for the peer to establish its identity using a two-way handshake. This is done only upon initial link establishment. After the link establishment phase is complete, an ID/Password pair is repeatedly sent by the peer to the authenticator until authentication is acknowledged or the connection is terminated.

PAP is not a strong authentication method. Passwords are sent over the circuit "in the clear", and there is no protection from playback or repeated trial and error attacks. The peer is in control of the frequency and timing of the attempts. Any implementations which include a stronger authentication method must offer to negotiate that method prior to PAP. This authentication method is most appropriately used where a plaintext password must be available to simulate a login at a remote host. In such use, this method provides a similar level of security to the usual user login at the remote host. PAP is not recommended in most cases. However, some authentication systems will fall back to PAP if no better authentication scheme is available. CHAP (Challenge Handshake Authentication Protocol) is an alternative protocol that avoids sending passwords in any form over the wire by using a challenge/response technique, as described under CHAP [60].

- Authentication Token

  An authentication token is a physical device that a person carries around and uses in authenticating and fall in the "what we have" category. There are

several forms of authentication token in use today. The most ubiquitous is the key that people use to unlock their home or car. Another common form of authentication token is the credit card. If a credit card includes a picture or a signature, it combines an authentication token with a primitive biometric device. Credit cards contain a magnetic strip that contains information. The advantages that magnetic strip cards offer over simple passwords is that they are hard to reproduce and can conveniently hold a large amount of data that most people are unable to memorize.

A better form of authentication token is the smart card. This is a device about the size of a credit card but with an embedded CPU and memory. When inserted in a smart card reader, the card carries on a conversation with the device. There are various forms of smart cards: Cryptographic calculator (or readerless smart card), Cryptographic challenge/response cards, Pin protected memory card, etc.

- Biometrics

  Biometric devices authenticate us according to "what we are". They measure our physical characteristics and match them against a profile. There are a variety of biometric devices available. All are too expensive to be in everyday use, but is some cases the costs are coming down to where they may become commonplace. The biometric technologies available now includes:

  Face Recognition - Looking at a digitized picture of a person, a computer can measure facial dimensions.

  Iris Scanner - This maps the distinctive layout of the iris of eye. It has the major advantage of having a less intimidating user interface.

  Retinal Scanner - Like an Iris Scanner, this device examines the tiny blood

vessels in the back of eye. The layout is as distinctive as a fingerprint and apparently easier to read. These devices are quite expensive.

Fingerprint Reader - This device would seem an obvious technology since fingerprints have been used as a method of identification for many years.

Handprint Reader - This is more widely used than fingerprint readers. It measures the dimensions of the hand: finger length, width, etc. However, it is not as accurate as fingerprint, but it is less expensive.

Voiceprint - It is possible to do a frequency spectrum analysis of someone's voice and get identification nearly as accurate as a fingerprint. This technology, however, can be defeated with a voice recording, and may refuse to authenticate someone whose voice has changed due to illness.

Signature - This is a classic human form of authentication, and there are human experts adept at determining whether two signatures were produced by the same person. When not just the signature is recorded, but also the actual timing of the movements that go into scribing the signature is recorded, there is sufficient information for authentication.

Keystroke timing checker - The exact way in which people type is quite distinctive, and experiments have been done with identification based on the way people type. However, various injuries can throw off the timing.

1.2. Attacks against Protocols

1.2.1. Passive and Active attacks

Cryptographic attacks can be directed against the cryptographic algorithms used in protocols, against the cryptographic techniques used to implement the algorithms and protocols, or against the protocols themselves. People can try various ways to attack a protocol. Someone not involved in the protocol can eavesdrop on some or all of the protocol. This is called a *passive attack*, because the attacker does not affect the protocol. All he can do is observe the protocol and attempt to gain information. This kind of attack corresponds to a ciphertext only attack. Since passive attacks are difficult to detect, protocols try to prevent passive attacks rather than detect them.

Alternatively, an attacker could try to alter the protocol to his own advantage. He could pretend to be someone else, substitute one message for another, delete existing messages, replay old messages, insert new messages in the protocol, interrupt a communications channel, or alter stored information in a computer. These are called *active attacks*, because they require active intervention. The form of these attacks depends on the network.

Passive attackers try to gain information about the parties involved in the protocol. They collect messages passing among various parties and attempt to cryptanalyze them. Active attacks, on the other hand, can have much more diverse objectives. The attacker could be interested in obtaining information, degrading system performance, corrupting existing information, or gaining unauthorized access to resources. Active attacks are much more serious, especially in protocols in which the different parties do not necessarily trust one another. The attacker

does not have to be a complete outsider. He could be a legitimate system user like the system administrator. There could even be many active attackers working together.

It is also possible that the attacker could be one of the parties involved in the protocol. He may lie during the protocol or not follow the protocol at all. This type of attacker is called a *cheater*. *Passive cheaters* follow the protocol, but try to obtain more information than the protocol intends them to. *Active cheaters* disrupt the protocol in progress in an attempt to cheat. It is very difficult to maintain a protocol's security if most of the parties involved are active cheaters, but sometimes it is possible for legitimate parties to detect that active cheating is going on.

1.2.2. Man-in-the-middle Attack

This attack is an attack in which an attacker is able to read, insert and modify at will, messages between two parties without either party knowing that the link between them has been compromised. The attacker must be able to observe and intercept messages going between the two victims. The man-in-the-middle attack is particularly applicable to the original Diffie-Hellman key exchange protocol when it is used without authentication. Most web applications do not require client-side certificates due to complexity, cost, logistical and effectiveness issues. This creates an opening for a man-in-the-middle attack, particularly for online banking.

The man-in-the-middle attack may include one or more of: 1) eavesdropping, including traffic analysis and possibly a known plaintext attack; 2) chosen ciphertext attack, depending on what the receiver does with a message that it decrypts;

3) replay attacks; 4) denial of service attack. For instance, the attacker may jam all communications before attacking one of the parties. The defense against a man-in-the-middle attack is for both parties to periodically send authenticated status messages and to treat their disappearance with paranoia. Man-in-the-middle is typically used to refer to active manipulation of the messages, rather than passively eavesdropping.

1.2.3. Known-Plaintext Attack

The known-plaintext attack (KPA) is an attack model for cryptanalysis where the attacker has samples of both the plaintext and its encrypted version (ciphertext) and is at liberty to make use of them to further reveal secret information; such as the secret key. Encrypted file archives such as ZIP are also very prone to this attack. For example, an attacker with an encrypted ZIP file needs only one unencrypted file from the archive which forms the "known-plaintext". Then using some publicly available software they can instantly calculate the key required to decrypt the entire archive. To obtain this unencrypted file the attacker could search the web site for a suitable file, find it from another archive they can open, or manually try to reconstruct a plaintext file armed with the knowledge of the filename from the encrypted archive. Classical ciphers are typically vulnerable to known-plaintext attack. For example, a Caesar cipher can be solved using a single letter of corresponding plaintext and ciphertext to decrypt a ciphertext entirely. In World War II, a known plaintext attack was used during the Sahara campaign.

1.2.4. Chosen-Plaintext Attack

A chosen plaintext attack (CPA) is an attack model for cryptanalysis which presumes that the attacker has the capability to choose arbitrary plaintexts to be encrypted and obtain the corresponding ciphertexts. The goal of the attack is to gain information that will reduce the security of the encryption scheme. In the worst case scenario, a chosen plaintext attack could reveal the scheme's secret key.

Two forms of chosen-plaintext attack can be distinguished: 1) Batch chosen-plaintext attack, where the cryptanalyst chooses all plaintexts before any of them is encrypted. This is often the meaning of an unqualified use of chosen-plaintext attack; 2) Adaptive chosen-plaintext attack, where the cryptanalyst makes a series of interactive queries, choosing subsequent plaintexts based on the information from the previous encryptions. For example, the air traffic controllers or pilots want to identify an airplane as friend or foe. They send a message to the plane, which encrypts the message automatically and sends it back. Only a friendly airplane is assumed to have the correct key. The system compares the message from the plane with the correctly encrypted message. If they are matched, the plane is our side; otherwise it is enemy. Of course, the adversary can send a large number of chosen messages to one of the planes and look at the resulting ciphertexts. If this allows them to deduce the key, the enemy can equip their planes and masquerade as friend.

1.2.5. Replay Attack

This attack is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed. This is carried out either by the originator or by an adversary who intercepts the data and retransmits it, possibly as part of a masquerade attack. Suppose Alice wants to prove her identity to Bob. Bob requests her password as proof of identity, which Alice dutifully provides. Meanwhile, an adversary can eavesdrop their conversation and record the password. After the interchange is over, the adversary connects to Bob pretending to be Alice; when asked for a proof of identity, the adversary can send Alice's password recorded from the earlier session.

1.2.6. Denial-of-Service Attack

A denial-of-service attack (DoS attack) is an attack on a computer system or network that causes a loss of service to users. Typically the loss of network connectivity and services is done by consuming the bandwidth of the victim network or overloading the computational resources of the victim system. A denial-of-service attack can be perpetrated in a number of ways. There are three basic types of attack: 1) consumption of computational resources, such as bandwidth, disk space, or CPU time; 2) disruption of configuration information, such as routing information; 3) disruption of physical network components. Denial of Service attacks can also lead to problems in the network 'branches' around the actual computer being attacked. For example, the bandwidth of a router between the Internet and a LAN may be consumed by a denial-of-service, meaning not only will the

intended computer be compromised, but the entire network will also be disrupted. If the denial-of-service is conducted in a sufficiently large scale, entire geographical swathes of Internet connectivity can also be compromised by incorrectly configured or flimsy network infrastructure equipment without the attacker's knowledge or intent. Attacks can be directed at any network device, including attacks on routing devices and Web, electronic mail, or Domain Name System servers. A "banana attack" is another particular type of DoS. It involves redirecting outgoing messages from the client back onto the client, preventing outside access, as well as flooding the client with the sent packets.

1.2.7. Guessing Attack

One way of guessing passwords is simply to type passwords at the system that is going to verify the password. To thwart such an *on-line attack*, the system can make it impossible to guess too many passwords in this manner. For example, ATM does not return our card if we type three consecutive incorrect passwords. Alternatively, the system can be designed to be slow, so as not to allow very many guesses per unit time. Also, with an on-line attack, the system can become suspicious that someone might be attempting to break in, based on noticing an unusually large number of incorrect passwords. The system might then dispatch a human to investigate. In contrast, in an *off-line attack*, an intruder can capture a quantity X that is derived from a password in a known way. Then the intruder can, in complete privacy, use an arbitrary amount of compute power to guess passwords, convert them in the known way, and see if X is produced. Because a source of good password guesses is from a dictionary, an off-line password guessing

attack is sometimes referred to as a *dictionary attack*.

A *dictionary attack* is a technique for defeating a cipher or authentication mechanism by trying to determine its decryption key or passphrase by searching a large number of possibilities. In contrast with a brute force attack, where all possibilities are searched through exhaustively, a dictionary attack only tries possibilities which are most likely to succeed, typically derived from a list of words in a dictionary. Generally, dictionary attacks succeed because most people have a tendency to choose passwords that are easy to remember, and from their native language. Dictionary attacks may be applied in two main situations: 1) in cryptanalysis, in trying to determine the decryption key for a given piece of ciphertext; 2) in computer security, in trying to circumvent an authentication mechanism for accessing a computer system by guessing passwords. In the latter case, the effectiveness of a dictionary attack can be greatly reduced by limiting the number of authentication attempts that can be performed each minute, and even blocking further attempts after a threshold of failed authentication attempts is reached. Generally, three attempts are considered sufficient to cope with mistakes made by legitimate users; beyond that, one can safely assume that the user is a malicious attacker.

A *brute force attack* is a method of defeating a cryptographic scheme by trying all possibilities. In most schemes, the theoretical possibility of a brute force attack is recognized, but it is set up in such a way that it would be computationally infeasible to carry out. Accordingly, one definition of "breaking" a cryptographic scheme is to find a method faster than a brute force attack. The selection of an appropriate key length depends on the practical feasibility of performing a brute force attack. For symmetric-key ciphers, a brute force attack typically means a brute-force search of the key space; that is, testing all possible keys in order to

recover the plaintext used to produce a particular ciphertext. In a brute force attack, the expected number of trials before the correct key is found is equal to half the size of the key space. For each trial of a candidate key the attacker needs to be able to recognize when he has found the correct key. The most straightforward way is to obtain a few corresponding plaintext and ciphertext pairs, that is, a known-plaintext attack. Alternatively, a ciphertext-only attack is possible by decrypting ciphertext using each candidate key, and testing the result for similarity to plaintext language. Symmetric ciphers with keys of length up to 64 bits have been broken by brute force attacks. DES, a widely-used block cipher which uses 56-bit keys, was broken by custom hardware in 1998 [61], and a message encrypted with RC5 using a 64-bit key was broken more recently by Distributed.net [62]. In addition, it is commonly speculated that government intelligence agencies can successfully attack a symmetric key cipher with long key lengths, such as a 64-bit key, using brute force. For applications requiring long term security, 128 bits is currently thought of as a sufficient key length for new systems using symmetric key algorithms. NIST has recommended that 80-bit designs be phased out by 2015 [63].

If keys are generated in a weak way, it is possible to exhaustively search for the key over a much smaller set, like words from a dictionary. For instance, a guessing attack consists of the attacker guessing a value g, and then verifying that guess in some way. The verification will be by the intruder using g to produce a value v, called the verifier. The verifier demonstrates that the guess was correct, i.e. an incorrect guess would not have led to this value. This verification can take a number of different forms: 1) the attacker knew v initially, or has seen v during the protocol run; 2) the attacker produced v in two different ways; or 3) v is an asymmetric key, and the attackers knows the inverse of v from somewhere [64].

1.2.8. Impersonation (Forgery) Attack

This attack deceives the identity of one of the legitimate parties. An attacker inserts a message and claims that it came from a legitimate sender. For example, in an electronic funds transfer service, if the attacker manages to register his public key as the public key of a bank account holder, he can digitally sign fraudulent fund transfer requests. This attack creates a requirement for key management in the case of public key cryptography. With public key cryptography, the goal of key management is to authenticate the public key ownership and associate a public key with the identity of the corresponding private key holder. Moreover, this attack threatens the certification authority itself. Although the potential damages are significant, the small number of certification authorities makes preventive measures easier to implement. A successful, intelligent impersonation attack requires knowledge about the probe algorithm and the communication protocol in order to bypass or defeat the checking mechanism.

1.2.9. Stolen-Verifier Attack

In the stolen-verifier attack, an attacker who has stolen the password-verifier from the server uses it directly to masquerade as a legitimate user and may further mount a guessing attack on it. The main purpose of an authentication scheme against the stolen-verifier attack is to reduce the direct damage to user authentication. The server stores users' passwords verifiers instead of the clear text of passwords.

1.2.10. Forward/Backward Secrecy

A protocol is said to have perfect forward secrecy if it is impossible for the adversary to decrypt a conversation between A and B even if adversary records the entire encrypted session, and then subsequently breaks into both A and B and steals their long-term secrets. Forward secrecy has been used as a synonym for perfect forward secrecy [65], since the term perfect has been controversial in this context. Forward secrecy means that if a long-term private key like a user password or server private key is compromised after a given session, it does not compromise any earlier sessions. However, at least one reference [66] distinguishes perfect forward secrecy from forward secrecy with the additional property that an agreed key will not be compromised even if agreed keys derived from the same long-term keying material in a subsequent runs are compromised.

The trick to achieving perfect forward secrecy is to generate a temporary session key not derivable from information stored at the node, and discard this key after the session concludes. If the session will last for a long time, it is common to generate and forget keys periodically so that even if the adversary seizes A's and B's computers while the conversation is still going on, he will not be able to decrypt messages received before the last key rollover.

- Forward Secrecy guarantees that a passive adversary who knows a contiguous subset of old group keys cannot discover subsequent group keys.
- Backward Secrecy guarantees that a passive adversary who knows a contiguous subset of group keys cannot discover preceding group keys.
- Weak Forward Secrecy guarantees that new keys must remain out of reach of former group members.

- Weak Backward Secrecy guarantees that previously used group keys must not be discovered by new group members.

1.3. Related Works

1.3.1. ElGamal-based Password Authentication Schemes

It is possible to design a system whose security relies on the difficulty of computing discrete logarithms. This was done by ElGamal in 1985 [67]. The ElGamal scheme can also be used to create both encryption scheme and digital signature.

ElGamal encryption scheme is as follows. A wants to send a message m to B. B chooses a large prime p and a primitive root α. Assume m is an integer with $0 \leq m < p$. If m is larger, break it into smaller blocks. B also chooses a secret integer a and computes $\beta \equiv \alpha^a \ (mod\ p)$. The information (p, α, β) is made public and is B's public key. A does the following:

- Receives (p, α, β),
- Selects a secret random integer k,
- Computes $r \equiv \alpha^k (mod\ p)$ and $t \equiv \beta^k * m (mod\ p)$,
- Sends the pair (r, t) to B.

B decrypts by computing $tr^{-a} \equiv m (mod\ p)$, where
$tr^{-a} \equiv \beta^k * m(\alpha^k)^{-a} \equiv (\alpha^a)^k * m * \alpha^{-ak} \equiv m (mod\ p)$.

ElGamal digital signature scheme is as follows. B wants to sign a message m. He chooses a large prime p and a primitive root α. B then selects a secret integer a such that $1 \leq a \leq p-2$ and computes $\beta \equiv \alpha^a (mod\ p)$. These values of p, α, and β are made public. The security of the system will be in the fact that a is kept private. It is difficult for an adversary to determine a from (p, α, β) since the discrete log problem is considered difficult. In order for B to sign a message m, he does the following:

- Selects a secret random integer k such that $gcd(k, p-1) = 1$,
- Computes $r \equiv \alpha^k (mod\ p)$ and $s \equiv k^{-1}(m - ar)(mod\ (p-1))$, where

  $sk \equiv (m - ar)(mod\ (p-1)), m \equiv sk + ar(mod\ (p-1))$, and a is a B's private key.
- (r, s) is B's signature on m, and then B sends m, r, and s to A.

A can verify the signature that is declared valid if and only if $g_1 \equiv g_2 (mod\ p)$, where $g_1 \equiv \beta^r * r^s (mod\ p)$ and

$g_2 \equiv \alpha^m (mod\ p) \equiv \alpha^{sk+ar} \equiv (\alpha^a)^r (\alpha^k)^s \equiv \beta^r * r^s (mod\ p)$.

Hwang et al. [2] proposed that a new remote user authentication scheme using smart cards that password or verification tables were not required for verifying legal user. It is based on ElGamal's public key cryptosystem. The server only keeps the secret key for computing the user passwords. Moreover, the Hwang et al. scheme [2] is able to resist the message replay attack. However, [1, 14, 15, 19] showed that the Hwang et al. scheme [2] cannot resist against the forgery attack. Awasthi et al. [14] and Shen et al. [19] proposed an improved scheme to resist against the forgery attack. Later, Leung et al. [17] showed that the Shen et al.

scheme [19] is not able to withstand forgery attack by using an attack similar to that of Chan et al. [1] or Chang [15]: a legitimate user is able to still impersonate other legal users to login a remote server. In 2004, Kumar [27] proposed that the idea of check digits to repair the Shen et als scheme, which removes the threats devised by Leung et al. [17]. In the same year, Awasthi et al. [68] also proposed a scheme to against these threats.

1.3.2. RSA-based Password Authentication Schemes

Named after the three inventors - Ron Rivest, Adi Shamir, and Leonard Adleman - RSA public-Key algorithm has since withstood years of extensive cryptanalysis [69]. Even though the cryptanalysis neither proved nor disproved RSA's security, it does suggest a confidence level in the algorithm. RSA gets its security from the difficulty of factoring large numbers. The public and private keys are functions of a pair of large prime numbers. Recovering the plaintext from the public key and the ciphertext is conjectured to be equivalent to factoring the product of the two primes. The RSA scheme can also be used to create both encryption scheme and digital signatures.

RSA encryption scheme is as follows.

To generate the two keys, A choose two distinct large prime numbers, p and q. For maximum security, he chooses p and q of equal length and computes the product, $n = pq$. He then randomly chooses the encryption key, e, such that $gcd(e,\ (p-1)(q-1)) = 1$; that is e and $(p-1)(q-1)$ are relatively prime. He sends the pair $(n,\ e)$ to B, but he keeps the values of p and q secret.

B, who could possibly be an enemy of A, never needs to know p and q to send her message to A securely. To encrypt a message m, first divide it into numerical blocks smaller that n. That is, if both p and q are 128 digit primes, then n will have just under 256 digits and each message block m_i, should be just under 256 digits long. The encrypted message, c, will be made up of similarly sized message blocks c_i, of about the same length, and sends c_i to A. The encryption formula is simply $c_i = m_i^e \ mod \ n$.

To decrypt a message, A uses the extended Euclidean algorithm to compute the decryption key, d, such that $ed \equiv 1(mod \ (p-1)(q-1))$. In other words, $d = e^{-1} \ mod \ ((p-1)(q-1))$, where d and n are also relatively prime. A takes each encrypted block c_i and compute $m_i = c_i^d \ mod \ n$, where

$$c_i^d \ = \ (m_i^e)^d \ = \ m_i^{k(p-1)(q-1)+1} \ = \ m_1 * m_1^{k(p-1)(q-1)} \ = \ m_1 * 1 = m_i \ (mod \ n).$$

Therefore, A can read the message. The numbers e and n are the public key; the number d is the private key. The two primes, p and q, are no longer needed. They should be discarded, but never revealed.

RSA digital signature scheme is as follows. B generates two large primes p, q, and computes $n = pq$. He selects e_B such that $1 < e_B < n$ with $gcd(e_B, n) = 1$, and computes d_B such that $e_B d_B \equiv 1(mod \ n)$. (e_B, n) is public key, d_B is private key, and (p, q) is kept private.

- B's signature is $s \equiv m^{d_B}(mod \ n)$,
- s is B's signature on m, and then B sends (m, s) to A.

A decrypts and accepts the signature as valid by computing $g \equiv m(mod \ n)$, where $g \equiv s^{e_B} \equiv (m^{d_B})^{e_B} \equiv m(mod \ n)$.

Password authentication scheme with smart cards is proposed by Yang et al. [70]. They do not need to store passwords or verification tables in the server, and user can change their own passwords. It is based on RSA public key cryptosystem. However, [8, 10, 20, 21, 35, 71] pointed out that the Yang et al. scheme [70] is vulnerable to forgery attack by impersonating a legitimate user and constructing a valid login request out of an intercepted login request. Fan et al. [10] proposed a simple improved scheme to withstand against forgery attack that puts a strict limit on the ID. Shen et al. [20] also proposed an enhancement of the Yang et al. scheme [70] that can withstood the forgery attack and the server spoofing attack. However, Chen et al. [72] pointed out that the Fan et al. scheme [10] cannot withstand the forgery attack, and Yang et al. [31] showed that the Shen et al. scheme [20] is still vulnerable to forgery attack. Moreover, Sun et al. [21] explained that the Chan et al. scheme [8] is unreasonable because [8] forge a clients identity, and the identity does not exist in the ID table of the remote server. Later, Yang et al. [35] proposed an improvement of Yang et al. scheme [70] to resist Sun et al.'s attack. However, Kim et al. [71] pointed out that [35] still vulnerable to the forgery attack [21, 31] and proposed their own improvements. Recently, Wang et al. [39] showed that the Kim et al. scheme [71] cannot withstand the forgery attack.

1.3.3. Hash-Based Strong-Password Authentication Scheme

When a user submits a password, the system has to be able to determine whether the user got it right. If the system stores the passwords unencrypted,

then anyone can steal the passwords with access to the system storage or backup disks. However, it is not necessary for the system to know a password in order to verify its correctness. Instead of storing the password, the system stores a hash of the password. When a password is typed, its hash is computed with a hash algorithm and compared with the stored value. If they match, the password is considered to be correct. If the attacker obtains the hashed password file, it is not immediately useful because the passwords cannot be derived from the hashes.

One-way hash function is a mathematical transformation that takes a message of arbitrary-length and computes from it a fixed-length number, $h : x \longrightarrow y \Longrightarrow y = h(x)$. It has the following properties:

- Given x, $h(x) = y$ can be calculated very quickly. However, given y, it is hard to compute $h^{-1}(y) = x$.
- Given x_1, it is computationally infeasible to find x_2 such that $x_1 \neq x_2$ but $h(x_1) = h(x_2)$.
- It is computationally infeasible to find messages x_1 and x_2 with $h(x_1)$ and $h(x_2)$.

One remedy is found the use of one-time password method by Lamport [73]. However, there are some practical difficulties in implementing this method, such as high overhead and password resetting. Another related method is CINON [74] which solves these problems, but requires two random numbers generated by the user, which must be stored by the user in some sort of mobile memory device. On the other hand, the PERM (Privacy Enhanced Information Reading and Writing Management) Protocol [75] stores one random number at the host, which is sent to the user for authentication. However, this system is susceptible to a man-in-the-middle attack if the adversary obtains the logs of two consecutive sessions.

Since the computation cost is lower than those of RSA-based and ElGamal-based password authentication schemes, many articles have proposed one-way hash based password authentication schemes. The SAS (Simple and secure password authentication protocol) protocol proposed in [5] is a simple strong-password authentication scheme that is superior to several well-known schemes. However, it was shown in [7] that the SAS protocol is vulnerable to the replay attack and the denial of service attack. The OSPA (Optimal Strong-Password Authentication) protocol given in [7] claims to be secure against stolen-verifier, replay, and denial of service attacks. Nevertheless, it was shown in [9] that the SAS and OSPA protocols cannot resist the stolen-verifier attack as claimed. Also, an impersonation attack performed in [76] on the OSPA method without an active attack on the server. Later on, an enhanced OSPA protocol was introduced in [18], which resists guessing, reply, impersonation, and stolen-verifier attacks. However, it was shown in [77] that the protocol is still vulnerable to reply and denial-of-service attacks. Furthermore, these two simple attacks can easily be launched without compromising the server in advance.

Recently, plenty of password authentication schemes with smart cards have been proposed [1, 4, 6, 12, 16, 23–25, 28, 29, 33, 34, 38, 78, 79]. Sun [4] proposed an efficient remote use authentication scheme with smart cards. Since it does not need a password table, the communication and computation costs are reduced. However, Chien et al. [79] pointed out that the Sun scheme [4] is able to neither allow users to choose or change their passwords freely nor achieve mutual authentication. Chien et al. [79] and Hwang et al. [12] proposed an efficient and practical solution to remote authentication scheme and a simple remote user authentication scheme, respectively. [79] and [12] claimed that their schemes are able to achieve low communication and computation cost, and withstand the replay attack. How-

ever, Hsu [16, 25] showed that the Chien et al. scheme [79] is vulnerable to the parallel session attack.

In 2004, a hash-based strong-password authentication scheme was described in [28] to against several attacks, including replay, password-file compromise, denial-of-service, and insider attacks. In the same year, Lee et al. [29] proposed an improved efficient scheme to remedy the parallel session attack problem. Chen et al. [23] also proposed two secure SAS-like password authentication schemes with lower storage, processing, and transmission overheads. Those two schemes can resist the stolen-verifier attack on SAS [5] and OSPA [7]. Das et al. [24] proposed a dynamic ID-based remote user authentication scheme with smart cards.

Recently, Liao et al. [33] showed that the Das et al. scheme [24] has some weakness and Yoon et al. [34] pointed out that the Hwang et al. scheme [12] is insecure if the secret key of the server is stolen. Moreover, Liao et al. proposed an improved scheme to against some weakness [33] and further proposed a new scheme to achieve all of their proposed requirements [38]. Chiang et al. [36] showed that the Chen et al. scheme [23] are still vulnerable to denial-of-service attack and the second scheme is not easily reparable.

1.3.4. ID-based Password Authentication Scheme using Smart Cards

Entity authentications are meant to provide secure communications. The concept of an ID-based cryptosystem was first proposed by Shamir [80]. He introduced a novel type of cryptographic scheme that enables any pair of users to communicate securely and to verify each other's signatures. The scheme does

without exchanging private or public keys, keeping key directories, and using the services of a third party. The corresponding secret key, however, is computed by a key generation center and issued to the user in the form of smart card when the user first joins the network. That secret key is fixed in [80]. In the network communication environment, many algorithms and protocols use the synchronized clock system. Transmission delay is unpredictable and a potential replay attack exists in all schemes [81]. In ElGamal's public key cryptosystem [82], the random number is as important as the private key. The attacker can obtain the random number by solving the equations intercepted from a session, if the same random number is used more than once. In a recently published paper, Kim, Lee, and Yoo (KLY) proposed an ID-based password authentication scheme using smart cards and fingerprints [83]. They claimed that their scheme does not require a dictionary of password or verification tables, and allows users choose their password freely of their own will. Finally, their scheme was shown to authenticate legitimate users and to withstand password guessing, message replay, and impersonation attacks.

1.3.5. Hash-Based Secure User Authentication Scheme

User authentication is an important service in network security. Recently, several user authentication protocols have been proposed; however, a scheme which withstands all known attacks is not yet available. An authentication scheme, called Lee-Li-Hwang (LLH) scheme [11], was proposed to circumvent the guessing attack in the Peyravian-Zunic (PZ) password scheme [3]. However, Yoon, Ryu, and Yoo (YRY) [30] discovered that the LLH scheme still suffers from the denial of service

attack, and proposed an enhancement of the LLH scheme to solve its security problems. More recently, Ku, Chiang, and Chang (KCC) [32] demonstrated that the YRY scheme is vulnerable to off-line guessing and stolen-verifier attacks.

1.3.6. User Remote Authentication Scheme

A remote authentication scheme with several distinct advantages was proposed in [4]. These advantages are including low communication and computation cost, and the lack of a password table. Furthermore, a user-friendly remote authentication scheme, with smart cards, without using a user password table was also presented in [84]. In this scheme, the user can choose or change the password. It was pointed out in [85] that the scheme in [84] is insecure due to password guessing and message forgery attacks. Additionally, [86] described another message forgery attack on the same scheme. Recently, the authors of [84] discovered that their scheme is insecure to the replay attack, and presented a modified scheme to overcome this weakness in [87].

1.3.7. Novikov and Kiselev User Authentication Scheme

Whenever users want to access remote systems, they should be authenticated. Once authorized, they can then access the resources of the server. The scatter of remote systems in difference places allows more efficient and convenient access geographically dispersed users. Lamport [73] proposed a remote password authentication scheme which authenticates remote users over an insecure channel.

However, it suffers from the stolen-verifier attack if the adversary has the ability to obtain the stored verifier. It also has some practical implementation difficulties, such as the problems of high overhead and password resetting. Since then, many remote authentication schemes have been proposed [4, 11, 26, 78, 87, 88]. In 2000, Hwang and Li [2] proposed a remote user authentication scheme using smart cards. It was shown that no password table is required to keep in their system. In 2003, Novikov and Kiselev [89] proposed an algorithm of reliable authentication of the user from a remote autonomous object. Recently, Yang, Lee, and Hsiao [90] have pointed out that [89] is insecure against the man-in-the-middle attack. Awasthi [91] also came to the same conclusion and stated that [89] is also vulnerable against the man-in-the-middle attack and the reflection attack.

1.3.8. Authenticated Key Agreement Scheme

The Diffie-Hellman key agreement protocol was developed by Diffie and Hellman in 1976 and published in the ground-breaking paper "New Directions in Cryptography [92]." The protocol allows two users to exchange a secret key over an insecure medium without any prior secrets. However, it is vulnerable to the man-in-the-middle attack, because it does not authenticate the participants. In 1994, Anderson-Lomas showed how collision-rich hash functions can be used to detect those attacks while they are in progress [93]. Later, Seo-Sweeney proposed an efficient simple key agreement protocol that is based on a pre-shared password method, and modifies the Diffie-Hellman scheme to provide user authentication [94]. In the Seo-Sweeney protocol, two parties that have shared a common password can establish a session key by exchanging two messages. This protocol

is more efficient than the Anderson-Lomas scheme in terms of computational time and exchanged messages. In 2000, Tseng [95] pointed out that the key validation of the Seo-Sweeney scheme cannot resist the replay attack. The adversary can successfully convince an honest party of a wrong session key. Tseng proposed an improved scheme to overcome this weakness. However, Ku-Wang showed that the Tseng's modified authenticated key agreement protocol is vulnerable to the modification attack and the backward replay attack without modification [96]. They then proposed an improved scheme to strengthen the protocol. In 2003, Hsu et al. showed that the Ku-Wang modified authentication key agreement scheme is vulnerable to the modification attack, and further proposed an improvement of the Ku-Wang scheme in [97]. Recently, Lee-Lee showed that the Hsu et al. scheme is also vulnerable to the modification attack. An attacker can alter the transmitted messages to deceive the communicating parties into believing a wrong session key [98]. They then proposed another improvement on the modified authenticated key agreement scheme.

1.3.9. Micropayment Scheme

NetBill [99] is an on-line transactional payment protocol with many advanced features that requires communication with the NetBill server for each transaction. However, this protocol has a flaw of double spending. Another scheme proposed in [100] is unappealing for micropayments for this same reason.

Millicent [101] and NetCents [102] are scrip-based off-line-friendly micropayment protocols. The trust model in Millicent defines three roles: vendors, customers, and brokers. Brokers act as intermediaries between vendors and cus-

tomers. A customer enters into a long-term relationship to buy digital money or scrip from brokers, who are assumed to be large entities such as banks or credit card issuers. Brokers are the most trusted party in this model; while customers are the least trusted. Since the monetary unit used in these protocols is vendor-specific, double-spending is very difficult. The assumption behind both protocols is that people tend to re-use the same merchants repeatedly. If this assumption holds, the interactions between the customer and the bank are kept at a minimum. A hidden assumption is that merchants have total information over their sales, so double spending with the same merchant is detectable. However, this scheme does not realize anonymity because the bank purchases scrip from the server on behalf of the client.

The WebMoney [103] schemes realize the anonymity of the customers and the divisibility of the coins. Moreover, it still has an on-line check for detecting double spending. However, it has a flaw in that the bank can deceive the client by rewriting the sum of cash. It also needs a high commission fee from the bank to the server, and requires the client to input a 16 decimal digits prepaid number for every purchase.

Digital cash-based systems [104, 105] provide attractive features such as inherent off-line operation, anonymity, identity revealing on double spending and the Chaum's blind signatures. However, [104] does not directly address the issue of double spending, [105] does not fulfill the divisibility of coins and requires an on-line check for detecting double spending which increases the computation cost for every purchase. The same drawback is apparent in the micropayment protocols, such as PayWord [106]. While the double-spending possibility is an inherent property of all such systems, none of the above protocols employ any kind of risk management scheme to address it.

Rivest and Shamir [106] introduced two simple micropayment schemes, "PayWord" and "MicroMint". Their goal was to minimize the number of public-key operations required per payment by using hash operations whenever possible. Generally, there are two positions of the bank with regard to the certificate. In Position 1, the bank takes full responsibility for the certificate and compensates all payments created by the customer's purchases. In Position 2, the bank does not redeem payments exceeding a limit set for the customer and shares the loss with the shop if trouble occurs.

Recently, Adachi et al. [107] have pointed out that the PayWord scheme has two security problems. A malicious customer can incur damages to the bank by purchasing in excess of the customer's credit. In general, a bank guarantees the customer's credit by issuing a certificate. The shop accepts the customer and initiates the transaction after checking the validity of the certificate that is kept by the customer. Because the certificate is not amended by the shop, malicious customers can use the same certificate at another shop and exceed their true credit level. In the PayWord scheme, the bank could reduce its risk by adopting Position 2 rather than Position 1. However, the bank can damage the shop in Position 2 by impersonating an imaginary customer and making the shop share the loss with the bank. Adachi et al. introduced two attacks: 1) customer certificate abuse attack and 2) bank falsification attack. They then proposed a new micropayment scheme that demands one on-line communication connection between the bank and the shop at the beginning of each transaction between the customer and the shop.

CHAPTER 2. A SIMPLE ATTACK ON A RECENTLY INTRODUCED HASH-BASED STRONG-PASSWORD AUTHENTICATION SCHEME

Recently, a hash-based strong-password authentication scheme was described in [28], which withstands to the several attacks, including replay, password-file compromise, denial-of-service, and insider attacks. However, we show that this protocol is still vulnerable to stolen-verifier, denial-of-service, replay, and impersonation attacks [50].

2.1. Ku's Hash-Based Strong-Password Authentication Scheme

The hash-based strong-password authentication scheme described in [28] comes with two protocols: the registration protocol and the login protocol. We introduce the notation used to describe the protocols below and explain the detailed steps of both of these protocols.

2.1.1. Notations

- U denotes the User, C denotes the Client, S denotes the Server, and A denotes the Adversary.
- h denotes a cryptographic hash function, such that $h(m)$ means the message m is hashed once, while $h^2(m)$ means it is hashed twice, i.e., $h^2(m) = h(h(m))$. Furthermore, $h(a, b)$ denotes the hash of concatenated a and b, i.e., $h(a, b) = h(a||b)$.
- N denotes an integer starting from 1 since U's initial registration.
- P denotes the strong password of U.
- K_S denotes the secret-key of S.
- T denotes the most recent time U initially registered or re-registered at S.
- $\oplus$ denotes the bitwise XOR operation, and $||$ denotes the concatenation.
- The expression $A \longrightarrow B \ : \ X$ means A sends the message X to B via an insecure channel.
- The expression $A \Longrightarrow B \ : \ X$ means A sends the message X to B via a secure channel.

2.1.2. Registration Protocol

This protocol is invoked whenever U initially registers or re-registers to S.

R1. U sends his registration request to S.

R2. $S \longrightarrow U \ : \ N, T.$

S sets T as the currently value of the time. If this is U's initial registration, S sets $N = 1$, otherwise S sets $N = N + 1$. Next, S sends N and T to U.

R3. $U \Longrightarrow S \ : \ h^2(S||P||N||T).$

U computes the verifier $h^2(S||P||N||T)$ and sends it to S.

R4. S computes the user storage key $K_U^{(T)} = h(U||h(K_S||T))$ and the sealed verifier $sv^{(N)} = h^2(S||P||N||T) \oplus K_U^{(T)}$, and then he stores $sv^{(N)}$, N, and T in the password file.

2.1.3. Login Protocol

This protocol is invoked whenever U logins to S.

L1. U sends his login request to S.

L2. $S \longrightarrow U \ : \ r, n, t.$

S selects a random nonce r and retrieves the values of $n = N$ and $t = T$ from $S's$ password file.

L3. $U \longrightarrow S \ : \ c_1, c_2, c_3.$

U sends c_1, c_2, and c_3 to S, where

$$\begin{aligned} c_1 &= h^2(S||P||n||t) \oplus h(S||P||n||t) \ , \\ c_2 &= h(S||P||n||t) \oplus h^2(S||P||n+1||t) \ , \\ c_3 &= h(h^2(S||P||n+1||t)||r) \ . \end{aligned}$$

L4. S computes $K_U^{(t)} = h(U||h(K_S||t))$, and then derives $h^2(S||P||n||t)$ from the stored sealed verifier $sv^{(n)}$ using

$$h^2(S||P||n||t) = sv^{(n)} \oplus K_U^{(t)} \ .$$

Then, S computes u_1 and u_2 using

$$\begin{aligned} u_1 &= c_1 \oplus h^2(S||P||n||t) &&= h(S||P||n||t) \ , \\ u_2 &= c_2 \oplus u_1 &&= h^2(S||P||n+1||t) \ . \end{aligned}$$

If the equalities $h(u_1) = h^2(S||P||n||t)$ and $h(u_2||r) = c_3$ hold, then S authenticates U. Otherwise, S rejects U's login request and terminates the session.

After a successful authentication, S computes a new sealed-verifier using

$$sv^{(n+1)} = u_2 \oplus K_U^{(t)} = h^2(S||P||n+1||t) \oplus K_U^{(t)} \ ,$$

and replaces $sv^{(n)}$ with $sv^{(n+1)}$, and sets $N = n+1$ for U's next login protocol. The value of T is unchanged, i.e., $T = t$.

2.2. Our Attack

We devise an attack assumption that the adversary steals a copy of user's password-verifier $h^2(S||P||N||T)$. Such scenarios are considered in other paper [9].

The second assumption we make is that A is capable of blocking the communication from U to S. After having stolen a copy of the password verifier, A launches an attack whenever it can block communication.

Therefore, our attack assumes that a stolen-verifier attack (by obtaining a copy of the password verifier) and a denial-of-service attack (by blocking the communication from U to S) have succeeded. We then show that under these two

assumptions (attacks), the attacker can now successfully login to the system using replay to impersonate the user.

Below we describe our attack step by step.

1. A steals a copy of U's password-verifier $h^2(S||P||N||T)$.

2. During the U's nth login process, A monitors the communication channel to see the request U made to S and the values r, n, and t sent by S. Next, A captures the values of c_1, c_2, and c_3 sent by U to S and blocks the communication channel from U to S to prevent these values are from reaching S.

3. A computes $h(S||P||n||t)$ and $h^2(S||P||n+1||t)$ with the help of the captured values c_1, c_2, and the previously stolen password-verifier $h^2(S||P||N||T)$ as

$$\begin{aligned} h(S||P||n||t) &= c_1 \oplus h^2(S||P||n||t) \ , \\ h^2(S||P||n+1||t) &= c_2 \oplus h(S||P||n||t) \ , \end{aligned}$$

   where $N = n$ and $T = t$.

4. Next, A sends c_1, c_2, and c_3 to S.

5. After receiving this message, S retrieves t from the password file and computes

$$K_U^{(t)} = h(U||h(K_S||t))$$

   and then uses $K_U^{(t)}$ to compute the verifier $h^2(S||P||n||t)$ with the help of the stored sealed verifier $sv^{(n)}$ as

$$h^2(S||P||n||t) = sv^{(n)} \oplus K_U^{(t)} \ .$$

6. Next, S computes

$$\begin{aligned} u_1 &= c_1 \oplus h^2(S||P||n||t) &&= h(S||P||n||t) \ , \\ u_2 &= c_2 \oplus u_1 &&= h^2(S||P||n+1||t) \ . \end{aligned}$$

If $h(u_1) = h^2(S||P||n||t)$ and $h(u_2||r) = c_3$ hold, S is supposed to authenticate the sender. Since these equalities will hold, S authenticates A as being U. Therefore, S allows the attacker A to login.

7. After this successful login, S updates the sealed verifier according to the Step L4 of login protocol. Therefore, the following will be executed by S. S computes

$$sv^{(n+1)} = u_2 \oplus K_U^{(t)} = h^2(S||P||n+1||t) \oplus K_U^{(t)} \ ,$$

and replaces $sv^{(n)}$ with $sv^{(n+1)}$, and then he sets $N = n+1$ for U's next login protocol. The value of T is unchanged, i.e., $T = t$.

At the end of Step 6, the adversary has successfully logged into the system impersonating the legitimate user. It can now launch other attacks within the system or access sensitive documents. If the user logs in after the attacker does, it may not discover that the attacker, impersonating the user, has logged into the system unless the user checks the login records. Until the user or system managers discover the attacker's successful login, the attacker can continue to impersonate the user.

2.3. Conclusions

In this section, we have shown that a hash-based strong-password authentication scheme proposed in [28] is vulnerable if the attacker is able to obtain a

copy of the verifier (stolen-verifier attack) and briefly block the communication from the user to the server (denial-of-service attack). Until the legitimate user or the system manager is able to notice the attack, the attacker can impersonate the user. If the time between two consecutive logins takes long, then the attacker is expected to inflict considerable damage by violating the security principles.

CHAPTER 3. A SIMPLE ATTACK ON A RECENTLY INTRODUCED HASH-BASED SECURE USER AUTHENTICATION SCHEME

The Lee-Li-Hwang (LLH) authentication scheme [11] was proposed to circumvent the guessing attack in the Peyravian-Zunic (PZ) password scheme [3]. However, Yoon, Ryu, and Yoo (YRY) [30] discovered that the LLH scheme still suffers from the denial of service attack, and proposed an enhancement for the LLH scheme to solve its security problems. More recently, Ku, Chiang, and Chang (KCC) [32] demonstrated that the YRY scheme is vulnerable to off-line guessing and stolen-verifier attacks. In this chapter, we show that the YRY scheme is also vulnerable to the denial-of-service attack. Furthermore, it was also claimed in [32] that the YRY scheme cannot achieve backward secrecy. We show in this chapter that this claim is not entirely valid [37].

3.1. LLH Scheme

A hash-based secure user authentication scheme was described in [30]. The scheme has 3 phases: Registration phase, User authentication phase, and Change password phase. We first introduce new notations used to describe these protocols (other notations are the same as chapter 2), and then the detailed steps of these protocols.

3.1.1. Notations

- UID denotes the identification of the user.
- P denotes the memorable password of the user.
- R_c and R_s denote random numbers generated by Client and Server, respectively.

3.1.2. Registration Phase

This registration phase is performed only once when a new user wants to join the system. On the other hand, the authentication phase is executed whenever the user wants to login to the system. The procedures of this phase are as follows:

R1. $U \Longrightarrow S \ : \ UID, \ HPW$

U randomly chooses UID and P, and then calculates a password verifier $HPW = h(UID, P)$.

R2. S stores UID and HPW in the verification table.

3.1.3. User Authentication Phase

In this phase, the user logs in to a server for accessing resources and the server authenticates the user. The procedures of this phase are as follows:

A1. $C \longrightarrow S \ : \ UID, \ R_c \oplus HPW, \ h(R_c)$.

U enters UID and P to C. C computes $HPW = h(UID, P)$ and randomly

chooses a number R_c, and then computes the hash value $h(R_c)$. Next, C sends UID, $R_c \oplus HPW$, and $h(R_c)$ to S.

A2. $S \longrightarrow C$: $R_s \oplus HPW,\ h(R_c, R_s)$.

S retrieves the $U's$ password verifier HPW from the verification table, and then obtains R_c by computing $(R_c \oplus HPW) \oplus HPW$. Next, S verifies the equality of the computed $h(R_c)$ with the obtained R_c and the received $h(R_c)$. If they are equal, S randomly generates a number R_s, and then computes $R_s \oplus HPW,\ h(R_c, R_s)$, and $AUTH^* = h(HPW, R_c, R_s)$. Next, S sends $R_s \oplus HPW$ and $h(R_c, R_s)$ to C.

A3. $C \longrightarrow S$: $UID, AUTH$.

C retrieves R_s by using $(R_s \oplus HPW) \oplus HPW$ and computes $h(R_c, R_s)$. If the computed and received $h(R_c, R_s)$ are equal, C computes $AUTH = h(HPW, R_c, R_s)$ and sends UID and $AUTH$ to S.

A4. S compares $AUTH$ with $AUTH^*$. If they are equal, S authenticates U. Otherwise, S rejects C's request and terminates the session.

3.1.4. Change Password Phase

The change password phase is invoked whenever client wants to change its password P with a new one, say $NewP$. The procedures of this phase are given below. Note that Steps C1 and C2 are the same as the ones in the user authentication phase.

C3. $C \longrightarrow S$: $UID,\ AUTH,\ Mask,\ V_{Mask}$.

C retrieves R_s by using $(R_s \oplus HPW) \oplus HPW$ and computes $h(R_c, R_s)$. If

the computed and received $h(R_c, R_s)$ are equal, then C computes

$$\begin{aligned} NewHPW &= h(UID, NewP) \ , \\ AUTH &= h(HPW, R_c, R_s) \ , \\ Mask &= NewHPW \oplus h(HPW, R_c + 1, R_s) \ , \\ V_{Mask} &= h(NewHPW, R_s) \ . \end{aligned}$$

Then, C sends UID, $AUTH$, $Mask$, and V_{Mask} to S.

C4. S retrieves the $U's$ HPW from the verification table. If $AUTH = AUTH^*$, S accepts C to change the $U's$ password, and then obtains new password verifier $NewHPW$ as $NewHPW = Mask \oplus h(HPW, R_c + 1, R_s)$. Next, S calculates $h(NewHPW, R_s)$ and compares it with V_{Mask}. If they are equal, S replaces the old HPW with the new password verifier $NewHPW$ in the verification table. Otherwise, S rejects C's change password request and terminates the session.

3.2. KCC Impersonation Attack with Stolen-Verifier

Suppose that the adversary has stolen the verifier $HPW = h(UID, P)$ of the user from the server. The adversary can compute R_c, $(R_c \oplus HPW) \oplus HPW$ by XORing, and then he can get more information in sequence, computing $h(R_c)$, R_s using $(R_s \oplus HPW) \oplus HPW$, $h(R_c, R_s)$, and $AUTH^* = h(HPW, R_c, R_s)$. After that, the adversary has all the information he needs to login into the server. If the adversary obtains an HPW through the stolen-verifier attack, he can then perform the following:

B1. A can make a random generated number R_a to compute $R_a \oplus HPW$ and $h(R_a)$. He sends $UID,\ R_a \oplus HPW$, and $h(R_a)$ to the server in Step A1.

B2. S retrieves the $R_a = (R_a \oplus HPW) \oplus HPW$ by XORing, and then S verifies the equality of the computed $h(R_a)$ and received $h(R_a)$. If they are equal, S randomly generates a number R_s and computes $R_s \oplus HPW$, $h(R_a, R_s)$, and $AUTH^* = h(HPW, R_a, R_s)$. S sends $R_s \oplus HPW$ and $h(R_a, R_s)$ to A in Step A2.

B3. A retrieves R_s using $(R_s \oplus HPW) \oplus HPW$ and computes $h(R_a, R_s)$. Next, if the computed and received $h(R_a, R_s)$ are equal, A computes $AUTH = h(HPW, R_a, R_s)$ and sends UID and $AUTH$ to S in Step A3.

B4. S compares $AUTH$ with $AUTH^*$. If they are equal, S authenticates A in Step A4.

After that, A can impersonate U.

Additionally, this attack can be adapted on the change password phase in the same way. This is described as below.

B5. A can get the R_s and $AUTH = h(HPW, R_c, R_s)$ after Steps C1 and C2, and he can then choose his new password P_a and the random number R_a. Next, A computes $NewHPW$, $Mask$, and V_{Mask} with his own P_a as

$$\begin{aligned} NewHPW &= h(UID, P_a)\ , \\ Mask &= NewHPW \oplus h(HPW, R_c + 1, R_s)\ , \\ AUTH &= h(HPW, R_a, R_s)\ , \\ V_{Mask} &= h(NewHPW, R_s)\ . \end{aligned}$$

Then, A sends $UID,\ AUTH,\ Mask$, and V_{Mask} to S in Step C3.

B6. After receiving these values, S retrieves U's HPW from the verification table and compares $AUTH = AUTH^*$. If they are equal, S accepts A to change the user $U's$ password P with $A's$ password P_a.

B7. S obtains the $A's$ new password verifier $NewHPW$ as $NewHPW = Mask \oplus h(HPW, R_c + 1, R_s)$, and then S compares $h(NewHPW, R_s)$ with V_{Mask}. Since $h(NewHPW, R_s) = V_{Mask}$, it accepts and S replaces the old HPW with the new password verifier $NewHPW$ in the verification table.

Thus, the adversary can impersonate as the user to login and change the password. He can then launch other attacks within the system. If the user logs in after an attack, she may not be able to discover that the attacker has logged into the system impersonating as her, without checking the login records. Until the user or the system manager discovers the attacker's login, the attacker may continue to impersonate the user.

3.3. Our Denial of Service Attack with the Stolen-Verifier

The adversary is able to prevent the client from logging in during the user authentication phase or change its password P with $NewP$ in the change password phase by making the server reject all login requests and change password requests. As mentioned in the impersonation attack, the adversary can replace all information that were related to the login and change password phases as follows:

- $R_c \longrightarrow R_a$, $NewP \longrightarrow P_a$,
- $NewHPW = h(UID, NewP) \longrightarrow NewHPW^* = h(UID, P_a)$,

- $AUTH = h(HPW, R_c, R_s) \longrightarrow AUTH^* = h(HPW, R_a, R_s)$,
- $Mask = NewHPW \oplus h(HPW, R_c + 1, R_s)$
  $\longrightarrow Mask^* = NewHPW^* \oplus h(HPW, R_a + 1, R_s)$

After receiving the replaced message, if the user tries to login the server, he will be rejected since both the password and the password verifier were changed.

DoS1. In the user authentication phase, U enters UID and P to C. C computes $HPW = h(UID, P)$ and randomly chooses a number R_c, and then computes $h(R_c)$. Next, C sends UID, $R_c \oplus HPW$, and $h(R_c)$ to S in Step A1.

Since S retrieves $A's$ new password verifier $NewHPW^* = h(UID,\ P_a)$ from the verification table, he obtains R_c^* that is different from R_c, R_c^* was obtained by computing $(R_c \oplus HPW) \oplus NewHPW^*$.

Next, S verifies the equality of the computed $h(R_c)$ and the received $h(R_c^*)$. They are not equal. Therefore, S rejects $C's$ request.

DoS2. Even though this attack happened after $U's$ successful login, the problem is the same as in the user change password phase since the request in Step C1 is the same as in Step A1.

DoS3. If this attack happened after Step C2,

C computes $NewHPW = h(UID, NewP)$, $AUTH' = h(HPW, R_c, R_s)$, $Mask = NewHPW \oplus h(HPW, R_c + 1, R_s)$, and $V_{Mask} = h(NewHPW, R_s)$, and then C sends UID, $AUTH$, $Mask$, and V_{Mask} to S in Step C3.

At this moment, $AUTH^* = h(HPW, R_a, R_s)$ is not equal to $AUTH' = h(HPW, R_c, R_s)$ that S computed in Step C2, not in Step C3. Therefore, S rejects C's to change U's password.

DoS4. If this attack happened after Step C3, C computes $NewHPW, AUTH, Mask$, and V_{Mask} the same as Step DoS3, and then C sends $UID, AUTH, Mask$, and V_{Mask} to S in Step C3. $AUTH' = h(HPW, R_c, R_s)$ is equal to $AUTH = h(HPW, R_c, R_s)$ that S computes in Step C2, accordingly, S accepts C to change the U's password. However, S obtains a different password verifier as $NewHPW' = Mask \oplus h(HPW, R_a + 1, R_s)$, which is not equal to U's new verifier $NewHPW$, since R_c was already changed with R_a by A. After that, S computes $h(NewHPW', R_s)$ and compares it with V_{Mask}. The value of $h(NewHPW', R_s)$ is not equal to $V_{Mask} = h(NewHPW, R_s)$. Consequently, S rejects C's change password request and terminates the session.

For those reason, both the user's authentication and change password requests are rejected until the user has re-registered with the server.

The adversary can interrupt or lock the account of any user. In addition, this attack works even if P is a strong password.

3.4. No Lack of Backward Secrecy

It was assumed in [32] that the adversary can steal the HPW. If C detects that HPW is compromised, it can invoke the password change phase to change password P with a new one, say $NewP$. However, by intercepting the messages transmitted in Step C1 and Step C2 of the change password phase, the adversary can use the stolen HPW to retrieve R_c and R_s, and compute $h(HPW, R_c+1, R_s)$. Moreover, by intercepting the message transmitted in Step C3 of the change password phase, the adversary can use the computed $h(HPW, R_c + 1, R_s)$ to retrieve

$NewHPW$ from $Mask(= NewHPW \oplus h(HPW, R_c + 1, R_s))$.

However, there is a limitation. Even though the adversary intercepts the messages in Step C1 and Step C2 of the change password phase, he cannot retrieve R_c and R_s, because the HPW is already changed to $NewHPW$, and it is not equal to HPW of the previous stolen verifier. If the adversary wants to get R_c and R_s after the change password phase, he needs to obtain the new password verifier. Only then can the adversary compute $h(HPW, R_c + 1, R_s)$. Therefore, the claim in [32] is not valid.

3.5. Conclusions

In this chapter, we have shown that a hash-based secure user authentication scheme proposed in [30], which resists the several attacks (such as replay, server spoofing, and denial-of-service attacks) is still vulnerable. If the adversary is able to obtain a copy of the verifier, he can launch denial-of-service, stolen-verifier, and impersonation attacks to interrupt communication between the user and the server. Furthermore, we show that the claim made in [32] about the lack of backward secrecy in the YRY scheme is not valid.

CHAPTER 4. VULNERABILITIES IN THE ADACHI-AOKI-KOMANO-OHTA MICROPAYMENT SCHEME

Rivest and Shamir presented two simple micropayment schemes [106], "PayWord" and "MicroMint," for making small purchases over the Internet. Recently, Adachi et al. [107] have pointed out that the PayWord scheme has two security problems, and proposed a new micropayment scheme to overcome these problems. Nevertheless, we show that their protocol is still vulnerable to impersonation and replay attacks [51].

4.1. Adachi et al's Micropayment Scheme

This scheme was described in [107]. We first introduce the notation used to describe the protocols then briefly show their scheme.

4.1.1. Notations

- $I_B,\ I_C, and\ I_S$ denote bank B, customer C, shop $S's$ ID, respectively.
- PK_B/PK_C denote the public keys of B and C.
- SK_B/SK_C denote the secret keys of B and C.

- $\{M\}_{SK_B}/\{M\}_{SK_C}$ denote a message M with its digital signature that was generated by $B/C's$ secret key.
- E denotes the expiration date of M.
- I denotes any additional information.
- r denotes a random nonce that was selected by S.

4.1.2. Adachi et al's Scheme

In the PayWord Scheme [106], the bank notices that a customer certificate abuse attack has occurred when it gets the hash coins with certificates from each shop used by the client. By this time, the shop has already sent goods or provided services to the client since it trusted the certificate. To cover the shop's, the bank withdraws the money corresponding to the length of the hash chain from the client's account and pools it in advance when the client starts a transaction with the shop. Moreover, the bank guarantees the validity of the customer's public key and can impersonate an imaginary customer as an avenue of an attack. To avoid this possibility, the validity of this key is guaranteed not by the bank but by the CA. In the Adachi et al's scheme, instead of the shop, the bank verifies the commitment M and guarantees its validity for the client's certificate. The shop can then check the validity of the commitment and the client's credit at the same time. Hence, their scheme can reduce the verification cost of the commitment without degrading security.

P1. C requests B to establish a bank account.

P2. B makes $C's$ bank account and replies to C.

P3. $C \longrightarrow S : M = \{I_S, w_0, n, E\}_{SK_C}$

C produces the value w_n at random and computes a hash chain, $w_n \xrightarrow{h} w_{n-1} \xrightarrow{h} \cdots w_1 \xrightarrow{h} w_0$. C can select the length n of the hash chain. Next, C calculates M and sends it to S.

P4. $S \longrightarrow B : I_C, M$, and r

S sends I_C, M, and r to B and requests a check for the validity of $C's$ credit.

P5. $B \longrightarrow S : C_C = \{I_C, M, Yes, r, I\}_{SK_B}$

B checks M and $C's$ credit. If they are valid, B withdraws the money corresponding to the hash length n from $C's$ account and keeps it as a pooled value in its database (DB). Next, B computes a certificate C_C and sends it to S.

P6. S checks C_C using PK_B to confirm $C's$ credit and $M's$ correctness.

P7. $C \longrightarrow S : h(w_i, i)$

C sends an order and the hash value $h(w_i, i)$ for the payment to S, where i is the index of the hash value.

P8. $S \longrightarrow C :$ Goods or services

S verifies $w_{i-1} = h(w_i)$ and confirms the validity of the payment. If S verifies all transactions, he sends goods or provides services required to C.

P9. $S \longrightarrow B : h(w_i, i)$

S sends the hash value $h(w_i, i)$ for the payment to B. In this step, S may send only the latest coin $h(w_k, k)$ received from C for the payment.

P10. B verifies C_C and $h(w_k, k)$ received from S using the information stored in $B's$ DB. If they are valid, then B puts money into $S's$ bank account from the pooled value of C in $B's$ DB. Next, B updates and stores only the latest coin in $B's$ DB.

4.2. Our Attacks

In Adachi et al's proposed scheme, we suppose that the adversary A monitors their communications. Our attack is briefly summarized below:

A1. $C \longrightarrow A : M = \{I_S, w_0, n, E\}_{SK_C}$

A intercepts the message $M = \{I_S, w_0, n, E\}_{SK_C}$ in Step P3. Next, A decrypts M with PK_C and obtains I_S, w_0, n, and E.

A2. $A \longrightarrow B : I_C, M$, and r_A

A pretends to be S and sends ID of C, M, and his own random number r_A to B. Next, he requests a check for the validity of $C's$ credit.

A3. $B \longrightarrow A : C_C = \{I_C, M, Yes, r_A, I\}_{SK_B}$

B checks M and $C's$ credit. If they are valid, B withdraws the money corresponding to the hash length n from $C's$ account and keeps it as a pooled value in its database (DB). Next, B computes a certificate C_C and sends it to A. A decrypts C_C with PK_B and obtains I_C, M, Yes, r_A, and I.

A4. $C \longrightarrow A : h(w_i, i)$

C sends an order and the hash value $h(w_i, i)$ for the payment to S. A intercepts this hash value $h(w_i, i)$.

A5. $A \longrightarrow B : h(w_i, i)$

A sends the hash value $h(w_i, i)$ for the payment to B and ask to put the money $A's$ account. In this step, A may send only the latest coin $h(w_k, k)$ received from C for the payment.

A6. B verifies C_C and $h(w_k, k)$ received from A using the information stored in $B's$ DB. If they are valid, then B puts money into $A's$ bank account from the pooled value of C in $B's$ DB. Next, B updates and stores only the latest coin in $B's$ DB.

When A intercepts the message in Step P3 and obtains $M = \{I_S, w_0, n, E\}_{SK_C}$, he can decrypt it with PK_C to get I_S, w_0, n and E. Next, A pretends to be S and replays I_C, M, and his own random number r_A to B in Step A2. B checks M and $C's$ credit. Since they are valid, B withdraws the money corresponding to the hash length n from $C's$ account and keeps it as a pooled value in his database. Next, B computes a certificate C_C and sends it to A in Step A3. A decrypts C_C with PK_B and obtains I_C, M, Yes, r_A, and I. Here, A has more information from I. C sends an order and the hash value $h(w_i, i)$ for the payment to S. A intercepts this hash value $h(w_i, i)$. A can replay again by sending the hash value $h(w_i, i)$ for the payment to B. In this step, A may send only the latest coin $h(w_k, k)$ received from C for the payment. After B verifies C_C and $h(w_k, k)$, B puts money into $A's$ bank account from the pooled value of C in $B's$ DB. In addition, in order to dig a pit for the third party, A would instead impersonate the third party and cause the bank to deposit money into the third party's account. This is achieved by sending information of which account the bank needs to deposit to in Step A6.

Even though A did not know anything before Step P7, if he can obtain the hash value $h(w_i, i)$ in Step P7, he can attack successfully. By sending $h(w_i, i)$ and

asking to put the money into $A's$ account, B will verify C_C and $h(w_k, k)$. If they are valid, B puts money into $A's$ bank account from the pooled value of C in $B's$ DB.

After Step A3, A obtains any additional information from I. In addition, since he gets information $h(w_i, i)$ in Step A4, he can compute all hash values. The attacker can continue to impersonate the customer and the shop. So he can attack again until consume the whole money which he can spend before the end of expiration date E.

Thus, A can attack successfully by replaying the original messages and modifying them to impersonate another party. Therefore, their proposed scheme is still vulnerable against replay and impersonation attacks.

4.3. Conclusions

In this chapter, we have shown that the solutions to the security problems found in Rivest and Shamir's PayWord scheme proposed in [107] are still vulnerable to impersonation and replay attacks.

CHAPTER 5. ENHANCED SECURITY FOR THE MODIFIED AUTHENTICATED KEY AGREEMENT SCHEME

Hsu et al. [97] showed that the Ku-Wang [96] modified authentication key agreement scheme is vulnerable to the modification attack and further proposed an improvement of the Ku-Wang scheme. Recently, Lee-Lee [98] showed that the Hsu et al.'s scheme is vulnerable to the modification attack and proposed another improvement on the modified authenticated key agreement scheme. However, we will show that the Hsu et al.'s scheme suffers from the off-line guessing attack and the Lee-Lee's scheme is still vulnerable to off-line guessing, man-in-the-middle, and reflection attacks. We then propose an enhanced secure scheme to eliminate these security flaws [52].

5.1. Hsu et al.'s scheme

There are two phases in this scheme: the key establishment phase and the key validation phase. We first introduce the notation to describe both phases and explain the detailed steps in each phase.

5.1.1. Notations

- A, B, and E denote the two communicating users and the adversary.
- ID_A and ID_B denote the identities of user A and B.
- a and b denote the random number chosen by A and B.
- n denotes a large prime number.
- g denotes a generator with the order n-1 in GF(n).
- K_1 and K_2 denote the session key of A and B.
- P denotes the common password shared between A and B.
- Q and Q^{-1} denote an integer computed from P and the inverse of $Q(mod\ n)$.

5.1.2. Key Establishment Phase

E1. $A \longrightarrow B\ :\ X_1$.

A computes $X_1 = g^{aQ} mod\ n$ and sends X_1 to B.

E2. $B \longrightarrow A\ :\ Y_1$.

B computes $Y_1 = g^{bQ} mod\ n$ and sends Y_1 to A.

E3. A computes the session key $K_1 = Y^a mod\ n = g^{ab} mod\ n$,

where $Y = Y_1^{Q^{-1}} mod\ n = g^b mod\ n$.

E4. B computes the session key $K_2 = X^b mod\ n = g^{ab} mod\ n$,

where $X = X_1^{Q^{-1}} mod\ n = g^a mod\ n$.

After Step E4, A and B obtain the same session key $K_1 = K_2 = g^{ab} mod\ n$.

5.1.3. Key Validation Phase of the Hsu et al.'s scheme

HV1. $A \longrightarrow B \ : \ X_2$.

A computes $X_2 = h(ID_A, \ K_1)$ and sends X_2 to B.

HV2. B verifies the validity of X_2 with $h(ID_A, \ K_2)$.

HV3. $B \longrightarrow A \ : \ Y_2$.

If the value is correct, B computes $Y_2 = h(ID_B, \ K_2)$ and sends Y_2 to A.

HV4. A verifies the validity of Y_2 with $h(ID_B, \ K_1)$.

After Step HV4, A and B share the common session key $K_1 = K_2 = g^{ab} mod \ n$.

5.2. Lee-Lee's scheme

The modified authenticated key agreement scheme described in [98] consists of two phases: the key establishment phase and the key validation phase. The key establishment phase is the same as the key establishment phase in the Hsu et al.'s scheme.

5.2.1. Key Validation Phase of the Lee-Lee's scheme

V1. $A \longrightarrow B \ : \ X_2$.

A computes $X_2 = h(ID_A, \ X_1, \ K_1)$ and sends X_2 to B.

V2. B verifies the validity of X_2 with $h(ID_A, \ X_1, \ K_2)$.

V3. $B \longrightarrow A \ : \ Y_2$.

If the value is correct, B computes $Y_2 = h(ID_B,\ Y_1,\ K_2)$ and sends Y_2 to A.

V4. A verifies the validity of Y_2 with $h(ID_B,\ Y_1,\ K_1)$.

After Step V4, A and B shared the common session key $K_1 = K_2 = g^{ab} mod\ n$.

5.3. Our Attack

First, we show an attack against [97] with the off-line guessing attack. Later, we show three possible attacks against [98]; the off-line guessing attack, the man-in-the-middle attack, and the reflection attack.

To attack these schemes, we suppose that the adversary E would eavesdrop and interpose the communication between A and B.

5.3.1. Off-line Guessing Attack on the Hsu et al.'s scheme

GAH1. $E \longrightarrow A \ : \ Y_E$.

E monitors and intercepts the message $X_1 = g^{aQ} mod\ n$ in Step E1. He randomly selects integer Z, and then computes his own value, $Y_E = g^z\ mod\ n$. Next, he sends Y_E to A.

GAH2. A computes the session key $K_1^* = Y^{*a} mod\ n = g^{ZaQ^{-1}} mod\ n$,
where $Y^* = Y_E^{Q^{-1}} mod\ n$.

GAH3. $A \longrightarrow B \ : \ X_2^*$.

To verify the validity of the session key K_1^*, A computes $X_2^* = h(ID_A,\ K_1^*)$ and sends X_2^* to B in Step HV1.

GAH4. E intercepts X_2^* and tries to find the suitable value. First, he computes Q_E and Q_E^{-1} from the guessing password P_E. Second, he computes his own value $X_E^* = h(ID_A,\ X_1^{Z(Q_E^{-1})^2})$. Next, he compares $X_2^* = h(ID_A,\ g^{ZaQ^{-1}} mod\ n)$ with X_E^*. If they match, this guessing attack is succeeded. Otherwise, E tries to find the password again in this Step GAH4.

Therefore, this off-line guessing attack can succeed when E impersonates B.

5.3.2. Off-line Guessing Attack on the Lee-Lee's scheme

GA1. $E \longrightarrow B\ :\ X_E$.

E monitors and intercepts the message $X_1 = g^{aQ} mod\ n$ in Step E1. He then computes his own value $X_E = g\ mod\ n$, and sends X_E to B.

GA2. $E \longrightarrow A\ :\ Y_E$.

E intercepts $Y_1 = g^{bQ} mod\ n$ in Step E2. He then replaces Y_1 with $Y_E = g\ mod\ n$, and sends Y_E to A.

GA3. A computes the session key $K_1^* = g^{aQ^{-1}} mod\ n$,
where $Y^* = Y_E^{Q^{-1}} mod\ n = g^{Q^{-1}} mod\ n$, and
$K_1^* = Y^{*a} mod\ n = g^{aQ^{-1}} mod\ n$.

GA4. B computes the session key $K_2^* = g^{bQ^{-1}} mod\ n$,
where $X^* = X_E^{Q^{-1}} mod\ n = g^{Q^{-1}} mod\ n$, and
$K_2^* = X^{*b} mod\ n = g^{bQ^{-1}} mod\ n$.

GA5. $A \longrightarrow B\ :\ X_2^*$.

To verify the validity of the session key K_1^*, A computes $X_2^* = h(ID_A,\ X_1,\ K_1^*)$ $= h(ID_A,\ g^{aQ} mod\ n,\ g^{aQ^{-1}} mod\ n)$, and sends X_2^* to B in Step V1.

GA6. E intercepts X_2^* and tries to find the suitable value. E computes Q_E and Q_E^{-1} from the guessing password P_E. He then computes his own value $X_E^* = h(ID_A,\ X_1,\ X_1^{(Q_E^{-1})^2})$. Next, he compares $X_2^* = h(ID_A,\ X_1,\ K_1^*) = h(ID_A,\ g^{aQ} mod\ n, g^{aQ^{-1}} mod\ n)$ with X_E^*. If they match, this guessing attack is successful. Otherwise, E tries to find the password again in Step GA6.

Thus, E can guess the password in this attack. Furthermore, if this attack is the on-line guessing attack, it might be successful. However, it will be detected by the system or the users, since E has to access several times more than the limited access time, if his attack fails. Therefore, this off-line guessing attack can be successful like the attack on the Hsu et al.'s scheme.

5.3.3. Man-In-The-Middle Attack

MA1. $A \longrightarrow B\ :\ X_1 \quad \Rightarrow \quad E \longrightarrow B\ :\ X_{E1}$.

A computes $X_1 = g^{aQ} mod\ n$ and sends X_1 to B. However, E monitors and intercepts the message X_1, and then replaces X_1 with his own value $X_{E1} = 1\ mod\ n$. Next, he sends X_{E1} to B.

MA2. $B \longrightarrow A\ :\ Y_1 \quad \Rightarrow \quad E \longrightarrow A\ :\ Y_{E1}$.

B computes $Y_1 = g^{bQ} mod\ n$ and sends Y_1 to A. However, E also intercepts the message Y_1, and then replaces Y_1 with his own value $Y_{E1} = 1\ mod\ n$. He then sends Y_{E1} to A. Next, E calculates $X_{E2} = h(ID_A,\ X_{E1},\ K_1^*)$ and $Y_{E2} = h(ID_B,\ Y_{E1},\ K_2^*)$.

MA3. For the session key, A computes

$Y^* = Y_{E1}^{Q^{-1}} mod\ n = 1\ mod\ n$ and $K_1^* = Y^{*a}\ mod\ n = 1\ mod\ n$.

MA4. By the same way, B computes

$X^* = X_{E1}^{Q^{-1}} mod\ n = 1\ mod\ n$ and $K_2^* = X^{*b}\ mod\ n = 1\ mod\ n$.

Finally, A and B obtain the same session key $K_1^* = K_2^* = 1\ mod\ n$. After Step MA4, user A and B start to verify their session key K_1^* and K_2^*.

MA5. $A \longrightarrow B\ :\ X_2^* \quad \Rightarrow \quad E \longrightarrow B\ :\ X_{E2}$.

A computes $X_2^* = h(ID_A,\ X_1,\ K_1^*)$ and sends it to B. E intercepts X_2^*, and then easily substitutes X_2^* with X_{E2}, since X_{E2} and Y_{E2} are already calculated after Step MA2. Next, he sends X_{E2} to B.

MA6. After received it, B verifies the validity of $X_{E2} = h(ID_A,\ X_{E1},\ K_1^*)$ with $h(ID_A,\ X_{E1},\ K_2^*)$.

MA7. $B \longrightarrow A\ :\ Y_2^* \quad \Rightarrow \quad E \longrightarrow A\ :\ Y_{E2}$.

Similarly, B computes $Y_2^* = h(ID_B,\ X_2,\ K_2^*)$ and sends Y_2^* to A. However, E substitutes Y_2^* with Y_{E2}, and sends Y_{E2} to A.

MA8. After A received it, he verifies the validity of $Y_{E2} = h(ID_B,\ Y_{E1},\ K_2^*)$ with $h(ID_B,\ Y_{E1},\ K_1^*)$.

Since the session key $K_1^* = K_2^* = 1\ mod\ n$, A and B get the common session key and convince it without doubt. Thus, E can attack Lee-Lee's scheme, if he can eavesdrop certain users' communications.

5.3.4. Reflection Attack

RA1. $A \longrightarrow E\ :\ X_1$.

A computes $X_1 = g^{aQ} mod\ n$ and sends X_1 to B.

RA2. $E \longrightarrow A \ : \ X_E$.

E intercepts the message X_1 in Step E1. He then computes his own value $X_E = g \ mod \ n$, and sends A the X_E instead of Y_1.

RA3. $A \longrightarrow E \ : \ X_2^*$.

A computes the session key $K_1^* = g^{aQ^{-1}} mod \ n$,

where $Y^* = X_E^{Q^{-1}} mod \ n = g^{Q^{-1}} mod \ n$ and $K_1^* = Y^{*a} mod \ n = g^{aQ^{-1}} mod \ n$.

To verify the validity of the session key K_1^*,

A computes $X_2^* = h(ID_A, \ X_1, \ K_1^*) = h(ID_A, \ g^{aQ} mod \ n, \ g^{aQ^{-1}} mod \ n)$, and sends X_2^* to B in Step V1. However, E intercepts X_2^*, and ferrets out the proper value as we attacked in Step GA6. Finally, he can compute Q_E and Q_E^{-1} with the guessing password P_E.

As we have shown in this attack, it is obvious that all users have communicated with malicious adversary E, not with the trusted parties. Thus, E is able to put up an illusion to deceive other users.

5.4. Our Enhanced Secure Scheme

To resist the above weaknesses, we propose an enhanced secure scheme taking into account off-line guessing, man-in-the-middle, and reflection attacks of [98].

5.4.1. Enhanced Key Establishment Phase

EE1. $A \longrightarrow B \ : \ X_1$.

A computes $X_1 = g^{aQ} \oplus Q$ and sends X_1 to B.

EE2. $B \longrightarrow A \ : \ Y_1$.

B computes $Y_1 = g^{bQ} \oplus Q$ and sends Y_1 to A.

EE3. A computes the session key $K_1 = g^{abQ} mod\ n$,

where $Y = Y_1 \oplus Q = (g^{bQ} \oplus Q) \oplus Q = g^{bQ}$ and $K_1 = Y^a mod\ n$.

EE4. B computes the session key $K_2 = g^{abQ} mod\ n$,

where $X = X_1 \oplus Q = (g^{aQ} \oplus Q) \oplus Q = g^{aQ}$ and $K_2 = X^b mod\ n$.

After Step EE4, A and B obtain the same session key $K_1 = K_2 = g^{abQ} mod\ n$.

5.4.2. Enhanced Key Validation Phase

In this phase, to fend off the man-in-the-middle attack, A and B check the value of $K_1 \neq 1$ and $K_2 \neq 1$, respectively. If the values are correct, then they can start this phase.

EV1. $A \longrightarrow B \ : \ X_2$.

A computes $X_3 = K_1^{-a} mod\ n = g^{bQ} mod\ n$ and $X_2 = h(ID_A,\ Y_1,\ K_1) \oplus h(X_3)$, and then sends X_2 to B.

EV2. $B \longrightarrow A \ : \ Y_2$.

B computes $X_3' = g^{bQ} mod\ n$, and verifies the validity of X_2 with $h(ID_A,\ Y_1,\ K_2)$ $\oplus h(X_3')$. If they are correct, B computes $Y_3 = K_2^{-b} mod\ n = g^{aQ} mod\ n$ and $Y_2 = h(ID_B,\ X_1,\ K_2) \oplus h(Y_3)$, and then sends Y_2 to A.

EV3. A computes $Y_3' = g^{aQ} mod\ n$, and verifies the validity of Y_2 with $h(ID_B,\ X_1,\ K_1)$ $\oplus h(Y_3')$. If they are correct, then A and B are able to share the common session key $K_1 = K_2 = g^{abQ} mod\ n$. If they are not, then they discard the session key.

5.5. Security Analysis

5.5.1. Off-line Guessing Attack

If E eavesdrops and intercepts the information for the off-line guessing attack, he can obtain information such as $X_1 = g^{aQ} \oplus Q$ and $Y_1 = g^{bQ} \oplus Q$. After that, E replaces X_1 and Y_1 with $X_E = g^{a'Q'} \oplus Q'$ and $Y_E = g^{b'Q'} \oplus Q'$, respectively. He then sends them to A and B in Step GA3 and GA4, respectively. Next, A and B compute K_1^*, K_2^*, X_2^*, and Y_2^*, where

$K_1^* = Y^{*a} mod\ n = (g^{b'Q'} \oplus Q' \oplus Q)^a mod\ n$

from $Y^* = Y_E \oplus Q = (g^{b'Q'} \oplus Q') \oplus Q$,

$K_2^* = X^{*b} mod\ n = (g^{a'Q'} \oplus Q' \oplus Q)^b mod\ n$

from $X^* = X_E \oplus Q = (g^{a'Q'} \oplus Q') \oplus Q$,

$X_2^* = h(ID_A,\ Y_E,\ K_1^*) \oplus h(X_3^*)$ from $X_3^* = g^{b"Q"} mod\ n$, and

$Y_2^* = h(ID_B,\ X_E,\ K_2^*) \oplus h(Y_3^*)$ from $Y_3^* = g^{a"Q"} mod\ n$.

They then send X_2^* and Y_2^* to each other for verification. E intercepts this information, and tries to find the appropriate values by the guessing attack. However, E cannot obtain the result. It is very difficult to solve the equation $K_1^* = g^{abQ} mod\ n$ without knowing a, b, and Q, because it meets the Diffie-Hellman problem [92]. In addition, A and B can detect the modified value X_E and Y_E from the key validation phase. As an example, A sends X_2^* to B, then B computes $X_3' = g^{bQ} mod\ n$ and verifies the validity of X_2^* with $h(ID_A,\ Y_1,\ K_2^*) \oplus h(X_3')$. B finds out that they are not matched, because of $Y_E \neq Y_1$ and $h(X_3^*) \neq h(X_3')$, even though K_1^* and K_2^* are the same. Therefore, our proposed scheme is secure against the off-line guessing attack.

5.5.2. Man-In-The-Middle Attack

If E wants to modify the value by using the man-in-the-middle attack, he can obtain and replace the information, such as X_1 to X_E, and Y_1 to Y_E. However, $E's$ attack still ends in failure. With the reason aforementioned, B makes out that they are not equal through the key validation phase, because Y_E is not equal to his own value Y_1, and $h(X_3^*)$ is not equal to $h(X_3')$. Furthermore, they already checked the values of $K_1 \neq 1$ and $K_2 \neq 1$ as a measure against as our man-in-the-middle attack. Thus, our proposed scheme withstands the man-in-the-middle attack.

5.5.3. Reflection Attack

Whenever E tries to attack with the reflection attack, he can intercept and replace the information such as $X_1 = g^{aQ} \oplus Q$ with $X_E = g^{a'Q'} \oplus Q'$, in Step EE1. After that, A computes the session key $K_1^* = Y^{*a} mod\ n = (g^{b'Q'} \oplus Q' \oplus Q)^a mod\ n$ from $Y^* = Y_E \oplus Q = (g^{b'Q'} \oplus Q') \oplus Q$, and $X_2^* = h(ID_A,\ Y_E,\ K_1^*) \oplus h(X_3^*)$ from $X_3^* = g^{b"Q"} mod\ n$. He then sends X_2^* to B for validation in Step EV1. E intercepts X_2^*, and tries to seek the suitable value, but it is hard to find a, b, and Q from the value $X_2^* = h(ID_A,\ Y_E,\ K_1^*) \oplus h(X_3^*)$ by using the information that E has, such as $X_1 = g^{aQ} \oplus Q$. Moreover, after Step EV1, E computes $Y_2^* = h(ID_B,\ X_E,\ K_2^*) \oplus h(Y_3^*)$ from $Y_3^* = g^{a"Q"} mod\ n$, and sends Y_2^* to A. A can detect that this phase is modified. If E sends Y_2^*, then A computes $Y_3' = g^{aQ} mod\ n$, and verifies the validity of Y_2^* with $h(ID_B,\ X_1,\ K_1^*) \oplus h(Y_3')$. Finally, A finds out that they are not matched, because of $X_E \neq X_1$ and $h(Y_3^*) \neq h(Y_3')$. In addition, if E does not respond to A after Step EV1, then A recognizes that this key agreement

protocol is modified or realizes that something is wrong. He then discards this session key. Consequently, our proposed scheme can resist the reflection attack.

5.6. Conclusions

In this chapter, we have shown that the Hsu et al.'s scheme suffers from the off-line guessing attack, and the Lee-Lee's scheme is still vulnerable to off-line guessing, man-in-the-middle, and reflection attacks. We then propose an enhanced secure scheme to eliminate those security flaws.

CHAPTER 6. TWO SIMPLE ATTACKS ON THE ID-BASED PASSWORD AUTHENTICATION SCHEME USING SMART CARDS AND FINGERPRINTS

Recently, Kim, Lee, and Yoo proposed an ID-based password authentication scheme [83], a timestamp based scheme and a nonce based scheme. The proposed schemes do not require a dictionary of passwords or verification tables. These are based on the concepts of ID-based schemes, fingerprint verification systems, and smart cards. They proposed that their schemes would be secure against off-line guessing, message replay, and impersonation attacks. Nevertheless, we show that the Kim-Lee-Yoo (KLY) protocol is still vulnerable to the replay attack and the impersonation attack [53].

6.1. KLY Scheme

This scheme is described in [83] and it comes with three phases: registration, login, and verification. We first introduce the notation used to describe the protocols. Then, we briefly show these protocols.

6.1.1. Notations

- n, g, and f denote public elements,
  where n is a large prime number, g is a generator of order $n-1$ in Z_n^*, and f is a one-way function.
- U_i denotes each legal user and E denotes the adversary.
- S denotes the remote system and SC denotes the smart card.
- $ID_i/PW_i/CID_i/FP_i$ denote the identity/password/SC's identifier/fingerprint of U_i.
- SK denotes the secret key maintained by the remote system.
- T_i/T_s denote the current timestamp of the input device/remote system.
- r_i/r_s denotes the random number generated by the smart card/remote system.

6.1.2. Timestamp based authentication scheme

1. **Registration phase** In the registration phase, the remote system issues smart cards to the users who request registration. In this phase, the remote system is not responsible for authenticating users. It is responsible for generating key information, issuing smart cards to new users, and serving password-changing requests for registered users.

   R1. $U_i \Longrightarrow S$: ID_i and PW_i.

R2. S generates CID_i, and computes S_i and h_i,

where $S_i = ID_i^{(SK)}(mod\ n)$,

$h_i = g^{(PW_i \cdot SK)}(mod\ n)$.

CID_i is for validating the legality of smart cards in the verification phase. Even though, ID_i and n are known, it is hard to find SK from S_i because of the discrete logarithm problem.

R3. $S \longrightarrow U_i$: SC.

S writes $n, g, f, ID_i, CID_i, S_i$, and h_i to SC and issues the card to U_i.

This registration phase is only required when new users request to join or registered users request to change their passwords.

R4. $U_i \longrightarrow SC : FP_i$.

The user registers his fingerprint on his own smart card based on minutia extraction and authenticates his ownership by matching the fingerprint with the use of minutia in [108].

2. **Login phase** In the login phase, the user attaches his smart card to the terminal, and keys in his identifier and password. The user, then, imprints his fingerprint on the fingerprint input device that causes the terminal to send a login request message to the remote system.

L1. $U_i \longrightarrow SC : ID_i$, PW_i, and FP_i.

L2. SC generates a random number r using the co-ordinates of input fingerprint minutia. The fingerprint is used to determine user ownership of SC. Then, it computes

$X_i = g^{r \cdot PW_i}(mod\ n)$,

$Y_i = S_i \cdot h_i^{r \cdot f(CID_i, T_i)}(mod\ n)$,

where $f(x, y)$ is a one-way function.

Whenever there is a fingerprint input, a different map of minutia is made and the input map is used as a one-time random number.

L3. $SC \longrightarrow S : M = \{ID_i, CID_i, X_i, Y_i, T_i\}$

3. **Verification phase** In the verification phase, the remote system verifies the correctness of submitted messages and determines whether the login request should be accepted.

V1. S verifies ID_i and CID_i, if unsuccessful, S rejects the login request.

V2. S checks that $\Delta T \geq (T_s - T_i)$,

where ΔT is the expected legal time interval for transmission delay.

If it is not satisfied, S stops the request.

V3. S checks whether the following equation holds

$$Y_i^{SK^{-1}} \equiv ID_i \cdot X_i^{f(CID_i, T_i)} (mod\ n)$$

If this equation holds, U_i is allowed to login to the remote system. If not, the login request is rejected.

6.1.3. Nonce based authentication scheme

The nonce based scheme also consists of three phases: registration, login, and verification. The registration phase is the same as that of the timestamp based scheme. Therefore, we shall only describe the login and verification phases. In this scheme, timestamp T is replaced with a nonce N as a measure against replay attacks.

1. **Login phase**

   L1. $U_i \longrightarrow S : M_1 = \{ID_i, CID_i\}$, initial login request.

   L2. $S \longrightarrow SC : N = f(CID_i, r_s)$.

   S verifies the validity of ID_i and CID_i. S reserves ID_i and CID_i for the verification phase, if its validity is satisfied. Next, S makes a new nonce $N = f(CID_i, r_s)$ and sends it back to SC.

   L3. SC checks the received N, if it is a new one, SC generates r_i using co-ordinates of input fingerprint minutia and computes

   $$X_i = g^{r_i \cdot PW_i} (mod\ n),$$

   $$Y_i = S_i \cdot h_i^{r_i \cdot N} (mod\ n).$$

   L4. $SC \longrightarrow S : M_2 = \{X_i, Y_i\}$, login request.

2. **Verification phase** In this phase, the remote system verifies the correctness of submitted authentication messages sent by a legitimate user and determines whether the user's login request is accepted.

   V1. S checks whether the following equation holds

   $$Y_i^{SK^{-1}} \equiv ID_i \cdot X_i^N (mod\ n).$$

   If the user uses the correct password that matches the registered key in the smart card and uses the correct nonce that is identical to the one the remote system generated, the login request is accepted. If not, it will be rejected.

6.2. Our Attack

6.2.1. Replay Attack

In the timestamp based authentication scheme, suppose that the adversary intercepts the login message $M = \{ID_i, CID_i, X_i, Y_i, T_i\}$ in Step L3 and blocks the communication. Then, the adversary directly sends M to the remote system. After that, the remote system verifies ID_i and CID_i, tests that $\Delta T \geq (T_s - T_i)$, and finally checks whether the equation $Y_i^{SK^{-1}} \equiv ID_i \cdot X_i^{f(CID_i, T_i)} (mod\ n)$ is correct. In the login phase, there is only one-time and one-way transaction, and no calculation of any information. This means that if the adversary intercepts the message, blocks the communication, and directly sends the message to the remote system, then the time difference test passes. In this phase, the time difference test is the most critical. Therefore, this verification procedure is easily circumvented and the adversary can login access successfully.

6.2.2. Impersonation Attack

If the adversary can obtain someone's smart card, then he will have information such as ID_i, CID_i, S_i, and h_i. The adversary registers and logins to the remote system with his own ID_E, PW_E, FP_E, CID_E, etc. Next, he can compute $X_E = g^{PW_E * r_E} (mod\ n)$ with r_E, a randomly generated number by the adversary. After that, the adversary starts to attack this protocol by impersonating as legitimate user U_i. The attack procedures are as follows.

A1. $E \longrightarrow S : M_1 = \{ID_i, CID_i\}$.

E sends ID_i and CID_i to the remote system for the initial login request.

A2. $S \longrightarrow E : N' = f(CID_i, r'_s)$.

S verifies the validity of ID_i and CID_i, and then reserves ID_i and CID_i for the verification phase. If its validity is satisfied, then S makes a new nonce $N' = f(CID_i, r'_s)$ and sends it back to E.

A3. E computes his own value R, X_E, and Y_E, where

$R = ID_i \cdot X_E^{N'} (mod\ n)$,

$X'_E = (R/ID_i)^{N'^{-1}} (mod\ n)$,

$Y_E = S_i \cdot h_i^{r_E \cdot N'} = (ID_i \cdot g^{PW_E \cdot r_E \cdot N'})^{SK} = (ID_i \cdot X_E^{N'})^{SK} (mod\ n)$.

A4. $E \longrightarrow S : M_2 = \{X'_E, Y_E\}$, login request.

A5. S checks $Y_E^{SK^{-1}} \equiv ID_i \cdot X_E'^{N'} (mod\ n)$.

In the left equation,

$Y_E^{SK^{-1}} = \{(ID_i \cdot X_E^{N'})^{SK} (mod\ n)\}^{SK^{-1}} = ID_i \cdot X_E^{N'} (mod\ n) = R$.

In the right equation,

$ID_i \cdot X_E'^{N'} (mod\ n) = ID_i \cdot \{(R/ID_i)^{N'^{-1}}\}^{N'} (mod\ n) = R$.

Since this equation is valid, the remote system accepts the login request. Even though the adversary does not know the user's password, the attacker can continue to impersonate the user. In addition, if the remote system makes a new nonce $N' = f(CID_i, r'_s)$ in the login phase, the attacker can still continue to impersonate the user. Therefore, their scheme is still vulnerable to the impersonation attack.

6.3. Conclusions

In this chapter, we have shown that the ID-based password authentication scheme using smart cards and fingerprints proposed in [83] is vulnerable to replay and impersonation attacks. In addition, it can be attacked without having the password, timestamp, and fingerprint.

CHAPTER 7. SECURITY IMPROVEMENTS ON A NEW USER REMOTE AUTHENTICATION SCHEME

In 2003, Wu-Chieu presented a user friendly remote authentication scheme [84], with smart cards, without using a user password table. However, Hwang-Liao [85] and Hwang et al. [86] pointed out that the Wu-Chieu scheme [84] is insecure. Recently, Wu-Chieu [87] proposed a modified scheme to overcome the weakness. Nevertheless, we show in this chapter that their new scheme is still vulnerable to the compromised attack and the impersonation attack. We then propose an improved scheme to overcome not only the weakness of the Wu-Chieu scheme [87], but also the password guessing attack [54].

7.1. Wu-Chieu Scheme

We first introduce the notation used to describe the protocols, and then briefly show the scheme in [87].

7.1.1. Notations

- U_i, S, SC, and E denote the user, the remote system, the smart card, and the adversary.

- PW_i denotes user $U_i's$ password.
- x denotes a secret key maintained by S.
- p denotes a large prime number.
- g denotes a public and primitive number in GF(p).
- T denotes the current date and time of the input device.

The user remote authentication scheme described in [87] comes with three phases: the registration, the login, and the authentication phases. These phases are described below.

7.1.2. Registration Phase

In the registration phase, user U_i submits his identifier ID_i and his chosen password PW_i to the system in person, or over a secure channel. The procedures of this phase are as follows:

R1. $U \Longrightarrow S$: ID_i and PW_i.

R2. S computes $A_i = h(ID_i, x)$ and $B_i = g^{A_i \cdot h(PW_i)} (mod\ p)$. Next, S personalizes the smart card with the secure information : $\{ID_i,\ B_i,\ h(\cdot),\ p,\ g\}$.

7.1.3. Login Phase

When U_i wants to login, he inserts his smart card into the card reader, and keys in his ID_i and PW_i^*. Then the smart card performs the following operations:

L1. $U \longrightarrow SC \ : \ ID_i$ and PW_i^*.

L2. $SC \longrightarrow S \ : \ M = \{ID_i, \ C_1, \ D_i^*, \ T\}$.

SC computes $C_1 = h(T \oplus B_i)$ and $D_i^* = g^{h(PW_i^*)}(mod\, p)$ and sends a message M to S.

7.1.4. Authentication Phase

In this phase, after receiving the message M at time T', the remote system S performs the following procedures for authentication U_i.

A1. S verifies the format of ID_i. If it is correct, S verifies the validity of the time interval between T and T'. If $\Delta T \leq (T' - T)$, S rejects the login request.

A2. S computes $A_i = h(ID_i, x)$, $B_i^* = (D_i^*)^{A_i}(mod\, p)$, and $C_1^* = h(T \oplus B_i^*)$. Next, S compares C_1 with C_1^*. If they are equal, S accepts the login request.

7.2. Our Attack

We show two possible attacks against [87]; compromise attack and impersonation attack.

7.2.1. Compromised Attack

To attack the new scheme of [87], we suppose that the adversary E monitors their communications and eavesdropping any message $M = \{ID_i, \ C_1, \ D_i^*, \ T\}$.

Once the long-term secret key x, maintained by S is compromised, then all of the previous keys (important information) will be released. It is described as follows:

EC1. $E \longrightarrow S \ : \ M_E = \{ID_i,\ C_E,\ D_i^*,\ T_E\}$.

Adversary E obtains $ID_i,\ C_1,\ D_i^*$, and T from the eavesdropping any message $M = \{ID_i,\ C_1,\ D_i^*,\ T\}$ in Step L2. After that, he can compute $A_i = h(ID_i, x)$ with the compromised $S's$ secret key x. He then calculates $B_i^* = (D_i^*)^{A_i}(mod\ p)$ and $C_E = h(T_E \oplus B_i^*)$, where T_E is the current date and time when E logs in. Next, E sends the message $M_E = \{ID_i,\ C_E,\ D_i^*,\ T_E\}$ that was made up of his own value C_E and T_E for login to S.

EC2. S verifies the format of ID_i. If it is correct, S verifies the validity of the time interval between T_E and T'. Since the format of ID_i and $\Delta T \geq (T' - T_E)$ are correct, S accepts the login request.

EC3. S computes $A_i = h(ID_i, x)$, $B_i^* = (D_i^*)^{A_i}(mod\ p)$, and $C_1^* = h(T_E \oplus B_i^*)$. Next, S compares C_E with C_1^*. Finally, they are equal, S accepts the login request.

Thus, once the long-term secret key is compromised, adversary E can obtain information that was needed for cracking the authentication system, such as A_i, B_i, and etc. After that, if E wants to attack again, he can successfully conclude his attack at any time. Therefore, the proposed scheme in [87] still suffers from the compromised attack.

7.2.2. Impersonation Attack

In this attack, we assume that the adversary E can steal or pick up $U_i's$ smart card. He can then copy and get information, such as $ID_i,\ B_i,\ h(\cdot),\ p$, and g from the smart card, and return the smart card to U_i. A detailed attack method is described as follows:

E1. $E \longrightarrow S\ :\ M_E = \{ID_i,\ C_E,\ D_i^*,\ T_E\}$.

Adversary E, eavesdrops the message $M = \{ID_i,\ C_1,\ D_i^*,\ T\}$ in Step L2. Next, he computes $C_E = h(T_E \oplus B_i)$ by using B_i that was obtained from the smart card, and then changes the message $M = \{ID_i,\ C_1,\ D_i^*,\ T\}$ to $M_E = \{ID_i,\ C_E,\ D_i^*,\ T_E\}$, where T_E is the current date and time when E logs in.

The remaining attack procedures are the same as $EC2$ and $EC3$. From $D_i^* = g^{h(PW_i)}(mod\ p)$, it is infeasible to calculate the password of the legitimate user, PW_i, because of the discrete logarithm problem. Even though, the adversary does not know the authorized user's password, he can impersonate a legal user. Because he just changes C_1 to C_E, and T to T_E, respectively, and then uses the value of D_i^* for computing B_i^*. Thus, E who obtains any message $M = \{ID_i,\ C_1,\ D_i^*,\ T\}$ can be any user at any time, and successfully conclude his attack. Therefore, the proposed scheme in [87] still suffers from impersonation attack.

7.3. Our Improved Scheme

We make an improvement on [87] by taking into account the compromise attack and the impersonation attack. E monitors their communications, and eaves-

drops the message $M = \{ID_i,\ C_1,\ D_i^*,\ T\}$. Next, he obtains the information he needs. Here we notice the weakness. If E can compromise S's secret key x, or obtain information B_i from the smart card, then he can replace C_1 and T, with C_E and T_E, respectively. Next, E sends the message $M_E = \{ID_i,\ C_E,\ D_i^*,\ T_E\}$ for login to S at any time. Therefore, we suggest the improved scheme as follows.

7.3.1. Registration Phase

In the registration phase of [87], user U_i submits his identifier ID_i and his chosen password PW_i to the system in person, or over a secure channel. However, the server computes information, and stores some of them in its database. The procedures of this phase are as follows:

R'1. $U \Longrightarrow S\ :\ ID_i$ and PW_i.

R'2. S computes $A_i = h(ID_i, x)$, $B_i = g^{A_i \cdot PW_i \cdot h(PW_i)} (mod\, p)$, and $C_i = h(ID_i, PW_i)$. Next, S stores ID_i and C_i in $S's$ database, and then personalizes the smart card with the secure information : $\{ID_i,\ B_i,\ h(\cdot),\ p,\ g\}$.

7.3.2. Login Phase

In the login phase of [87], when U_i wants to login, he inserts his smart card into the card reader, and keys in his ID_i^* and PW_i^*. The smart card then performs the following operations:

L'1. $U \longrightarrow SC\ :\ ID_i^*$ and PW_i^*.

L'2. $SC \longrightarrow S\ :\ M = \{ID_i^*,\ Q_i,\ K_i,\ T\}$.

SC computes $Q_i = h(T \oplus B_i \oplus C_i^*)$ and sends message M to S, where $C_i^* = h(ID_i^*, PW_i^*)$ and $K_i = g^{PW_i^* * h(PW_i^*)} (mod\ p)$.

7.3.3. Authentication Phase

In this phase, after receiving the message M at time T', the remote system S, performs the following procedures for authentication U_i.

A'1. S verifies the format of ID_i. If it is correct, S verifies the validity of the time interval between T and T'. If $\Delta T \leq (T' - T)$, S rejects the login request.

A'2. S computes $A_i^* = h(ID_i^*, x)$, $B_i^* = (K_i)^{A_i^*} (mod\ p)$, and $Q_i^* = h(T \oplus B_i^* \oplus C_i)$ by using C_i in its database. Next, S compares Q_i with Q_i^*. If they are equal, S accepts the login request.

7.4. Security Analysis

In this section, we briefly explain why the proposed scheme is secure against the compromise attack and the impersonation attack.

7.4.1. Compromised Attack

Even though, once the long-term secret key x, maintained by S is compromised, and E obtains $ID_i,\ Q_i,\ K_i$, and T from the eavesdropping any message $M = \{ID_i^*,\ Q_i,\ K_i,\ T\}$ in Step L'2, only two information will be released such as $A_i = h(ID_i^*, x)$ and $B_i^* = (K_i)^{A_i} (mod\ p)$. Next, he wants to change message

$M = \{ID_i^*,\ Q_E,\ K_i,\ T_E\}$ with his own value Q_E and T_E, and sends M to S in Step L'2. However, he cannot get the important information $Q_E = h(T_E \oplus B_i \oplus C_i^*)$, because he does not know $C_i^* = h(ID_i^*, PW_i^*)$. Consequently, this attack is failed. In addition, if E sends message $M = \{ID_i^*,\ Q_i,\ K_i,\ T_E\}$ to S without modification, then S knows $Q_i = h(T \oplus B_i \oplus C_i^*)$ is not equal to $Q_i^* = h(T_E \oplus B_i \oplus C_i)$ in Step A'2. $E's$ log in request is also rejected. Thus, E cannot attack our scheme using this attack.

7.4.2. Impersonation Attack

If the message M was eavesdropped in Step L'2, then E can only obtain ID_i and K_i without knowing the password. Next, he would want to change message $M = \{ID_i^*,\ Q_E,\ K_i,\ T_E\}$ to his own value Q_E and T_E in Step L'2. However, according to the above compromised attack, he cannot get the important information $Q_E = h(T_E \oplus B_i \oplus C_i^*)$. Consequently, this attack is failed. Moreover, if E sends message $M = \{ID_i^*,\ Q_i,\ K_i,\ T_E\}$ to S without modification, after receiving M, S also knows Q_i is not equal to Q_i^* in Step A'2. Therefore, E cannot attack our scheme using the impersonation attack.

7.4.3. Password Guessing Attack

Now we check to see why "password guessing" does not work, and here are the reasons: If the long-term secure key is compromised and the attacker can get B_i, then the attacker computes $A_i = h(ID_i, x)$. He then tries to guess a password PW' from $B_i = g^{Ai*PW'*h(PW')}(mod\ p)$. If the attacker intercepts K_i of a user instead, he

then tries to get password PW' from $K_i = g^{PW' * h(PW')} (mod\ p)$ using the password guessing attack. However, those are difficult to guess the password PW' unless he can solve the Diffie-Hellman problem [92]. The Diffie-Hellman problem depends on the discrete logarithm problem for its security. It is computationally infeasible to calculate $k = g^{ab} mod\ p$ given the two public values $g^a mod\ p$ and $g^b mod\ p$ when the prime p is sufficiently large. Therefore, our proposed scheme is secure against the password guessing attack and its security depends on the security of the Diffie-Hellman.

7.5. Conclusions

In this chapter, we show that the new protocol given in [87] still suffers from the compromised attack and the impersonation attack. We then propose a new scheme that overcomes not only the weakness of the Wu-Chieu scheme, but also the password guessing attack.

CHAPTER 8. IMPROVING THE NOVIKOV AND KISELEV USER AUTHENTICATION SCHEME

Novikov and Kiselev [89] proposed an authentication method of a user from a remote autonomous object. Recently, Yang et al. [90] and Awasthi [91] have pointed out that the Novikov-Kiselev scheme is insecure against the man-in-the-middle attack. In this article, we propose an improved version of the Novikov-Kiselev scheme to overcome such vulnerability [55].

8.1. Novikov-Kiselev Scheme

8.1.1. Notations

- U_i, O, and E denote the user, the remote autonomous object, and the adversary.
- ID_i and K denote the user's identifier and the control command.
- (S_{PK_U}, S_{SK_U}) and (S_{PK_O}, S_{SK_O}) denote a pair of session keys of U_i and O.
- T_i denotes the time parameter.

The Novikov-Kiselev authentication scheme described in [89] comes in two stages described below.

8.1.2. The First Stage

The first stage is the pre-tuning of the parameters U_i and O. U_i produces ID_i and synchronizes T_i with the remote object. This processing is executed just once. ID_i and T_0 are produced and stored in the operative memory of O by U_i.

8.1.3. The Second Stage

The second stage is the communication session between U_i and O. The procedures of this stage are as follows:

S1. $U_i \longrightarrow O$: *signal* S.

U_i sends start request signal S to O.

S2. $O \longrightarrow U_i : \ S_{PK_O}$.

O computes a pair of session keys S_{PK_O} and S_{SK_O} by using the RSA algorithm [69]. Next, O sends S_{PK_O} to U_i, and then turns on the timer to record the session beginning at time T_1.

S3. $U_i \longrightarrow O : \ E_{S_{PK_O}}(ID_i, S_{PK_U})$.

U_i generates a pair of session keys S_{PK_U} and S_{SK_U}, and then encrypts ID_i and S_{PK_U} with S_{PK_O} using the encryption function of the RSA algorithm. Next, he sends it to O.

S4. $O \longrightarrow U_i : \ E_{S_{PK_U}}(X)$.

O decrypts the received message with S_{SK_O} using the decryption function. O records T_2, which indicates the time that the message was received, and

checks that $\Delta T = T_2 - T_1$. If $\Delta T \geq T_0$, then this communication is terminated. Otherwise, O checks received ID_i and stored ID_i in its own memory. If they are correct, O encrypts the message X which includes the command K with S_{PK_U}. Next, O sends it to U_i and records the time T_3.

S5. $U_i \longrightarrow O : \ E_{S_{PK_O}}(newID, K)$.

U_i decrypts the received message with S_{SK_U}, and then obtains X. He derives the command K from the message X, and then encrypts the command K and new identifier $newID$ with S_{PK_O}. Next, U_i sends it to O. After that, U_i records the value of $newID$ in his memory and destroys his pair of session keys (S_{PK_U}, S_{SK_U}) and S_{PK_O}.

S6. O checks $\Delta T = T_3 - T_2$. If $\Delta T \geq T_1$, then this communication is terminated. If it is valid, then O decrypts the received message with S_{SK_O} and obtains the command K. Next, O replaces $newID$ in his memory and destroys a pair of session keys (S_{PK_O}, S_{SK_O}). Finally, O executes the command K.

8.2. Yang-Lee-Hsiao Attack

The procedures of attack are briefly described as follows:

In Step S1, the adversary E intercepts the signal S.

In Step S2, E intercepts S_{PK_O}.

In Step S5, E intercepts $E_{S_{PK_O}}(newID, K)$ and replaces it with $E_{S_{PK_O}}(newID', K')$, and then sends it to O.

The adversary can attack by replacing the legal identifier of the user. Therefore, E can easily supplant the legal user.

8.3. Awasthi Attack

8.3.1. The First Stage

The user U_i sends ID_i and T_0 to the object O. On this communication, the adversary E intercepts the information and sends the tuple ID_i and T_* instead of the original one.

8.3.2. The Second Stage

AS1. $U_i \longrightarrow O$: *signal* S.

U_i sends start request signal S to O. E intercepts S and sends it to O.

AS2. $O \longrightarrow U_i : \ S_{PK_O}$.

The object O computes a pair of session keys S_{PK_O} and S_{SK_O} by using the RSA algorithm. Next, O sends S_{PK_O} to U_i, and then turns on the timer to record the session beginning at time T_1.

AS3. $E \longrightarrow U_i : \ S_{PK'_O}$.

E intercepts S_{PK_O} and sends a self generated $S_{PK'_O}$ to the user.

AS4. $U_i \longrightarrow O : \ E_{S_{PK'_O}}(ID_i, S_{PK_U})$.

U_i generates a pair of session keys S_{PK_U} and S_{SK_U}, and then encrypts ID_i and S_{PK_U} with $S_{PK'_O}$ using the encryption function of the RSA algorithm. Next, he sends it to O.

AS5. $E \longrightarrow O : \ E_{S_{PK_O}}(ID_i, S_{PK'_O})$.

E intercepts this encrypted message and decrypts it using $S_{SK'_O}$. He modifies

the encrypted message as $E_{S_{PK_O}}(ID_i, S_{PK'_O})$ and sends it to O.

AS6. $O \longrightarrow U_i : \ E_{S_{PK'_O}}(X)$.

O decrypts the received message with S_{SK_O} using the decryption function. O records T_2, which indicates the time that the message was received, and checks that $\Delta T = T_2 - T_1$. If $\Delta T \geq T_*$, then this communication is terminated. O encrypts the message X which includes the command K with $S_{PK'_O}$. Next, O sends it to U_i.

AS7. $E \longrightarrow U_i : \ E_{S_{PK_U}}(X)$.

E intercepts this message and decrypts $E_{S_{PK'_O}}(X)$ using $S_{SK'_O}$. Next, he encrypts X with S_{PK_U} and sends it to U_i.

AS8. $U_i \longrightarrow O : \ E_{S_{PK'_O}}(newID, K)$.

U_i decrypts the received message with S_{SK_U}, and then obtains X. He derives the command K from the message X, and then encrypts the command K and the new identifier $newID$ with $S_{PK'_O}$. Next, U_i sends it to O.

AS9. $E \longrightarrow O : \ E_{S_{PK'_O}}(newID, K)$.

E intercepts the message $E_{S_{PK'_O}}(newID, K)$ and decrypts it with $S_{SK'_O}$. E can get $newID$ and K. After that, E can make whatever modifications he wants.

8.4. Our Improved Scheme

We will make an improvement on the Novikov-Kiselev scheme by taking into account the man-in-the-middle attack of [90] and [91]. E monitors their communications and then eavesdrops the messages, such as S, S_{PK_O}, and $E_{S_{PK_O}}(newID, K)$.

Next, he sends the replaced messages to O. Here we notice the weakness. Since O did not check the command K, E can obtain S_{PK_O} and replace $E_{S_{PK_O}}(newID, K)$ with $E_{S_{PK_O}}(newID', K')$ at any time. We modify the Novikov-Kiselev scheme as follows:

8.4.1. The Synchronization Phase

The user U_i sends ID_i to the object O. O computes a pair of session keys (S_{PK_O}, S_{SK_O}) by using the RSA algorithm and sends his session public key S_{PK_O} to U_i. Next, they synchronize according to time T_0. During this phase, these transactions are done via a secure communication channel. Next, O stores ID_i and T_0 in its memory.

8.4.2. The Authentication Phase

A1. $U_i \longrightarrow O : E_{S_{PK_O}}(ID_i,\ T_i)$.

U_i checks the current time T_i and encrypts ID_i and T_i by using session public key S_{PK_O}. Next, U_i sends start request signal to O with encrypted message $E_{S_{PK_O}}(ID_i,\ T_i)$.

A2. $O \longrightarrow U_i :\ E_{S_{SK_O}}(T_{j+1},\ X)$.

When O receives the message, as a first step, it records the current time T_j, and then decrypts the received message by using session private key S_{SK_O} and obtains ID_i and T_i. Next, O compares the time $\Delta T_x = T_j - T_i$ with T_0 that is stored in the memory at the synchronization phase. If $\Delta T_x \geq T_0$, then this communication is terminated. If they are valid, O turns on the timer

and records the session beginning time T_{j+1}, and then sends $E_{S_{SK_O}}(T_{j+1},\ X)$ to U_i.

A3. $U_i \longrightarrow O:\ E_{S_{PK_O}}(T_{i+1},\ T^*_{j+1},\ T_{i+2},\ K')$.

When U_i receives the message, he records the current time T_{i+1} with priority, and then computes $D_{S_{PK_O}}[E_{S_{SK_O}}(T_{j+1},\ X)]$ to decrypt and obtain T^*_{j+1} and X. He can derive the command K' from the message X, and then encrypt $T_{i+1},\ T^*_{j+1},\ T_{i+2}$ and K' with S_{PK_O}, where T_{i+2} is the new current time of U_i. Next, U_i sends it to O. After that, U_i destroys session public key S_{PK_O}.

A4. When O receives the message, it begins by recording the current time T_{j+2}, and then computes $D_{S_{PK_O}}[E_{S_{PK_O}}(T_{i+1},\ T^*_{j+1},\ T_{i+2},\ K')]$ to decrypt and obtain $T_{i+1},\ T^*_{j+1},\ T_{i+2}$ and K'. Next, O checks that T^*_{j+1} equals to T_{j+1} that was sent by O in Step A2. If they are the same, then O compares the time $\Delta T_y = T_{j+1} - T_{i+1}$ and $\Delta T_z = T_{j+2} - T_{i+2}$ with T_0. If $\Delta T_y \geq T_0$ or $\Delta T_z \geq T_0$, then this communication is terminated. If both are valid, then O checks the received commands K' and K that were sent in Step A2. If they are correct, O executes the command K and destroys a pair of session keys (S_{PK_O}, S_{SK_O}).

8.4.3. The Change ID Phase

C1. - C2. First two Steps are the same as the authentication phase $A1$ and $A2$.

C3. $U_i \longrightarrow O:\ E_{S_{PK_O}}(T_{i+1},\ T^*_{j+1},\ T_{i+2},\ K',\ old\ ID_i,\ new\ ID_i)$.

When U_i receives the message, he records the current time T_{i+1} with priority, and then computes $D_{S_{PK_O}}[E_{S_{SK_O}}(T_{j+1},\ X)]$ to decrypt and obtains T^*_{j+1} and X. He can derive the command K' from the message X, and then encrypts

$T_{i+1},\ T^*_{j+1},\ T_{i+2},\ K',\ old\ ID_i$ and $new\ ID_i$ with S_{PK_O}, where T_{i+2} is the new current time of U_i. Next, U_i sends it to O.

C4. $O \longrightarrow U_i :\ E_{S_{SK_O}}(T_k,\ new\ ID^*_i)$.

When O receives the message, it begins by recording the current time T_{j+2}, and then computes $D_{S_{SK_O}}[E_{S_{PK_O}}(T_{i+1},\ T^*_{j+1},\ T_{i+2},\ K',\ old\ ID_i,\ new\ ID_i)]$ to decrypt and obtain $T_{i+1},\ T^*_{j+1},\ T_{i+2}$ and K'. Next, O checks that T^*_{j+1} equals to T_{j+1} that was sent by O in Step C2. If they are correct, then O compares the time $\Delta T_y = T_{j+1} - T_{i+1}$ and $\Delta T_z = T_{j+2} - T_{i+2}$ with T_0. If $\Delta T_y \geq T_0$ or $\Delta T_z \geq T_0$, then this communication is terminated. If both are valid, then O checks the received commands K' and K that was sent in Step C2. If they are correct, O changes old ID_i to new ID_i. After that, O encrypts $new\ ID^*_i$ and T_k with S_{SK_O} and sends it to U_i, where T_k is the new current time of O. Next, O destroys a pair of session keys (S_{PK_O}, S_{SK_O}).

C5. When U_i receives the message, he records the current time T_{k+1} with priority, and then decrypts the message with S_{PK_O} and obtains T_k and $new\ ID^*_i$. Next, U_i compares the time $\Delta T_c = T_{k+1} - T_k$ with T_0 that was stored in the memory at the synchronization phase. If $T_0 \geq \Delta T_c$ and $new\ ID^*_i = new\ ID_i$, then U_i destroys session public key S_{PK_O}. If they are invalid, U_i considers that this communication was forged and he sets up $new\ ID'_i$ and a pair of new session keys (S'_{PK_O}, S'_{SK_O}) again through the synchronization phase.

8.5. Security Analysis

In this section, we briefly explain why the proposed scheme is secure against the man-in-the-middle attack and also more efficient.

In the synchronization phase, U_i and O transact the user's session public key S_{PK_O} and synchronize time T_0 via a secure communication channel. Even though E intercepts the start signal and $E_{S_{PK_O}}(ID_i,\ T_i)$ in Step A1, he cannot change or replace this information, since he neither knows nor is able to intercept S_{PK_O}. This can prevent the man-in-the-middle attack of [90] and [91].

In addition, we substitute the encrypted new ID_i and T_k with S_{PK_O} in our scheme in order to improve security. If E changes the value of newID_i or K to his own value newID_i^* or K^*, respectively, it should be known to U_i in Step C5. This is because U_i compares the time $\Delta T_c = T_{k+1} - T_k$ with T_0 that was stored in the memory at the synchronization phase. If $\Delta T_c \geq T_0$ or $new\ ID_i^* \neq\ new\ ID_i$, then this communication is terminated. Therefore, E cannot obtain new ID_i without S_{PK_O}.

In the Yang-Lee-Hsiao scheme, E can eavesdrop and replace the message $E_{S_{PK_O}}(newID, K)$ with $E_{S_{PK_O}}(newID', K')$ in Step S5. It is impossible to replace this message without knowing S_{PK_O} in our scheme. However, even though it occurs in our scheme, O checks the time difference $\Delta T_y = T_{j+1} - T_{i+1}$ and $\Delta T_z = T_{j+2} - T_{i+2}$ with T_0 in Step C4. If $\Delta T_y \geq T_0$ or $\Delta T_z \geq T_0$, then this communication is terminated. Moreover, U_i checks that the time $T_0 \geq \Delta T_c$ and $new\ ID_i^* =\ new\ ID_i$ in Step C5. If they are invalid, U_i will find out that this communication was forged and he will set up $new\ ID_i'$ and a pair of new session keys (S'_{PK_O}, S'_{SK_O}) again through the synchronization phase. E does not have enough time to eavesdrop and to perform replacement within ΔT_y and ΔT_z. Therefore, E cannot attack our scheme.

Thus, our proposed scheme is more secure against Yang-Lee-Hsiao's man-in-the-middle attack.

8.6. Cost Comparisons

We compare the computational cost of the Novikov-Kiselev scheme with our proposed scheme. We define some notations and show the comparative results as follows.

- DT: Data Transmission,
- E/D: Encryption/ Decryption,
- SK_O/SK_U: A pair of Session Keys of the Object/ User

Novikov-Kiselev	**Our Authentication**	**Our Change ID**
$5DT + 3E + 3D$	$3DT + 3E + 3D$	$4DT + 4E + 4D$
SK_O	SK_O	SK_O
SK_U	X	X

Since the command is formalized and O does not recheck command K' in [89], E is able to replace $E_{PK_O}(newID, K)$ with $E_{PK_O}(newID', K')$ at any time. Moreover, whenever U_i wants to use O, U_i not only has to change his old ID_i, but also has to create and record *new* ID_i in his memory. However, our scheme is able to reduce the time and the storage that were needed in the presence of $newID$, if U_i does not want to change his old ID_i. In the real world, not many people would want to change their IDs frequently. Furthermore, our scheme does not need to generate and store a pair of session keys for the users.

8.7. Conclusions

In this paper, we propose a new scheme that overcomes the weakness of the Novikov-Kiselev scheme. Our scheme is more efficient; it requires less time and storage space provided that the users do not change their IDs.

CHAPTER 9. A SECURE HASH-BASED STRONG-PASSWORD AUTHENTICATION PROTOCOL USING ONE-TIME PUBLIC-KEY CRYPTOGRAPHY

Secure communication is an important issue in networks and user authentication is a very important part of the security. Several strong-password authentication protocols have been introduced, but there is no fully secure authentication scheme that can resist all known attacks. Recently, a hash-based strong-password authentication scheme was described in [28], which withstands several attacks, including replay, password-file compromise, denial-of-service, and insider attacks. However, more recently, Kim-Koc [50] showed that the Ku's scheme [28] is still vulnerable to stolen-verifier, denial-of-service, replay, and impersonation attacks. In this chapter, we propose enhanced secure schemes with registration and login protocols, and add the "forget password" and password/verifier change protocols. We show that our scheme is more secure against guessing, stolen-verifier, replay, denial-of-service, and impersonation attacks than previously introduced protocols.

9.1. The Proposed Scheme

There are some cases for our scheme: 1) If someone forgets his password, he should use the "forget password" protocol. 2) If the user just wants to change

password, then he should use the password/verifier change protocol. This protocol should be used after a user logs in successfully. 3) Lastly, if someone wants to change user ID and password, then he should use the register protocol.

9.1.1. Notations

- $E_{S_{pu}}$ denotes the encryption with the public key of the server.
- $D_{S_{pr}}$ denotes the decryption with the private key of the server.
- K_u is a random generated key selected by U.
- P represents the strong password of U.
- T_s denotes the timestamp.
- $Auth_Q/Auth_A$ denotes the authentication question/answer for the registration, "forget password" and password/verifier change protocols.

9.1.2. Registration Protocol

R1. $U \longrightarrow S : PV = h(K_u||P) \oplus K_u$

U inputs his ID, password, and private key into the client system. The client system computes the user's password verifier $PV = h(K_u||P) \oplus K_u$, and sends it to S for a registration request.

R2. $S \longrightarrow U : R, Auth_Q$

S stores PV and computes $R = PV \oplus T_s$. Next, S sends R and $Auth_Q$ to U.

R3. $U \longrightarrow S : E_{S_{pu}}(UV,\ T'_s,\ U_{id},\ K'_u,\ P',\ Auth_Q \oplus Auth_A)$

U derives T'_s by XORing R with PV, and computes the user's important verifier $UV = h(K'_u||P'||T'_s||U_{id}) \oplus K'_u$. Next, U encrypts $UV,\ T'_s,\ U_{id},\ K'_u,\ P'$ and $Auth_Q \oplus Auth_A$ with S's public key, and sends it to S.

R4. S decrypts $D_{S_{pr}}(E_{S_{pu}}(UV,\ T'_s,\ U_{id},\ K'_u,\ P',\ Auth_Q \oplus Auth_A))$ and derives $UV,\ T'_s,\ U_{id},\ K'_u,\ P'$, and $Auth_Q \oplus Auth_A$. S computes $h(K'_u||P') \oplus K'_u$. S then compares $h(K'_u||P') \oplus K'_u$ and T'_s with PV and T_s, respectively, that were stored and sent in Step R2. If both are equal, then S stores the sealed-verifier $SV = h(K'_u||P'||U_{id}) \oplus K_{pr},\ PV,\ UKP = E_{S_{pu}}(U_{id},\ K'_u,\ P')$, and $QAK = Auth_Q \oplus Auth_A \oplus K_{pr}$ in his password file, where K_{pr} is the server's private key.

9.1.3. Login Protocol

L1. $U \longrightarrow S : PV' = h(K'_u||P') \oplus K'_u$

U inputs his ID, password, and private key into the client system. The client system computes the user's password verifier $PV' = h(K'_u||P') \oplus K'_u$, and sends it to S for a login request.

L2. $S \longrightarrow U : PV,\ r_s$

S compares PV' with the PV that was stored in R2. If they are equal, then S generates a random nonce r_s, and then sends $PV$and r_s to U.

L3. $U \longrightarrow S : L$

U compares PV' with PV. If they are equal, then U computes $h(K_u||P||U_{id})$. Next, U computes $L = h(h(K_u||P||U_{id}) \oplus PV') \oplus h(K_u||P||U_{id}) \oplus PV' \oplus r_s$,

and sends it to S.

L4. S derives $C_1 = h(h(K_u||P||U_{id}) \oplus PV') \oplus h(K_u||P||U_{id}) \oplus PV'$ by XORing L with r_s. S then computes $C_2 = SV \oplus K_{pr} = h(K_u||P||U_{id})$ using the stored SV and K_{pr} in Step R4 and $C_3 = h(C_2 \oplus PV) \oplus C_2 \oplus PV$. Next, S checks $C_1 = C_3$. If they are equal, S authenticates U.

9.1.4. "Forget Password" Protocol

FP1. $U \longrightarrow S$: "forget password" request.

FP2. $S \longrightarrow U : Auth'_Q,\ R_F$

S generates a random nonce R_F, and then sends $Auth'_Q$ and R_F to U.

FP3. $U \longrightarrow S : E_{S_{pu}}(FP,\ U'_{id})$

U computes $FP = Auth'_Q \oplus Auth'_A \oplus R_F$, and encrypts it and U'_{id} with S's public key. Next, S sends $E_{S_{pu}}(FP,\ U'_{id})$ to S.

FP4. $S \longrightarrow U : Auth_A \oplus K'_u,\ Auth_A \oplus P'$

S decrypts $D_{S_{pr}}(E_{S_{pu}}(FP,\ U'_{id}))$, and derives $D_1 = Auth'_Q \oplus Auth'_A$ by XORing FP with R_F. S then derives $D_2 = Auth_Q \oplus Auth_A$ by XORing QAK with K_{pr} that was stored in Step R4. After that, S checks $D_1 = D_2$. If they are equal, S decrypts UKP, $D_{S_{pr}}(E_{S_{pu}}(U_{id},\ K'_u,\ P'))$ that was stored in Step R4. Otherwise, S rejects this request. Next, S derives K'_u, U_{id} and P', and then checks $U'_{id} = U_{id}$. If they are equal, S computes $Auth_A \oplus K'_u$ and $Auth_A \oplus P'$, and then sends these values to U. If not, S terminates this session.

FP5. U obtains the former password P and private key K_u by XORing $Auth_A \oplus K'_u$ and $Auth_A \oplus P'$ with $Auth_A$.

9.1.5. Password/Verifier Change Protocol

PC1. $U \longrightarrow S$: password-change request.

PC2. $S \longrightarrow U$: $Auth'_Q,\ R_C$

S generates a random nonce R_C, and sends $Auth'_Q$ and R_C to U.

PC3. $U \longrightarrow S$: $E_{S_{pu}}(W_1,\ P_{new},\ K_{u_{new}},\ U_{id_{new}})$

U computes $W_1 = Auth'_Q \oplus Auth'_A \oplus R_C \oplus h(K_u||P||U_{id})$, and encrypts W_1 and the new values of $K_{u_{new}},\ P_{new}$, and $U_{id_{new}}$ with S's public key. Next, U sends $E_{S_{pu}}(Auth'_Q \oplus Auth'_A \oplus R_C \oplus h(K_u||P||U_{id}),\ K_{u_{new}},\ P_{new},\ U_{id_{new}})$ to S.

PC4. S decrypts $D_{S_{pr}}(E_{S_{pu}}(W_1,\ K_{u_{new}},\ P_{new},\ U_{id_{new}}))$, and obtains $W_1,\ K_{u_{new}},\ P_{new}$, and $U_{id_{new}}$. S then computes $P_2 = Auth_Q \oplus Auth_A$ by XORing QAK with K_{pr} that was stored in Step R4 and $W_3 = SV \oplus K_{pr} = h(K_u||P||U_{id})$ using the stored SV and K_{pr} in Step R4. Next, S computes $W_4 = W_2 \oplus R_C \oplus W_3$ and checks $W_1 = W_4$. If they are equal, then S stores a new $SV' = h(K_{u_{new}}||P_{new}||U_{id_{new}}) \oplus K_{pr}$, a new $PV' = h(K_{u_{new}}||P_{new}) \oplus K_{u_{new}}$, a new $UKP' = E_{S_{pu}}(U_{id_{new}},\ K_{u_{new}},\ P_{new})$, and $QAK = Auth_Q \oplus Auth_A \oplus K_{pr}$ in his password file.

9.2. Security Analysis

We will briefly demonstrate that the proposed scheme is secure against guessing, stolen-verifier, replay, denial-of-service, and impersonation attacks.

9.2.1. Guessing Attack

By nature, due to the use of a strong password, this scheme is able to resist the off-line guessing attack. Additionally, the user's password is always secretly concealed with the private key K_u within the hash function, since it is hard to find P and K_u from $h(K_u||P) \oplus K_u$. Therefore, no one can reveal the user's password P without U's permission.

9.2.2. Stolen-Verifier Attack

The server stores the verifier of user's password instead of the clear text of the password. In the stolen-verifier attack, the adversary who has stolen the password verifier from the server uses it directly to masquerade as a legitimate user. If the adversary obtains a copy of the password verifier $h(K_u||P) \oplus K_u$ in Step R1, he also can obtain T_s by copying $h(K_u||P) \oplus K_u$ and computing $PV \oplus T_s$ with the previous stolen verifier PV in Step R2. However, the adversary can neither get any information nor manipulate after Step R3 without the server's private key. Since it is hard to find P and K_u in $h(K_u||P) \oplus K_u$, the adversary cannot derive $h(K_u||P||U_{id}) \oplus K_{pr}$ to obtain the sealed-verifier SV from $h(K_u||P) \oplus K_u$. Even though the adversary intercepts password verifiers in Steps R1 and R2, the

adversary cannot use them since there is no way to derive U_{id} before Step R2 for registration. In Step R3, since U_{id} is encrypted with the server's public key, the attacker cannot obtain U_{id} without the server's private key K_{pr}. The adversary also cannot obtain $h(h(K_u||P||U_{id}) \oplus PV)$ for login from $h(K_u||P) \oplus K_u$. Even if the adversary steals $SV = h(K_u||P||U_{id}) \oplus K_{pr}$ and $Auth_Q \oplus Auth_A \oplus K_{pr}$ from the server, he cannot open them without the server's private key. If the adversary obtains the server's private key, he is able to get any information. However, we assumed the server's private key K_{pr} is kept as a top secret on the server. If K_{pr} is released, not only does the server's private key need to be changed, but all users should be re-registered too. Since we use the verifier and other unknown values (e.g. K_{pr} or K_u) together, even if the attacker steals the verifier, he will not use it anywhere without knowing K_{pr} or K_u. Thus, our scheme can resist any stolen-verifier attacks.

9.2.3. Replay Attack

The replay attack is an offensive action in which the adversary impersonates or deceives another legitimate participant through the reuse of information obtained in protocols. It indicates an attempt by an unauthorized third party to record exchanged messages. In Step L3, since U_{id} is hashed with two other unknown values K_u and P, the attacker cannot obtain U_{id} without the knowledge of K_u and P. The adversary is able to steal PV in Step R1 and r_s in Step L2, and then obtain $h(h(K_u^*||P^*||U_{id}) \oplus PV') \oplus h(K_u^*||P^*||U_{id})$. However, he cannot get any information for login, "forget password" and change password protocols. After that, the adversary will try to change $C_1^* = h(h(K_u^*||P^*||U_{id}^*) \oplus PV') \oplus h(K_u^*||P^*||U_{id}^*) \oplus PV'$ with

his own values P^*, U_{id}^* and PV'. However, the server will detect it as modified(i.e. $C_1^* \neq C_2$) in Step L4, since the attacker needs the encrypted values with K_{pr} such as K_u, P, and $Auth_A$. The adversary can steal PV and $Auth_Q$ in Step R1 and R2, respectively. After that, he will use them for the replay attack. However, this attack cannot be successful, since the adversary needs to know the sealed-verifier SV and the server's private key K_{pr} for this attack. Consequently, our proposed scheme can resist a replay attack.

9.2.4. Denial-of-Service Attack

This attack is characterized by the explicit attempt of an attacker to prevent legitimate users of a service from using that service. This attempt includes several different flavors: disrupting service to a specific system or user, preventing a particular user from accessing a service, or denying requests issued by a legitimate user. The adversary, however, is unable to change the user's password without the user's permission in our scheme, since it is hard to find P and K_u not only in $h(K_u||P) \oplus K_u$, but also in $h(h(K_u||P||U_{id}) \oplus K_{pr}) \oplus h(K_u||P||U_{id}) \oplus K_{pr}$. There is no chance to change the password or verifier in Step R3 or L3. Therefore, our improved scheme can resist a denial-of-service attack.

9.2.5. Impersonation Attack

This attack deceives the identity of one of the legitimate parties. An attacker inserts or changes a message and claims that it originated from a real sender. If the adversary impersonates U and wants to get the user's former password P, he

should attack the "forget password" protocol. Since he does not know $Auth_A$, he cannot obtain the password. If the attacker wants to get the password, he needs to know K_{pr} for decryption. Moreover, in our protocol, $Auth_A$ is always protected with a $S's$ private key K_{pr} and other unknown values such as R_C, R_F, and $h(K_u||P||U_{id})$. If the adversary logs in the system successfully, he could try to change it with his own password P^*. However, to change the password, the adversary would need to attack the password/verifier change protocol and know K_u, $Auth_A$, and SV, which is impossible. Thus, our proposed scheme can also resist a impersonation attack.

9.3. Conclusions

In this chapter, we have proposed a secure hash-based strong-password authentication protocol using one-time public-key cryptography that includes not only secure registration and login authentication, but secure "forget password" and password/verifier change protocols. It is more secure against guessing, stolen-verifier, replay, denial-of-service, and impersonation attacks.

CHAPTER 10. CONCLUSION

This dissertation outlines the background and the current state of our researches. We introduce the necessary fundamental background, such as authentication schemes, attacks against protocols, and related works. We then show our works that are published in journals and present our works that are in processing at various journals.

This dissertation is consist of two categories. One is to describe specific cryptanalytic attacks on existing protocols and show their vulnerabilities in order to design more secure protocols, such as chapter 2,3,4, and 6. Another one is to propose improved security schemes to overcome certain security defects, such as 5, 7, 8, and 9.

It is the belief of the author that topic of this dissertation is the most fundamental area for authentication and research here is paramount for the protection against hacking or breaking of authentication protocols. We hope that this dissertation is useful for authentication in many researches, applications, and commercial communication markets.

Bibliography

[1] C. K. Chan and L. M. Cheng, "Cryptanalysis of a remote user authentication scheme using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 4, pp. 992–993, 2000.

[2] M. S. Hwang and L. H. Li, "A new remote user authentication scheme using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 1, pp. 28–30, 2000.

[3] M. Peyravian and N. Zunic, "Methods for protecting password transmission," *Computers & Security*, vol. 19, no. 5, pp. 466–469, 2000.

[4] H. M. Sun, "An efficient remote use authentication scheme with smart cards," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 4, pp. 958–961, Nov. 2000.

[5] M. Sandirigama, A. Shimizu, and M. T. Noda, "Simple and secure password authentication protocol (SAS)," *IEICE Transactions on Communications*, vol. E83-B, no. 6, pp. 1363–1365, June 2000.

[6] H. Y. Chien, J. K. Jan, and Y. M. Tseng, "A modified remote login authentication scheme based on geometric approach," *jss*, vol. 55, pp. 287–290, 2001.

[7] C. L. Lin, H. M. Sun, and T. Hwang, "Attacks and solutions on strong-password authentication," *IEICE Transactions on Communications*, vol. E84-B, no. 9, pp. 2622–2627, Sept. 2001.

[8] C. K. Chan and L. M. Cheng, "Cryptanlysis of timestamp-based password authentication scheme," *Computers & Security*, vol. 21, no. 1, pp. 74–76, 2002.

[9] C. M. Chen and W. C. Ku, "Stolen-verifier attack on two new strong-password authentication protocols," *IEICE Transactions on Communications*, vol. E85-B, no. 11, pp. 2519–2521, Nov. 2002.

[10] L. Fan, J. H. Li, and H. W. Zhu, "An enhancement of timestamp-based password auhentication scheme," *Computers & Security*, vol. 21, no. 7, pp. 665–667, 2002.

[11] C. C. Lee, L. H. Li, and M. S. Hwang, "A remote user authentication scheme using hash functions," *ACM Operating System Review*, vol. 36, no. 4, pp. 23–29, Oct. 2002.

[12] M. S. Hwang, C. C. Lee, and Y. L. Tang, "A simple remote user authentication scheme," *Mathematical and Computer Modelling*, vol. 36, pp. 103–107, 2002.

[13] T. C. Yeh, H. Y. Shen, and J. J. Hwang, "A secure one-time password authentication scheme using smart cards," *IEICE Transactions on Communications*, vol. E85-B, no. 11, pp. 2515–2518, 2002.

[14] A. K. Awasthi and S. Lal, "A remote user authentication scheme using smart cards with forward secrecy," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 1246–1248, 2003.

[15] C. C. Chang, "Some forgery attack on a remote user authentication scheme using smart cards," *International Journal Informatica*, vol. 14, no. 3, pp. 289–294, 2003.

[16] C. L. Hsu, "Security of two remote user authentication schemes using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 1196–1198, 2003.

[17] K. C. Leung, L. M. Cheng, A. S. Fong, and C. K. Chan, "Cryptanalysis of a modified remote user authentication scheme using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 1243–1245, 2003.

[18] C. W. Lin, J. J. Shen, and M. S. Hwang, "Security enhancement for optimal strong-password authentication protocol," *ACM Operating System Review*, vol. 37, no. 2, pp. 7–12, Apr. 2003.

[19] J. J. Shen, C. W. Lin, and M. S. Hwang, "A modified remote user authentication scheme using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 2, pp. 414–416, 2003.

[20] ——, "Security enhancement for the timestamp-based password authentication scheme using smart cards," *Computers & Security*, vol. 22, no. 7, pp. 591–595, 2003.

[21] H. M. Sun and H. T. Yeh, "Further cryptanalysis of a password authentication scheme with smart cards," *IEICE Transactions on Communications*, vol. E86-B, no. 4, pp. 1412–1415, Apr. 2003.

[22] B. Wang, J. H. Li, and Z. P. Tong, "Cryptanalysis of an enhanced timestamp-based authentication scheme," *Computers & Security*, vol. 22, no. 7, pp. 643–645, 2003.

[23] T. H. Chen, W. B. Lee, and G. Horng, "Secure sas-like password authentication schemes," *Computer Standards and Interfaces*, vol. 27, no. 1, pp. 25–31, 2004.

[24] M. L. Das, A. Saxena, and V. P. Gulati, "A dynamid id-based remote user authentication scheme," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 2, pp. 629–631, 2004.

[25] C. L. Hsu, "Security of chien et al.s remote user authentication scheme using smart cards," *Computer Standards and Interfaces*, vol. 26, no. 3, pp. 167–169, 2004.

[26] W. S. Juang, "Efficient multi-server password authenticated key agreement using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 1, pp. 251–255, 2004.

[27] M. Kumar, "New remote user authentication scheme using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 2, pp. 597–600, 2004.

[28] W. C. Ku, "A hash-based strong-password authentication scheme without using smart cards," *ACM Operating System Review*, vol. 38, no. 1, pp. 29–34, Jan. 2004.

[29] S. W. Lee, H. S. Kim, and K. Y. Yoo, "Improved efficient remote user authentication scheme using smart cards," *IEICE Transactions on Communications*, vol. 50, no. 2, pp. 565–567, May.

[30] E.-J. Yoon, E.-K. Ryu, and K.-Y. Yoo, "A secure user authentication scheme using hash functions," *ACM Operating System Review*, vol. 38, no. 2, pp. 62–68, Apr. 2004.

[31] C. C. Yang, H. W. Yang, and R. C. Wang, "Cryptanalysis of security enhancement for the timestamp-based password authentication scheme using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 2, pp. 578–579, 2004.

[32] W. C. Ku, M. H. Chiang, and S. T. Chang, "Weaknesses of Yoon-Ryu-Yoo's hash-based password authentication scheme," *ACM Operating System Review*, vol. 39, no. 1, pp. 85–89, Jan. 2005.

[33] I. Liao, C. C. Lee, and M. S. Hwang, "Security enhancemento for a dynamic id-based remote user authentication scheme," in *International Conference on Next Generation Web Services Practices (NWeSP05)*. IEEE CS Press, Aug 2005, pp. 437–440.

[34] E. J. Yoon, E. K. Ryu, and K. Y. Yoo, "An improvement of hwang-lee-tangs simple remote user authentication schemes," *Computers & Security*, vol. 24, no. 4, pp. 50–56, 2005.

[35] C. C. Yang, R. C. Wang, and T. Y. Chang, "An improvement of the yang-shieh password authentication schemes," *Applied Mathematics and Computation*, vol. 162, no. 3, pp. 1391–1396, 2005.

[36] M. H. Chiang and W. C. Ku, "Weaknesses of two sas-like password authentication schemes," *IEICE Transactions on Communications*, vol. E89-B, no. 2, pp. 594–597, 2006.

[37] M. Kim and Ç. K. Koç, "A simple attack on a recently introduced hash-based secure user authentication scheme," *International Journal of Computer Science and Network Security*, vol. 6, no. 5B, pp. 157–160, May 2006.

[38] I. Liao, C. C. Lee, and M. S. Hwang, "A password authentication scheme over insecure networks," *Journal of Computer and System Sciences*, vol. 72, pp. 727–740, 2006.

[39] R. C. Wang and C. C. Yang, "Cryptanalysis of two improved password authentication schemes using smart cards," *International Journal of Network Security*, vol. 3, no. 3, pp. 283–285, Nov 2006.

[40] M. Matsui, "New block encyrption algorithm MISTY," in *Fast Software Encryption*, E. Biham, Ed. Springer Verlag, LNCS Nr. 1267, 1997, pp. 54–68.

[41] National Institute for Standards and Technology, "Data Encryption Standard (DES)," Oct. 1999, FIPS 46-3.

[42] A. Shimizu and S. Miyaguchi, "Fast data encipherment algorithm FEAL," *IEICE Transactions*, vol. J70-D, no. 7, pp. 1413–1423, July 1987.

[43] M. Strobel, "Design of roles and protocols for electronic negotiations," *Electronic Commerce Research*, vol. 1, no. 3, pp. 335–353, 2001.

[44] O. Chun, M. K. Lars, and B. Jonathan, "A formal service specification for the Internet open trading protocol," in *Applications and Theory of Petri Nets 2002: 23rd International Conference, ICATPN 2002.* Springer Verlag, LNCS Nr. 2360, 2002, pp. 352–373.

[45] V. F. Kleist, "A transaction cost model of electronic trust: Transactional return, incentives for network security and optimal risk in the digital economy," *Electronic Commerce Research*, vol. 4, no. 1-2, pp. 41–57, 2004.

[46] I.-W. Lee, H.-J. Park, and S.-H. Kim, "Development of Electronic Commerce Micropayment System with a Pay-later Payment Method," in *Communication Systems and Applications(CSA 2004).* ACTA Press, 2004, pp. 161–165.

[47] V. Patil and R. K. Shyamasundar, "An Efficient, Secure and Delegable Micro-Payment System," in *the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE-04)*. IEEE Computer Society Press, March 2004, pp. 394–404.

[48] Y. Kawatsura, "Secure Electronic Transaction (SET) Supplement for the v1.0 Internet Open Trading Protocol (IOTP)," June 2003, RFC 3538.

[49] B. Meng and H. Zhang, "An Electronic Commerce System Prototype and Its Implementations," in *Proceedings of the 2005 The Fifth International Conference on Computer and Information Technology (CIT05)*. IEEE Computer Society Press, Sep 2005, pp. 966–970.

[50] M. Kim and Ç. K. Koç, "A simple attack on a recently introduced hash-based strong-password authentication scheme," *International Journal of Network Security*, vol. 1, no. 2, pp. 77–80, Sep 2005.

[51] ——, "Vulnerabilities in the adachi-aoki-komano-ohta micropayment scheme," *International Journal of Network Security*, vol. 4, no. 2, pp. 235–239, Mar 2007.

[52] ——, "Enhanced security for the modified authenticated key agreement scheme," *International Journal of Computer Science and Network Security*, vol. 6, no. 7B, pp. 164–169, Jul 2006.

[53] ——, "Two simple attacks on the id-based password authentication scheme using smart cards and fingerprints," *International Journal Informatica*, In process, 2006.

[54] ——, "Security improvement on a new user remote authentication scheme," *Journal of Computer and System Sciences*, In process, 2005.

[55] ——, "Improving the novikov and kiselev user authentication scheme," *International Journal of Network Security*, To appear in 2007.

[56] ——, "A secure hash-based strong-password authentication protocol using one-time public-key cryptography," *Journal of Information Science and Engineering*, In process, 2006.

[57] T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext Transfer Protocol - HTTP/1.0," May 1996, RFC 1945.

[58] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1," June 1999, RFC 2616.

[59] J. Franks, P. Hallam-Baker, S. L. J. Hostetler, P. Leach, A. Luotonen, and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication," June 1999, RFC 2617.

[60] W. Simpson, "PPP Challenge Handshake Authentication Protocol (CHAP)," Aug. 1996, RFC 1994.

[61] Electronic Frontier Foundation, *Cracking DES - Secrets of Encryption Research, Wiretap Politics and Chip Design.* O'Reilly Press, 1998.

[62] Distributed.net, "Project RC5," in *http://www.distributed.net/rc5/*, Sep 2002.

[63] B. Burr, "NIST Cryptographic Standards Program," in *http://csrc.nist.gov/wireless/*, Dec 2002.

[64] G. Lowe, "Analyzing protocols subject to guessing attacks," in *Workshop on Issues in the Theory of Security (WITS02)*, Jan 2002.

[65] IEEE 1363-2000, "IEEE Standard Specifications For Public Key Cryptography, Institute of Electrical and Electronics Engineers," in *http://grouper.ieee.org/groups/1363/*, 2000.

[66] Telecom Glossary 2000, "T1 523-2001, Alliance for Telecommunications Industry Solutions (ATIS)," in *http://www.atis.org/*, 2000.

[67] T. ElGamal, "A subexponential-time algorithm for computing discrete logarithms over $GF(p^2)$," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 473–481, July 1985.

[68] A. K. Awasthi and S. Lal, "An enhanced remote user authentication scheme using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 2, pp. 583–586, 2004.

[69] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[70] W. H. Yang and S. P. Shieh, "Password authentication schemes with smart cards," *Computers & Security*, vol. 18, no. 8, pp. 727–733, 1999.

[71] K. W. Kim, J. C. Jeon, and K. Y. Yoo, "An improvement on yang et al.s password authentication schemes," *Applied Mathematics and Computation*, vol. 170, no. 1, pp. 207–215, 2005.

[72] K. F. Chen and S. Zhong, "Attacks on the (enhanced) yang-shieh authentication," *Computers & Security*, vol. 22, no. 8, pp. 725–727, 2003.

[73] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[74] A. Shimizu, "A dynamic password authentication method by one-way function," *IEICE Transactions*, vol. E73-DI, no. 7, pp. 630–636, July 1990.

[75] A. Shimizu, T. Horioka, and H. Inagaki, "A password authentication method for contents communication on the internet," *IEICE Transactions on Communications*, vol. E81-B, no. 8, pp. 1666–1673, Aug. 1998.

[76] T. Tsuji and A. Shimizu, "An implementation attack on one-time password authentication protocol OSPA," *IEICE Transactions on Communications*, vol. E86-B, no. 7, pp. 2182–2185, July 2003.

[77] W. C. Ku, H. C. Tsai, and S. M. Chen, "Two simple attacks on Lin-Shen-Hwang's strong-password authentication protocol," *ACM Operating System Review*, vol. 37, no. 4, pp. 26–31, Oct. 2003.

[78] M. S. Hwang, "Cryptanalysis of a remote login authentication scheme," *Computer Communications*, vol. 22, no. 8, pp. 742–744, 1999.

[79] H. Y. Chien, J. K. Jan, and Y. M. Tseng, "An efficient and practical solution to remote authentication: smart card," *Computers & Security*, vol. 21, no. 4, pp. 372–375, 2002.

[80] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings CRYPTO 84*. Springer Verlag, LNCS Nr. 196, 1984, pp. 47–53.

[81] L. Gong, "A security risk of depending on synchronized clocks," *ACM Operating System Review*, vol. 26, no. 1, pp. 49–53, Jan. 1992.

[82] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, July 1985.

[83] H.-S. Kim, S.-W. Lee, and K.-Y. Yoo, "Id-based authentication scheme using smart cards and fingerprints," *ACM Operating System Review*, vol. 37, no. 4, pp. 32–41, Oct. 2003.

[84] S. T. Wu and B. C. Chieu, "A user friendly remote authentication scheme with smart cards," *Computers & Security*, vol. 22, no. 6, pp. 547–550, 2003.

[85] K. F. Hwang and I. E. Liao, "Two attacks on a user friendly remote authentication scheme with smart cards," *ACM Operating System Review*, vol. 39, no. 2, pp. 94–96, Apr. 2005.

[86] M. S. Hwang, J. W. Lo, C. Y. Liu, and S. C. Lin, "Cryptanalysis of a user friendly remote authentication scheme with smart card," *Journal of Applied Sciences*, vol. 5, no. 1, pp. 99–100, 2005.

[87] S. T. Wu and B. C. Chieu, "A note on a user friendly remote authentication scheme with smart cards," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E87-A, no. 8, pp. 2180–2181, Aug. 2004.

[88] K. Tan and H. Zhu, "Remote password authentication scheme based on cross-product," *Computer Communications*, vol. 18, pp. 390–393, 1999.

[89] S. N. Novikov and A. A. Kiselev, "The authentication of the user from the remote autonomous object," in *4th Siberian Russian Workshop and Tutorials EDM 2003: Section II, Erlagol*. NSTU, 2003, pp. 137–138.

[90] C. Y. Yang, C. C. Lee, and S. Y. Hsiao, "Man-in-the-middle attack on the authentication of the user from the remote autonomous object," *International Journal of Network Security*, vol. 1, no. 2, pp. 81–83, 2005.

[91] A. K. Awasthi, "On the authentication of the user from the remote autonomous object," *International Journal of Network Security*, vol. 1, no. 3, pp. 166–167, 2005.

[92] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, pp. 644–654, Nov. 1976.

[93] R. J. Anderson and T. M. A. Lomas, "Fortifying key negotiation schemes with poorly chosen passwords," *Electronics Letters*, vol. 30, no. 13, pp. 1040–1041, 1994.

[94] D. H. Seo and P. Sweeney, "Simple authenticated key agreement algorithm," *Electronics Letters*, vol. 35, no. 13, pp. 1073–1074, 1999.

[95] Y. M. Tseng, "Weakness in simple authenticated key agreement protocol," *Electronics Letters*, vol. 36, no. 1, pp. 48–49, 2000.

[96] W. C. Ku and S. D. Wang, "Cryptanalysis of modified authenticated key agreement protocol," *Electronics Letters*, vol. 36, no. 21, pp. 1770–1771, 2000.

[97] C. L. Hsu, T. S. Wu, T. C. Wu, and C. Mitchell, "Improvement of modified authenticated key agreement protocol," *Applied Mathematics and Computation*, vol. 142, no. 2, pp. 305–308.

[98] N. Y. Lee and M. F. Lee, "Further improvement on the modified authenticated key agreement scheme," *Applied Mathematics and Computation*, vol. 157, no. 3, pp. 729–733, 2004.

[99] B. Cox, D. Tygar, and M. Sirbu, "NetBill security and transaction protocol," in *In Proceedings of the First USENIX Workshop on Electronic commerce.* USENIX, Jul 1995, pp. 77–88.

[100] E. Foo and C. Boyd, "A Payment Scheme Using Vouchers," in *In Proceedings of the Second International Conference on Financial Cryptography.* Springer Verlag, 1998, pp. LNCS Nr. 1465, pp. 103–121.

[101] H.-P. Messmer, *The Indispensable Pentium Book.* Addison-Wesley, 1995.

[102] T. Poutanen, H. Hinton, and M. Stumm, "NetCents: A Lightweight Protocol for Secure Micropayments," in *In Proceedings of the Third USENIX Workshop on Electronic Commerce.* USENIX, Sep 1998, pp. 25–36.

[103] WebMoney Corporation, "WebMoney for Internet shopping," in *http://www.webmoney.com*, 2005.

[104] M. Bellare, J. Garay, C. Jutla, and M. Yung, "VarietyCash: a Multi-Purpose Electronic Payment System," in *In Proceedings of the Third USENIX Workshop on Electronic Commerce.* USENIX, Sep 1998, pp. 9–24.

[105] J. B. Friis, "Digicash implementation," in *http://www.appliedcrypto.com*, 2003.

[106] R. L. Rivest and A. Shamir, "Payword and micromint : Two simple micropayment schemes," in *Fourth Cambridge Workshop on Security Protocols.* Springer Verlag, April, 1996, pp. 69–87.

[107] N. Adachi, S. Aoki, Y. Komano, and K. Ohta, "Solutions to security problems of Rivest and Shamir's PayWord scheme," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E88-A, no. 1, pp. 195–202, Jan. 2005.

[108] N. K. Ratha and A. K. Jain, "A real-time matching system for large fingerprint database," *IEEE Transactions on Pattern Anal. Mach. Intell.*, vol. 18, pp. 799–813, 1996.