A New Protocol for Password Authentication and Key Exchange

Mohammad Albataineh

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

March, 2006

Your file Votre référence
ISBN: 0-494-14251-0
Our file Notre référence
ISBN: 0-494-14251-0

Abstract

A New Protocol for Password Authentication and Key Exchange

Mohammad Albataineh

Computer security is a field of computer science concerned with the control of risks related to computer use. The importance of computer security is continuously increasing with the rapid growth of computer technology, and becomes nowadays the most essential aspect in the computing world. One of the key requirements of computer security is authentication, which is to establish the authenticity of someone or something by verifying the real identity of that entity.

In this thesis, we researched authentication, its techniques and requirements. Then, we shifted our focus on passwords, their history, subjective attacks, passwords systems, password authentication protocols, their history, evolution, goals and characteristics, formal definitions and proofs, and provided a summary of all existing protocols accompanied with comparisons.

We included some statistics gathered from a password survey. Then based on our results, we suggested some enhancements on existing password systems to strengthen their advantages and minimize their weaknesses.

We also introduced our protocol Mutual Authentication and Key Exchange using Usernames and Passwords (MAKE-UP), to implement our key idea on practical setting and help in solving numerous problems in this crucial security field.

Our idea is to keep the user ID secret, and to use it as a compulsory form of authentication with the passwords.

ACKNOWLEDGMENTS

I would like to take this opportunity to express my sincere gratitude to my supervisor Dr. Abdeslam En-nouaary, for his continuous guidance and generous support through each and every step of writing this thesis which has helped me gain experience, increase my understanding, and enhance my knowledge.

I also owe a debt of gratitude for all the teachers who have taught me through my undergraduate studies and to the members of the examination committee who have read and reviewed my thesis.

I dedicate this work to my father, my mother, my sister, my brother, my friends, and the whole world.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

The computer has a universal influence on our way of life. It is considered to be one of the most important technological advancement that has brought a lot of benefits to mankind. As the computer technology advances, especially with networking, intranet, and internet our dependency on the computer systems increases day by day, in conjunction with the relative increase in computer crimes and security breaches.

The world of security is a very complex and extensive knowledge domain to master, because there are many facets to security, which are frequently changing. After spending years in this discipline, practitioners have realized that the more they know, the more they need to know.

Computer security is a field of computer science concerned with the control of risks related to computer use. One of the key requirements of computer security is authentication, which is to establish the authenticity of someone or something by verifying the real identity of that entity. Face to face human communication solves this problem effortlessly by visual recognition. However, when communicating entities exchange messages over a medium where they can not see each other, authentication is no longer an option but a requirement.

1.1 The objectives of the thesis

Authentication is the subject of this thesis. We propose a simple idea that improves the authentication process, and helps in solving numerous problems in this crucial security field.

Albert Einstein said “Most of the fundamental ideas of science are essentially simple”. Our approach is based on logical reasoning, thus if you have two locks on your door there is no logical reason to lock one and give away the second key to everyone. Certainly, two locks are better than one.

Users are required to identify themselves so that they can be authenticated as the individuals they claim to be. In the authentication process, you always use a pair of a username and a password, although the username is never secret. Attackers simply guess at username and password combinations until they either get lucky or give up. This guessing process is simplified if the attacker was able to deduce a legitimate user name. The question that keeps repeating itself is “why would someone need to know my username?”

It is a fact that the more someone knows about you the more you are vulnerable to an attack. This thesis suggests to keep the username secret and to use it as a compulsory form of user authentication. Our idea is general and easy to implement and overcomes most of the difficulties of previously suggested methods for user authentication, and will efficiently strengthen the password authentication scheme and result in a more secure environment.

This thesis will extensively research password authentication and related problems, and provide some useful key points to reinforce the weakest link in the authentication process. However, our main contribution is the introduction of a new protocol Mutual Authentication and Key Exchange using Username and Passwords (MAKE-UP), used for the password-based key exchange that is used for authentication and to provide the communicating parties with a session key that is used to encrypt the rest of the communication through an unsecured medium.

1.2 Organization of the thesis

The rest of the thesis is organized as follows. Chapter 2 provides an overview of computer security and explains what is authentication, the different techniques used to identify the authenticity of the user, and the authentication requirements that should be satisfied by authentication protocols, accompanied with a comparison between authentication techniques based on these requirements. It also introduces passwords and their history, attacks subjected on passwords, some existing solutions for strengthening password authentication systems and their drawbacks, and finally it presents our recommendations for strengthening password authentication systems. Chapter 3 then provides the evolution of password authentication mechanisms by listing the problems and solutions. Chapter 4, introduces password-only authentication and key exchange protocols, their goals and characteristics, highlights the history of such protocols, summarizes their genealogy, presents the formal proofs and definitions of the available protocols, and discusses the password verifier protocols and threshold authentication. Chapter 5 introduces our protocol, Mutual Authentication and Key Exchange using Usernames and Passwords (MAKE-UP) that implements our main idea, then describes the implementation details, and provides a security analysis. Finally, Chapter 6 concludes this paper and highlights our achievements and future works.

Chapter 2: Overview of Computer Security

Computer security is considered to be a subfield of security engineering, and it is defined as the effort to create a secure computing platform. It is concerned with assets of computer systems: the information and services which they provide. This chapter provides an overview of computer security, and presents authentication techniques as well as issues related to passwords. Finally, it introduces some suggested enhancements to result in more secure password authentication systems.

2.1 Computer Security

Computer security is the prevention and protection of computer assets from unauthorized access, use, alteration, degradation, destruction, and other threats. It is divided into two main subparts:

- Physical computer security: Entails physical, concrete, or touchable protection devices, such as locks, cables, fences, or vaults.
- Logical computer security: Entails non-physical protection, such as the security services provided by authentication and encryption.

In this thesis, we focus on logical computer security, and will extensively discuss authentication and encryption techniques.

Computer security is classified into three aspects. First, security attacks: any action that compromises the security of information. Second, security mechanism: a mechanism that is designed to detect, prevent, or recover from a security attack. Third, security services:

the services intended to counter security attacks and they make use of one or more security mechanisms. The security services are as follows:

- Authentication: both the sender and the receiver should be able to confirm the identity of the other party involved in the communication. To confirm the other party is indeed who or what he or she claims to be.
- Confidentiality: only the sender's intended receiver should be able to understand the content of the transmitted message.
- Message integrity: even if the sender and receiver are able to authenticate each other, they also want to ensure that the content of their communication is not altered, either maliciously or by accident, while in transmission.
- Non-repudiation: verifies that the sender and the recipient were, in fact, the parties who claimed to send or receive the message, respectively. In other words, non-repudiation of origin proves that data has been sent, and non-repudiation of delivery proves it has been received.
- Availability and Access control: services must be accessible and available to properly authorized users.

User authentication is one of the most important security services in the security world; as it is necessary to verify the identities of the communicating parties before they start a new connection.

In computer security, authentication is the process by which a computer, computer program, or another user attempts to confirm that the computer, computer program, or user from whom the second party has received some communication is, or is not, the claimed first party.

2.2 Authentication and authorization

Authentication and authorization are often thought to be identical, which is with no doubt a faulty assumption. Authorization is protecting computer resources by only allowing resource consumption by those who have been granted authority to use them. However, authentication is verifying the real identities of the communicating entities.

To exemplify the difference between authentication and authorization, let us consider location security. In location security, access control refers first to restricting entrance to a location to those authorized (authorization), secondly to confirm the identity of the people who claim to be authorized (authentication). A computer system is supposed to be used only by those authorized, and must attempt to detect and exclude the unauthorized. Access to it is therefore usually controlled by insisting on an authentication procedure before access is granted.

2.3 Authentication techniques

There exist three different techniques to identify the authenticity of a user:

1) Relying on something users know (such as passwords)

The authenticity of the user is established by asking the user to provide some piece of information that only the legitimate user knows. The classic username and password combination is by far the most common implementation of this authentication strategy. Variations of this method of authentication, is asking the user to answer a secret question (such as "What is your mother maiden name?").

2) Something they have (such as physical keys)

Instead of relying on user's memory, the system could require that users actually have in their possession some artifact or token that is not easily reproducible.

3) Some physical attributes of themselves (such as fingerprints)

This methodology is based on a biometric device that measures some unique property of the user that cannot be easily forged or altered, thus providing an extremely accurate method of identifying an individual user. Biometric authentication method includes face topology and geometry fingerprints, eye patterns, hand topology and geometry, and voice. Ideally, a combination of two or more of these methods should be used.

2.4 Authentication requirements

The authentication process demands some requirements for its success, which are based on the needs of the user and the system. These requirements determine the choice of one of the three authentication mechanisms:

- Accuracy: the accuracy of an authentication mechanism can be measured in terms of the percentage of legitimate users who attempt to authenticate themselves but are rejected by the system, and by the percentage of unauthorized users, who are able to deceive the system.
- Availability: where and when the services must be accessible and available to properly authorized users.

- Cost: service providers' view cost as a key requirement and they strive for the least cost possible. Of course, the three techniques have different costs in terms of implementing, operating, and maintaining the authentication process. For example, we all know that biometric measurements introduce variable costs in terms of the specific choice used.
- Convenience, the system should be as friendly as possible and the authentication process should be as invisible as possible. This is one of the major aspects in authentication, as it plays a major role in the user's encouragement to use the system. The balance between security and convenience should be considered to the furthermost point without falling into the edge of vulnerability.
- Practicality: The practicality of the authentication mechanisms is crucial. For example, it is not always possible for the user to carry any form of authentication with him such as a key, card, or any other form of physical matter. At the same time, it is not easy to reproduce such an object if it is lost or damaged. Nonetheless, technological advancements have made it possible to fake physical attributes, such as: geometry fingerprints, eye patterns, and some other forms of biometric measurements. One may raise the question: what happens if a biometric ID is compromised?
- Robustness and reliability: The system should perform as designed while being resilient to failures and attacks.

In general, there is not a universal best way to set an appropriate balance between all these factors. There is always a risk in giving away one property to achieve another. The decision should be made with full comprehension of both upsides and downsides. In terms of extensive research and further intensive study in this domain, it has been shown that among the three different techniques introduced so far, "relying on something a user knows" achieves the most proper balance between cost, simplicity, convenience, practicality, and security. Therefore passwords are the most adequate method of authentication, and will most probably continue to be extensively used in the foreseeable future. In the following section, we introduce passwords and discuss some related issues.

2.5 Passwords

The use of passwords goes back to ancient times. Sentries guarding a location would ask for a password, and will only allow a person in if he or she knew the password. A *password* is a form of secret authentication data that is used to control access to a resource. The password is kept secret from those who are not allowed access, and those wishing to gain access are tested on whether or not they know the password and are granted or denied access accordingly. A password is a sequence of characters that one must input to gain access to a file, application, or a computer system. "Open Sesame" is notably the most famous password in literature as it gave Ali Baba access to vast treasure. In the realm of technology, computer passwords also give access to valuable treasures; they are better thought of as the virtual keys to priceless information assets. Despite the name, there is no need for passwords to be actual words; indeed passwords which are not actual words are harder to guess (a desirable property). It is noteworthy that a password is often used to describe what would be more accurately called a pass phrase.

Passwords are the frontline of computer security; they are often the first and only line of defense against intruders. Unfortunately, they are considered to be the weakest link.

2.5.1 Strong and week passwords

One of the important weaknesses of password authentication systems arises from the limitations of human memory. Users choose weak passwords, simply because they find it easier to remember such passwords. Although choosing a weak password is equivalent to locking your door and leaving the key under the doormat.

A weak password is short or is default. In other words, they are easy to guess, possible to predict, or most probably will be found by searching a subset of all possible passwords such as words in the dictionary, proper names, words based on the user name or common variations on these themes. On the other hand, a strong password is sufficiently long, random, and complex, so that guessing it will require a long time.

The length of time deemed to be too long will vary with the attacker, with the attacker's resources, with the ease with which a password can be tried, and with the importance of the password to the attacker. So a student's password might not be worth more than a few seconds of computer time, whilst a password controlling access to a large bank's electronic money transfer system might be worth many weeks of computer time.

2.5.2 Attacks on passwords

Passwords are the weakest link in the authentication process, because they can be forgotten, stolen, sniffed, cracked, and subjected to multiple attacks such as:

- Eavesdropping: The attacker listens on the line and tries to learn some useful information from ongoing communication.
- Replay: The attacker records messages sent in past communications and resend them at a later time.
- Man-in-the-middle: The attacker intercepts the messages between the parties and replaces them with his or her own messages. The attacker plays the role of the user in the messages sent to the server and at the same time plays the role of the server in the messages sent to the user.
- Guessing attacks: The attacker tries different passwords until he or she gets lucky or gives up. Such attacks can be classified into three categories:
    - Dictionary Attack, the attacker uses a file that contains commonly used passwords found in a dictionary. There are primarily two ways in which the attacker can use the dictionary:
        - Off-line attack: The adversary records past communication to obtain a ciphertext of the password. Even though the attacker can not reverse the cryptographic hash, the attacker goes over the dictionary, picks a password, performs the same hash, and checks if the result is the same as the stolen quantity. If that is the case, the attacker concludes that it is the password of the user. Otherwise, he or she picks another password and continues the same process.

        - On-line attack: The attacker repeatedly picks a password from the dictionary and tries to use it in order to impersonate the user. If the impersonation fails, the attacker eliminates this password from the dictionary and tries again, using a different password.
    - Brute Force Attack, the attacker tries every possible combination of letters, numbers and special characters. In other words, he or she will search the whole key space.
    - Hybrid Attack, the attacker concatenates extra characters to dictionary words and tries different combinations.

2.6 Secure password systems

Some of the presented attacks can be solved by technological means such as encryption, hashing, and password-based authentication protocols; although the human factor plays the most important role in the security of the authentication system. Each password based authentication system is, without exception, vulnerable to certain attacks if a weak password is chosen. Conversely, using a strong password will not do any good if it is used in weak or seriously compromised password authentication systems.

We have gathered some statistics from a questionnaire that was distributed over 500 participants with different backgrounds. We will use our survey as a reference for some statistics that will highlight important key points.

We present five solutions that should be used to solve the problems encountered in using passwords as the only form of authentication; these solutions actually work out as avoidance means to prevent problems in the first place.

1) User education: Educate users about the significance of passwords and the importance of choosing strong passwords. In our survey, around 40% of respondents have agreed to exchange their password for a bar of chocolate. This makes it crystal clear, that password users are inconsiderate to the importance of the password they possess, and this highlights the fact that extra effort in educating users is a must need. The most important reason for our insecurity and vulnerability is shortsightedness; which is defined as the ignorance of security issues by dismissing many security vulnerabilities, and viewing them as un-exploitable or considering their exploitation to be very unlikely. We know in the back of our minds that those security issues exist, but we do not address them unless we are attacked or made aware of a major threat. The art of war teaches us to rely not on the likelihood of the enemy not coming, but on our own readiness to receive him; not on the chance of his not attacking, but rather on the fact that we have made our position unassailable.

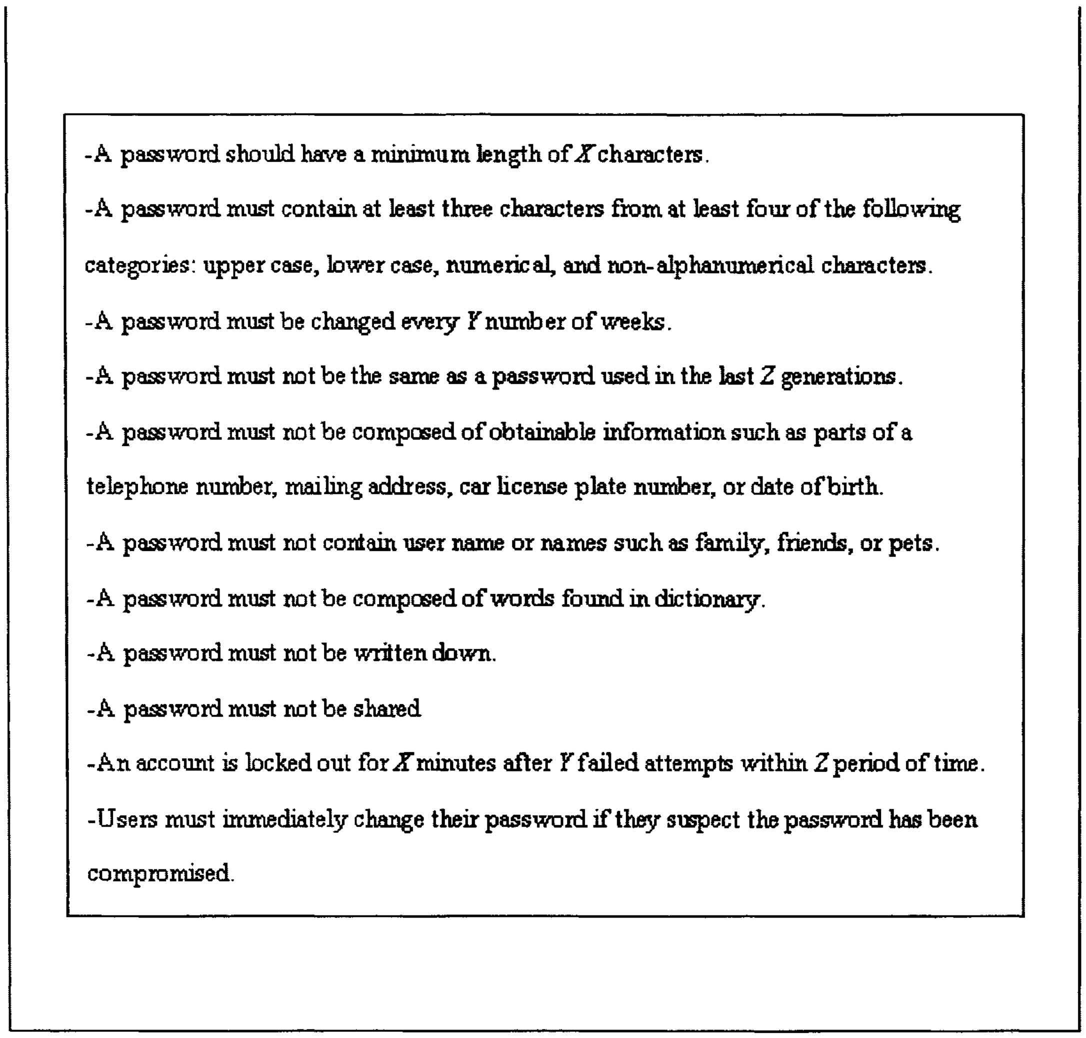
2) Computer generated passwords: Use random password generators to provide strong passwords for users. Random passwords generated on behalf of the system are often difficult to memorize, especially if they are changed frequently. Thus, users may write them down and so hindering the security of the authentication system.

3) Reactive password checking: The system periodically runs its own password cracker to find guessable passwords. The system eliminates any password that is guessed and notifies the user. Reactive password checking suffers from two drawbacks: first, the dictionary used may not be comprehensive enough to screen out weak passwords and second, the system may require more time than the attacker needs to figure out a password.

4) Proactive password checking: Users can not judge the strength of their passwords. For example novice users may mistakenly believe that reversing a word makes a strong password. The system is responsible for advising the strength of the password being chosen. Proactive password checking has been a common tool to enforce password policies and prevent users from choosing easily guessable passwords in the first place. In this scheme, users are allowed to select their own passwords. However, at the time of selection the system checks to see if the password is allowable and, if not rejects it. Such checkers are based on the assumption that with sufficient guidance from the system, users can select memorable passwords from a fairly large password space, which are not likely to be guessed in a dictionary attack. Nowadays, proactive password checking algorithms are based on the philosophy of the dictionary based checking, and they often fail to prevent some weak choices of passwords.

5) Password policy: Adopting strong password policies is one of the most effective ways to ensure the security of the system. The definition of a password policy requires careful considerations and cautious judgments on the tradeoffs between the

feasible boost in system security, and the cost of other significant human factors. An example of password policy may state the guidelines shown in figure 1.



-A password should have a minimum length of *X* characters.

-A password must contain at least three characters from at least four of the following categories: upper case, lower case, numerical, and non-alphanumerical characters.

-A password must be changed every *Y* number of weeks.

-A password must not be the same as a password used in the last *Z* generations.

-A password must not be composed of obtainable information such as parts of a telephone number, mailing address, car license plate number, or date of birth.

-A password must not contain user name or names such as family, friends, or pets.

-A password must not be composed of words found in dictionary.

-A password must not be written down.

-A password must not be shared

-An account is locked out for *X* minutes after *Y* failed attempts within *Z* period of time.

-Users must immediately change their password if they suspect the password has been compromised.

Figure 1: A sample of password policy guidelines

Password policy guidelines could be dangerous in the sense that difficult to follow up policies ends up in reducing the system's security. For example, forcing users to change their passwords frequently may prompt users to choose predictable password sequences such as password1, password2, password3, etc. Another danger of password policies is that in some cases guidelines indicate a specific way of choosing passwords, which many of the users will follow exactly thus granting the attacker a head start.

2.7 Strengthening password systems

In this section, we will further investigate the proposed solutions and the results of our password survey to recommend enhancements and modifications to be augmented in return of establishing a more secure password authentication system[1].

Prior to explaining and discussing our enhancements and modifications to existing solutions, we introduce below the factors that influence password policy guidelines.

2.7.1 Factors that affect a password system

Size of passwords: The shorter the password the more probable it is to be observed, guessed or cracked. Thus, longer passwords are certainly better.

Complexity: Users should be told to choose passwords that contain numbers and special characters as well as capital and small letters. If such a lead is not given, then most of them will choose passwords from a very small subset of the total password space.

Password aging: Forcing users to change passwords periodically ensures that a valid password in the wrong hands will eventually become unusable. However, forcing them to change it too frequently may make users more likely to forget which password is current, and there is a consequent temptation for users to either write their passwords down or to reuse an earlier password, and in some cases give less thought or creativity in choosing them, which may negate any added security benefit. Thus, careful consideration of human factors has to be given to such guidelines. Password aging is usually handled by saving old password strings and comparing them all against future choices. As a matter of fact, in our survey we found out that 28% of users never change their passwords.

Password sharing: Users might share their passwords without even realizing that by revealing the password; they are increasing the risk of a potential break-in. One of the most critical ways of obtaining sensitive data is simply to ask for it, both in a direct or an indirect way; this is what social engineering is all about. In our survey, we found out that 32% of users share their passwords with others.

Number of users per password: Allotting separate passwords to each user of a system, is certainly preferable to having a single password shared by legitimate users of the system. This is partly because people are more willing to tell another person, who may not be authorized, a shared password than one, which is exclusively for their own use. Per user passwords are also essential if users are to be held accountable for their activities.

Procedures for changing passwords: Usually, a system must provide a way to change a password, either because a user believes the current password has been or might have been compromised, or as a precautionary measure. If a new password is passed to the system in an unencrypted form, security can be sacrificed before the new password can even be installed in the passwords' database. Automated systems to change passwords are currently used, where the user's identity is verified by asking questions and comparing the answers to ones previously stored. Typical questions include "What is your mother maiden's name?", or "What is your favorite movie?" In many cases the answers to these questions can be guessed, determined by research, or obtained through social engineering. Nevertheless, users should not be able to choose the same password that was used in the past specific period of time, eliminating the probability that they might have been exposed.

Rate at which an attacker can try out guessed passwords: The rate at which an attacker can submit guessed passwords is a key factor in determining security of a system. This should be regarded from two points of view, namely long term and short term. The former consists of counting the number of impersonation attempts and notifying or possibly obliging the user to change his password when the number has reached a specified threshold. The latter, consists of locking users' accounts after a few unsuccessful attempts (for example, the account is locked for half an hour after three unsuccessful attempts). This scheme delays the attacker and prevents him from checking too many passwords in a reasonable time. However, this scheme suffers from DOS (Denial of Service) attacks; where the attacker tries to login into an account with invalid password with the intention of entering the account locking mode and thus preventing the legitimate user from accessing his account for the locking period specified in the protocol. For example, eBay users who compete in auctions use these methods to block accounts of other users who compete in the same auctions.

2.7.2 Using password entropy to strengthen password systems

To show the importance of using password entropy to measure passwords' strength, let us consider the following four sets from which passwords are constructed. Indeed, we have 26 capital letters, 26 small letters, 10 numbers, and 31 special characters.

Set	**Characters**
Capital letters	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Small letters	a b c d e f g h i j k l m n o p q r s t u v w x y z
Numbers	0 1 2 3 4 5 6 7 8 9
Special Characters	! @ # $ % ^ & * ( ) - _ + = [{ }] ; : ' \ " | , < . > / ? ~

Table 1: Sets of characters

Figure 2 shows the password space for different ways of choosing a password of length 8.

- Use only numbers: $10^8 \approx 10^8$
- Use only capital **or** small letters: $26^8 \approx 2.09 * 10^{11}$
- Use only special characters: $31^8 \approx 8.53 * 10^{11}$
- Use capital **or** small letters **and** numbers: $36^8 \approx 2.82 * 10^{12}$
- Use numbers **and** special characters: $41^8 \approx 8 * 10^{12}$
- Use capital **and** small letters: $52^8 \approx 5.35 * 10^{13}$
- Use capital **or** small letters **and** special characters: $57^8 \approx 1.11 * 10^{14}$
- Use capital **and** small letters **and** numbers: $62^8 \approx 2.18 * 10^{14}$
- Use capital **or** small letters **and** numbers **and** special characters: $67^8 \approx 4.06 * 10^{14}$
- Use capital **and** small letters **and** special characters: $83^8 \approx 2.25 * 10^{15}$
- Use capital **and** small letters **and** numbers **and** special characters: $93^8 \approx 5.6 * 10^{15}$

Figure 2: Calculations based on a password constructed from different sets

From the above calculations it is apparent that choosing the characters of a password from all of the four sets will increase the size of the password space thus making the password strong and hard to crack. It is also obvious, that increasing the length of a password will also make it even stronger and harder to crack. For example, let your password be nine characters and chosen from the four sets mentioned above. The password space is $93^9 = 5.2 * 10^{17}$.

In our survey, we distinguished between users who have been exposed to password policies and the users who have not. When, the questionnaire was passed to the people who have been exposed to a password policy and thus knew the characteristics of a strong password, and to the people who have never been exposed to a password policy and thus did not know the characteristics of a strong password, we have established the statistics shown in table 2.

	never exposed to a password policy	exposed to a password policy
Use password of length 8	50%	50%
Use numbers	80%	84%
Use a mixture of capital and small letters	10%	70%
Use special characters	5%	58%

Table 2: Statistics showing the impact of password policy on password quality

The statistics from table 2 are clearly shown in Figure 3.

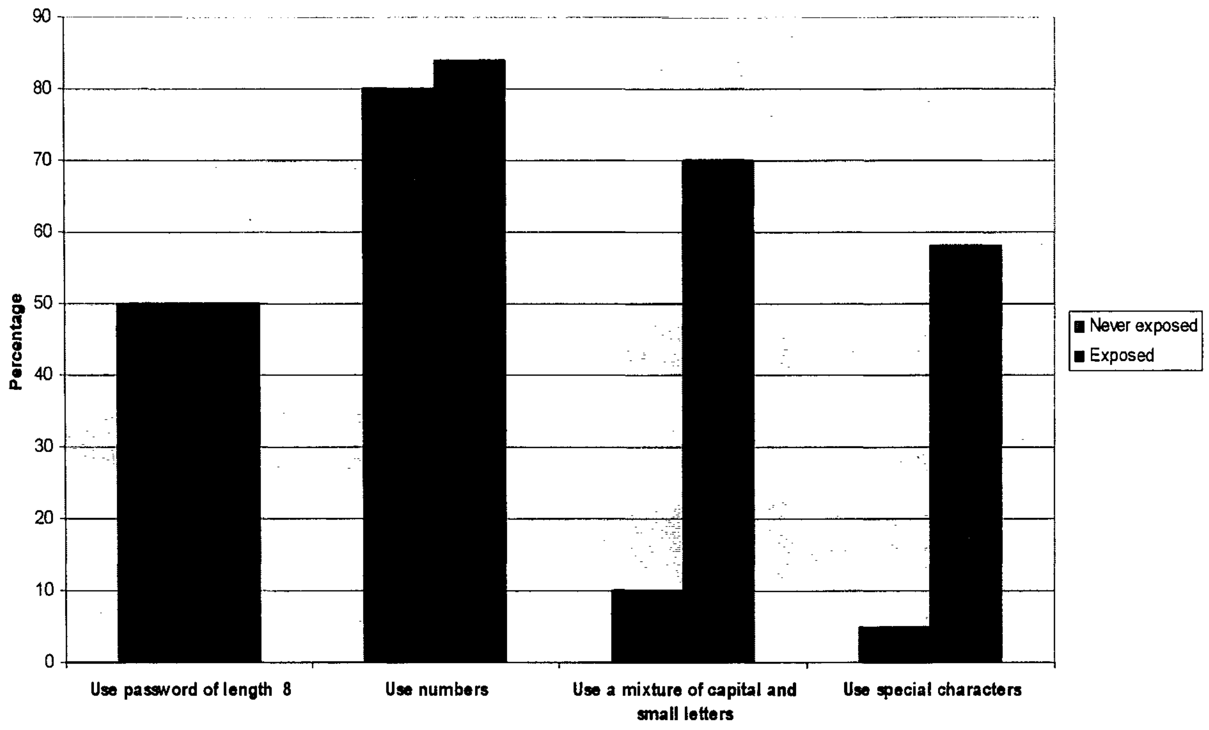


Figure 3: Impact of password policy on password quality

From the statistics of our survey, we conclude the following. Firstly, the use of password policies and user education has resulted in a noticeable improvement in strengthening password choices. Secondly, the users who have been educated on how to choose strong passwords, and who have been exposed to password policies, still choose passwords from smaller sets. Therefore, we ascertain that users' education and password policies are a must because they help users in choosing strong passwords.

2.7.3 Mnemonics based passwords

In our survey we have found out that 62% of respondents have written their password down at least once; due to the fact that they find it hard to remember long and complex passwords. The solution to this problem is to use mnemonics. Mnemonics is a memory aid that does not only rely on repetition to remember facts, but also on associations between easy-to-remember constructs and lists of data, based on the principle that the human mind much more easily remembers data attached to spatial, personal or otherwise meaningful information than that occurring in meaningless sequences. For example, encourage users to choose passwords by abbreviating first letters of a phrase with appropriate substitutions for different letters; "I teach 8 classes at Concordia University" becomes "It<8>c@CU". In 2000, Cambridge university have performed a study on mnemonic based passwords and have made some tentative remarks; users should be instructed to choose mnemonic based passwords as these are just as memorable as naively selected passwords while being just as hard to guess as randomly chosen ones.

To prove the truthfulness of the above observation, let us look at the problem from a different dimension. Assume a password that is 8 characters long and consists of 3 small letters, 3 capital letters, and 2 numbers. An example of such password is "ABCdef12". This password has low entropy and is easier to guess than "Ad1Be2Cf"; which is nothing but the same set of characters arranged in a different order. We stress that the password strength does not depend solely on the length of the password but also on the number of characters chosen from each set and their order (complexity). This means that the above calculations do not represent the actual strength of the password and that passwords strength should be measured based on entropy.

Entropy is expressed in terms of the number of bits of complexity that the password contains. Let us assume, a password of 8 characters in length with 8 bits of entropy per byte, we will then have 8 * 8 = 64 bits of entropy. In other words, $2^{63} \approx 9.2 * 10^{18}$ passwords have to be tried before guessing the password in hand. Note that it is difficult to achieve 8 bits of entropy per character, due to the fact that the keyboard makes it hard to enter some characters and users might forget completely random passwords without any associations. The approximations to the number of bits of entropy per character are as follows: weakly chosen passwords have 2.5 bits of entropy per character, strongly chosen ones have 4.3 bits, and randomly chosen passwords have 6.5 bits; while mnemonics based passwords give about 5.8 bits of entropy per character. These approximations with the consideration of human factors support the research results obtained by Cambridge University researchers. Measuring password based on entropy makes it obvious that mnemonics provide high entropy that is very close to the entropy of randomly chosen passwords, while being easy to remember; thus providing the best of both. Therefore, we stress out that password policies should encourage users to choose mnemonics based passwords.

2.7.4 Enhanced proactive password checkers

Forcing users to choose passwords from at least three of the four sets may not be enough to ensure that they have picked a real strong password. Thus, we complement this by using reactive and proactive password checkers. Proactive password checkers seem to be one of the best solutions that lead to a strong password system, although they suffer from a weakness due to the philosophy of dictionary based checking, and they often fail to prevent some weak passwords with low entropy. This thesis highlights the importance of using entropy based checking to complement dictionary based checking. This is done by first measuring the entropy of the chosen password and if the entropy is high enough then the system will compare user choices against a list of unacceptable passwords and ensure that the chosen password is not in the disapproved list.

Note that the entropy based checking is performed before the dictionary based checking primarily for two main reasons. First, the time it takes to measure the password entropy is notably less than the time taken by the proactive password checker to check if the password is in the disapproved list especially if the database is massive. Second, the probability of having a high entropy password in the disapproved list is negligible when compared to the probability of having a low entropy password that passes the dictionary based proactive password checker.

Feedback to the user is given through an indicator light that shows green if the password is strong enough, orange if the password needs enhancement and red if the password is weak.

2.7.5 Enhancement to the account locking strategy to minimize DOS attack

We suggest here a solution to the DOS attack introduced so far. After the x failed attempts, instead of entering the account locking mode the system challenges the user for something very simple for a human user to answer but is almost infeasible for automated systems, such as "Type the letters as seen in the box". This will certainly distinguish between a human being and a machine trying to do such an attempt. This suggestion has the advantage of delaying the attacker and in some cases preventing a machine from continuing the guessing process. If another x failed attempts have been attempted while in the first phase of the simple challenge, the system may not allow further attempts if the user does not answer the secret question.

In some other applications where the DOS attack is not critical. The account locking could be used but cautiously, thus minimizing its drawbacks. We suggest increasing the locking period incrementally. For example, after x failed attempts in the simple challenge phase the account is locked for 5 minutes, then after the second x failed attempts the account is locked for 10 minutes, then 20 minutes, and so on.

2.8 Summary

In this chapter, we overviewed computer security, and discussed authentication and its requirements and techniques. Then, we deeply researched password authentication and all its related topics. We ended this chapter by suggesting enhancements and modification to strengthen password authentication systems.

In the next chapter, we will present password authentication protocols and their evolution.

Chapter 3: Password Authentication Protocols

So far we have discussed authentication and extensively studied passwords and all their related topics. In this chapter, we start by highlighting the evolution of password authentication protocols and the root problems that led to their existence.

3.1 Evolution of password authentication

In the history of password security, the practitioners have made incredible contributions to reach a great state of security, although up to this date problems still exist but in a different dimension. To exemplify the evolution of password based authentication, let us review some of the past protocols and the encountered problems that have guided to better protocols.

3.1.1 Passwords in the clear

The simplest and weakest password authentication protocol transmits the password in the clear from the user to the server. For example, in the past, Internet was considered to be relatively secure network and thus UNIX utilities such as, telnet and ftp, simply sent the password in the clear. An example of such protocols is Password Authentication Protocols (PAP); it is a simple authentication protocol that requires users to enter a password before accessing a secure system. The user's name and password are sent in an unencrypted format (in the clear) to a server, where they are compared with a database of user account names and passwords. This technique is vulnerable and provides no protection whatsoever. Thus, any eavesdrop attack will lead to revealing the password.

3.1.2 Challenge and response

To solve the replay attacks, practitioners proposed the use of challenge-response mechanisms, where the passwords are never sent in the clear and the freshness of the authentication is insured. To exemplify such a method, the server will send a challenge to the user, such as a random string, the user has then to reply with a response that is based on the challenge and the password. As a trivial example to indicate the difficulties that arise, consider the simple authentication protocol in which one party sends a random nonce r as a challenge and the other party replies with $y = H\ (pw;\ r)$, where pw represents the shared password and H is a cryptographic hash function. An adversary who obtains r *and* y can recover the password by mounting an off-line dictionary attack, trying all values of pw' until one satisfies the formula of $y = H\ (pw';\ r)$.

3.1.3 Plaintext equivalence

The passwords are stored as a cleartext in a database on the server. Authentication is established by comparing the typed password with the stored one. Such systems are compromised once the password database is revealed.

3.1.4 Hashed passwords

To deal with the plaintext issue, instead of storing the password as a cleartext on the server, the password is hashed using a one-way hash function that is computationally infeasible to invert. Authentication is established by hashing the typed password and comparing it to the stored value.

3.2 Inconvenient authentications

Due to the absence of strong and simple password-based mechanism, some practitioners have strived for the security at the expense of convenience, and have therefore designed systems that are not entirely password-based, and require some extra overhead on part of the user, administrator, or occasionally both.

3.2.1 One time passwords

As the name implies the password is used only once and then discarded. The advantage of such a system is that if the password is intercepted the security of the protocol is not hindered because the password is inactive. One time password schemes such as S/KEY [2] are vulnerable to password guessing attacks, and result in a great inconvenience to the user, because he or she will have to manage a list of one time passwords or use a program that generates one time passwords on the fly. Moreover, the inconvenience continues when the user's list of one time passwords needs regeneration.

3.2.2 Third parties

A trusted third party is involved in the authentication process, such as Key Distribution Center (KDC), or Certification Authorities (CA). For example, when introducing a third party such as Public Key Infrastructure (PKI), it is assumed that each party has generated a secret-key and deposited a corresponding public-key with some trusted server(s). The latter server(s) may be accessed at any time by any user. As an example of such protocols, is Kerebros, which is used in several applications and in particular it is the actual authentication protocol under Microsoft Windows. As a matter of fact such protocols are easily constructed and are considered to be strong, although they suffer from loss of simplicity and they do not solve the problem but instead stretch it to a different dimension.

Such protocols increase the overhead on the server and client, as they need more than just a password such as public keys and certificates. The attacks on the password will instead be subjected on public keys, and the authenticity of the certification authority will need to be verified. Other problems exist, such as how the authentication process between the client, server, and the certification authority is done and how passwords, public keys, and other information are communicated securely.

3.3 Summary

In this chapter, we looked over the evolution of the password authentication protocols, which made it crystal clear that we seek strong password authentication protocols, so that an attacker eavesdropping on an ongoing communication will not have enough information about the password to impersonate the user. We also look for a password-only based methodology that makes the password an independent factor, and simplifies the protocol without hindering the security of the system. Our needs have been fulfilled with the introduction of Password-only Authenticated Key Exchange (PAKE). PAKE protocols have received a growing attention in recent years, because of their practicality and simplicity in the sense that the user should hold a password only and no certificates, certification authorities or public key infrastructure involved. In the next chapter, we will intensively discuss PAKE protocols and their related issues.

Chapter 4: Password-only Authenticated Key Exchange

In this chapter, we will further study authentication form the view point of the password authentication systems and will intensively research the protocols used for password authentication, their history, evolutions, goals and characteristics, drawbacks, a summary of all existing password-based authentication protocols, their formal definition and proofs[3].

The communicating parties would like to confirm each other identities by mutual authentication, and to share a fresh secret session key by Authenticated Key Exchange (AKE), to be used for protecting their subsequent communication over insecure networks. Thus, PAKE protocols consist of two stages: A key agreement phase, which is a mutual act that ends with both parties sharing a common session key. Then a second phase, key confirmation phase, where both parties verify that they share the same session key. PAKE protocols have been practically used in a wide range of applications because they do not require anything more than a memorized password, making them much easier to use, simpler to implement, and less expensive to deploy than any other form of authentication mechanism.

4.1 PAKE goals and characteristics

The following are a list of PAKE goals and characteristics obtained from the requirements listed in RFC 1704.

1. Provide mutual authentication.
2. Prevent off-line dictionary attack.

3. Provide security against an active adversary who can start multiple sessions between the client and server, and create, read, add, modify, delay, or delete messages.
4. Survive on-line dictionary attack.
5. The active adversary will not be able to obtain information about the password more effectively than by direct on-line guessing.
6. Provide security against Denning-Sacco attack, learning already distributed session keys will not help the adversary attacker to discover passwords, verifiers or new session keys.
7. Provide forward secrecy, if the adversary learns the client password or the server password verifier, still the adversary will not be able to ascertain anything about previously distributed session keys.
8. A passive adversary (eavesdropper) will not be able to ascertain anything about session keys even after learning the password.
9. If the adversary learns the password for a specific server, the adversary will still have to perform a dictionary attack in order to impersonate the client or the server.
10. The user must have only a password and no other information.

4.2 PAKE drawback: On-line dictionary attacks

Understanding the goals and characteristics of PAKE protocols, will reveal that they prevent off-line dictionary attacks but take no serious action against on-line dictionary attacks. In fact, they try to limit the adversary to on-line attacks only. This problem arises because PAKE protocols assume that on-line dictionary attacks are easily detected due to the fact that they need feedback from the login server. Multiple techniques have been proposed to tackle the on-line attacks such as account locking or delayed response. However, account locking leads to Denial of Service (DOS) attacks and may cause an increase in administrators' load and hence a noticeable increase in the cost of such systems, especially in the case of manually unlocking accounts. Nevertheless, DOS attacks hinder the reliability of the service availability.

Unfortunately, attackers can rent thousands of "Zombies" networks of infected home computers to launch multiple simultaneously parallel attacks using different usernames to verify a huge number of passwords during such an on-line attack. There are about two million zombie workstations worldwide, and those zombie machines represent nothing but a huge bandwidth to be used by attackers.

This drawback is solved by our protocol that will be presented in chapter 5.

4.3 Summary of password-only authentication protocols

The history of PAKE protocols go back to 1976 [4], when the Diffie-Hellman key agreement was introduced during collaboration between Whitfield Diffie and Martin Hellman and was the first practical method for establishing a shared secret over an insecure communications channel. The Diffie-Hellman theory is based on the difficulty of breaking exponential key exchange that have a direct impact on conceptual difficulty and the increased time to verify a guessed attack.

Figure 4 provides a genealogy of all the available protocols that highlights their relation in time and origin. This genealogy is typically based on our opinion.

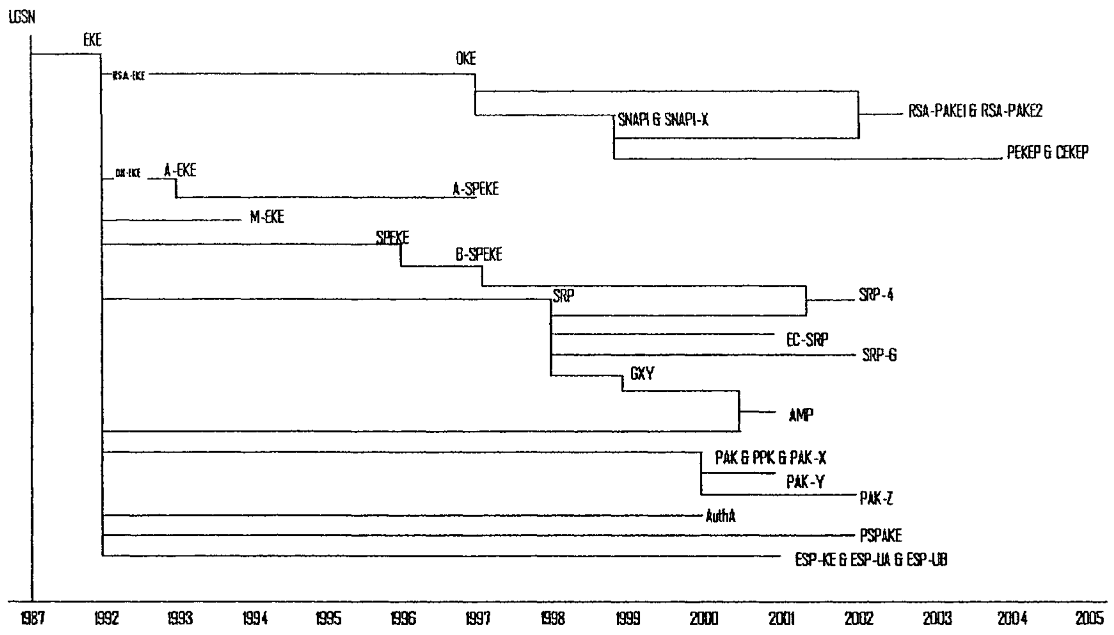


Figure 4: Genealogy of Password Protocols.

In 1989, the first paper that accepted passwords as poorly chosen secrets and dealt with it as an actual fact of life and instead looked for ways to improve the authentication protocol was introduced by LGSN [5]. Although, their scheme depended on a trusted third party; it was the initial consideration of password-based authentication. LGSN have formed the basis for much future work in this area.

In 1992, Bellovin and Merritt presented a new protocol known as Encrypted Key Exchange (EKE) [6]. EKE protocol was secure due to the fact that it leaks insufficient information about the password for the attacker to verify a guessed password. Thus, EKE family have been extended by several other protocols such as: A-EKE, M-EKE, SPEKE, A-SPEKE, B-SPEKE, OKE, GXY, SNAPI, SNAPI-X, AuthA, KOY, ESP-KE, ESP-LIA, ESB-LIB, PAPAKE, RSA-PAK1, RSA-PAKE2, PEKEP, CEKEP, PAK, PPK, PAK-X, PAK-Y, PAK-Z, SRP, SRP-4, SRP-6, TP-AMP, and AMP and its variants.

EKE was applied to asymmetric cryptosystems and public key distribution systems. It was implemented using RSA [7], ELGamal [8], and it achieved enormous success when implemented in the exponential key exchange that was later called DH-EKE. In DH-EKE, the two parties execute an encrypted version of the Diffie-Hellman key exchange protocol, in which each flow is encrypted using the password shared between these two users as the symmetric key. Due to the success of this implementation, EKE based on the framework of Diffie-Hellman [4] was proceeded by several protocols.

4.3.1 EKE review

EKE represented the initial consideration of an "only" password-based authentication protocol, and thus was considered a great landmark for PAKE protocols.

Let us review the EKE protocol; assume that two parties, A and B, wish to establish a secret, authenticated session key; while sharing only a password pw. The authentication goes as follows:

1) A picks a random number r_A and calculates $pw(\alpha^{r_A}(\mathrm{mod}\, n))$, than sends $A, pw(\alpha^{r_A}(\mathrm{mod}\, n))$ to B.

2) B picks a random number r_B and calculates $pw(\alpha^{r_B}(\bmod n))$. B uses pw to decrypt $pw(\alpha^{r_A}(\bmod n))$. B then calculates $(\alpha^{r_A r_B})(\bmod n)$; where the session key K is derived from this value.

   Finally, B transmits $pw(\alpha^{rB}(\bmod n)), K[Challenge_B]$ to A.

3) A uses pw to decrypt $pw(\alpha^{r_B}(\bmod n))$. Then calculates K to decrypt $K[Challenge_B]$, and sends $K[Challenge_A, Challenge_B]$ to B.

4) B decrypts $K[Challenge_A, Challenge_B]$, and verifies that $Challenge_B$ is correct. Then sends $K[Challenge_A]$ to A.

5) A decrypts to obtain $Challenge_A$.

The original EKE protocol suffers from a major drawback due to the fact that the protocol requires that both parties have cleartext versions of the password. The server should never store the password in either clear form or reversibly-encrypted form; because the server itself could be compromised and thus will reveal all the passwords stored in the server. Nonetheless, the original EKE protocol also required a key distribution center.

Although the original EKE was flawed, the surviving and enhanced forms of it effectively amplify a shared password into a shared key. Up to this moment, the family of EKE protocols signify efficient, practical, and secure PAKE protocols.

4.3.2 The extension of the DH-EKE subfamily

In 1993, DH-EKE was enhanced to form Augmented Encrypted Key Exchange (A-EKE) [9] where the hosts store the password in an encrypted form. A-EKE which was also introduced by Bellovin and Merritt to be the first verifier-based protocol to resist a password file compromise and to be the first to make use of a salt[(1)] that was discovered 10 years earlier by Morris and Thompson [10]. Nevertheless, this augmentation was traded off with the forward secrecy property. In 1994, DH-EKE was modified to result in a simplified protocol, which was called a minimal EKE protocol (M-EKE) [11].

In1996, Jablon brought up a Simple Password Exponential Key exchange (SPEKE)[12] [1]that was formed on the foundation of DH-EKE and provided forward secrecy. Then in1997 [13], Jablon extended his studies by applying A-EKE to SPEKE to result in a protocol called A-SPEKE. In the same paper Jablon augmented SPEKE to form B-SPEKE another password verified based protocol.

In 1998, Wu introduced the Secure Remote Password protocol (SRP) [14] that was based on the mathematical structure of the DH problem to provide forward secrecy and to be considered as another verifier-based protocol that protects against server compromise. SRP imposes some limitation on the ordering of its protocol's messages, and it also suffers from "two-for-one" attack; where an attacker can verify two passwords per impersonation attempt. In 2002, Wu improved and refined SRP to propose SRP-6 [15]. SRP-6 applied a simple change on the original SRP solving the "two-for-one" attack and adding more simplicity and flexibility, while maintaining the same security level of SRP.

---

[1] A "salt" is a random number that is added to the encryption key or to a password to protect them from disclosure. It is a value used to modify a hash of a password.

SRP have been adopted by *telnet* and *ftp,* and two IETF have been approved: RFC 2944 and RFC 2945.

In 2002, Jablon proposed a hybrid of SRP and B-SPEKE schemes resulting in a PAKE protocol called SRP-4 that jointed the benefits of each. It uses an optimized exponential computation similar to that used in SRP, and a prime order password-derived generator similar to that is used in B-SPEKE.

In 1999, Kwon and Song derived from SRP a protocol that they called GXY [16]. One year later, SRP and GXY were the motivation for Kwon to suggest a new protocol, called Authentication and key-agreement via Memorable Passwords (AMP) [17], which was considered as another verifier-based protocol that provided forward secrecy. AMP general idea was to amplify the low entropy of the password. Kwon also proposed in the same paper several variants of AMP, such as AMP^i, AMP^n, AMP+, and AMP++.
In 2003, Hwang, Yum, and Lee introduced Efficient Password-based protocol for Authenticated key Exchange (EPA) and its variant (EPA+) [18]. Then in 2005, Kwon, Park, and Lee [19] proved that EPA is vulnerable to server compromise and EPA+ is impotent in terms of scalability.

PAK, PPK, PAK-X are developed by Boyko, MacKenzie, and Patel [20], in 2000. The PAK protocol, which is a variant of DH-EKE with a very simple encrypted function, extended the group of verifier-based protocols. The More PAK variants PAK-EC, PAK-

R, PAK-XTR, PAK-Y, and PAK-Z were introduced in [21, 22]. PAK-Z was especially introduced to provide the forward secrecy property.

In 2004, Kwon blended the PAK and AMP to result in Three-Pass Authenticated key agreement via Memorable Passwords (TP-AMP) [23].

In 2000, Bellare and Rogaway came up with AuthA [24]. AuthA extends the DH-EKE subfamily, and its ideology lies behind the symmetric encryption primitive that is instantiated via either a cipher or a mask generation function that consists of the product or a hash of the password.

In 2001, M. Scott [25] introduced ESP-KE that is based on DH-EKE, and then modified it to be a verifier based protocol resulting in ESP-LIA. In his paper he pointed out that simple modification on ESP-LIA would result in ESP-LIB which was cheaper in terms of cost, because it reduced the processing burden on the client side

In 2002, Pretty Simple Password Authenticated Key Exchange (PSPAKE) was introduced by Kobara and Imai [26].

In 2001, Katz, Ostrovski, and Yung discovered a practical and provably secure protocol KOY [27], based on the decisional Diffie-Hellman assumption. One year later, they modified the KOY protocol to declare the satisfaction of the forward secrecy property.

Of all the proposed protocols, SPEKE [12], SRP [14], AMP [17], PAK [20], and AuthA [24] are being discussed by the IEEE P1363 standards working group for standardization on password-based public key cryptographic techniques.

4.3.3 The extension of the RSA-EKE subfamily

Lucks [28] called attention to the fact that using a secret key to encrypt a public key is counter intuitive; and thus constructed OKE by sending the public key in the clear and encrypting the second flow only. In his paper, he found out that his scheme is not really efficient in the RSA framework; thus he enhanced his original scheme to come up with the Protected OKE designed specifically to extend the RSA-EKE subfamily of Bellovin and Merritt.

In 1999, Mackenzie, Patel, and Swaminathan [21, 29] proved that Protected OKE was flawed and provided a solution in the context of a new protocol called (Secure Network Authentication with Password Identification) SNAPI. Unfortunately, SNAPI uses a prime public exponent *e* larger than the RSA modulus *n*. Therefore, Zhu et al. [30] proposed an interactive protocol RSA-PAKE1 and RSA-PAKE2 that avoided the use of large public exponents.

In 2004, Zhang introduced PEKEP and CEKEP [31] that allowed the RSA public exponents to take large and small values of prime numbers. CEKEP was derived from PEKEP, by adding two additional flows to achieve reduction in the computational load on entities.

4.4 Formal Definitions and Proofs

In the past, protocols used to be presented with no proofs; they were based on trial and error bases. Each protocol is presented, later on some other researcher(s) will discover flaws and augment the protocol to resolve the problem and come out with a solution in terms of a new protocol, and so on. This situation continued until recently, when practitioners in this field decided to set well structured foundations for this domain in terms of formal definitions and proofs.

In 1993, Gong, Lomas, Needham and Saltzer were the first to introduce heuristic resiliency to off-line password guessing attacks [32] in the asymmetric PKI model; where the user possesses a password and a public key. Later, Bellare and Rogaway presented the first formal model of security [33] for the symmetric two party case, and then extended their work to be applicable to the three party case [34]. In 1997, Wilson, Johnson, and Menezes [35] extended the work to be applicable in the asymmetric setting. It is very important to note that all these proofs did not include any consideration of password authentication.

In 1998, Halevi and Krawczyk [36] provided the first rigorous treatment of the issue of password authentication formulated by definitions and proofs of security in the PKI setting. They assumed that the server used a public key, while the user was assumed to have access to that public key either by direct access to a reliable server or by keeping what they called a "public-password" which was nothing but a digest of the public key.

Their study imposed inconveniency on the user side, because the user was required to keep handy a hard to memorize public key. In addition if the user needs to communicate with multiple servers then he must store multiple public keys. Alternatively, it has been suggested that a user can keep handy one public key to a Certification Authority (CA) and then CA is contacted multiple times for completing the authentication process. Unfortunately, this did not solve the problem in hand, instead the suggestion stretched it to another dimension. It is noteworthy to mention that their proposal lacks the consideration of two human beings communicating rather than a human being and a server. One year later, Boyarsky [37] extended their protocol to be applicable in the multi-user case.

In 2000, formal definitions were given by Bellare, Pointcheval, and Rogaway [38] for a more difficult setting, where the parties share only a password, based on the central idea of EKE and building on[33, 34]. They proposed an indistinguishability-based definition; together with protocols analyzed in the random oracle/ideal cipher models.
For example, AuthA is proved to be secure in the ideal cipher and soon after in the random oracle model.

On the other hand Boyko, MacKenzie, Patel, and Swaminathan [20] proposed a simulation-based definition building on the work of Bellare, Canetti, Krawczyk [39], and Shoup [40]. They introduced a new RSA and Diffie-Hellman-based protocols, and proved their security in the random oracle model.

In the random oracle model, the communicating parties are assumed to have access to random universal functions. Their protocol have been proved insecure in the case were the random universal function is replaced by some kind of a specific function, such as uniformly selected functions. As an example of the protocol that have been approved in the random oracle model, SNAPI, SNAPI-X, AMP, TP-AMP, PAK, PAK-X, PAK-Z, OKE, and AuthA.

In 2001, Goldreich and Lindell [41] introduced a new security definition and also gave the first provably secure solution to this problem in the standard model, based on general assumptions and no ideal ciphers/ random oracles. Although their protocol conceded a great idea, it was just impractical. Their protocol was recently simplified at the expense of achieving a weaker security guarantee by Nguyen and Vadhan [42]. PSPAKE is an example of the protocols that have been proved in the standard model.

In 2001, the first practical and provably secure solution was proposed by Katz, Ostrovsky, and Yung [27], based on the decisional Diffie-Hellman assumption, their protocol KOY completely eliminated the need of random oracle but assumed the communicating parties had access to public parameters.

In 2004, KOY protocol was generalized and abstracted by Gennaro and Lindell [43] which allowed them to construct a general framework in the common reference string model. The parties in the KOY/GL protocol exchange secure encryptions of the password, encrypted with the public-key found in the common reference string, and then compute the session key by combining smooth projective hashes of the two ciphertext/password pairs. The smooth projective hashing was introduced by Cramer and Shoup in 1999 [44].

In 2005 Canetti, Halevi, Katz, Lindell, and MacKenzie agreed that none of the above definitions stressed on the realistic setting where the password-based protocol is in reality a part of a larger protocol. Thus, they came out with Universally Composable (UC), a new definition of security for password-based key exchange protocols [45, 46] within the UC security framework [47]. Their protocol is based on KOY/GL, and it assumed that in addition to low entropy passwords, all parties share an ideal functionality that acts as a centralized trusted service available to the communicating parties. For example, KEA+C [48] was proved in this model.

4.5 Password-verifier Based Protocols and Threshold Authentication

A-EKE, AMP, TP-AMP, B-SPEKE, SRP, GXY, SNAPI-X, AuthA, PAK-Z are classified as password-verifier based protocols that protected from the compromise of the server's password file, so that an attacker with the password file will still have to perform a dictionary attack. Some work has been done to further protect against the server compromise problem; the ideology is to share the user's password with multiple severs, and these servers cooperate in a threshold manner to authenticate the user.

Thus Ford and Kaliski [49] and Jablon [50] proposed a threshold in which *n* servers have to all cooperate to check the authenticity of the user. The protocol is secure if utmost *n-1* of these servers were compromised. Both of these protocols have not been rigorously researched and thus provided no proof of security. Then in 2002, MacKenzie et al. [51] presented a protocol where *x* out of *n* servers have to cooperate in authenticating the user. The protocol is secure if utmost *x-1* of these servers were compromised. They have proved their protocol to be secure in the random oracle model. One year later, Raimondo and Gennaro [52] proposed a protocol that is secure if utmost one-third of the servers are compromised. Their protocol has been proven secure in the standard model. In the same year, Brainard et al. [53] have introduced "Nightingale" protocol that uses only two servers, and he proved its security in the random oracle model. In 2005, Katz, Mackenzie, Taban, and Gligor [54] proposed a two-server protocol that and proved to be secure in the standard model.

4.6 Summary

In this chapter we discussed PAKE protocols that have received growing attention recently because such protocols are much easier to use, simpler to implement, and less expensive to deploy than any other authentication mechanism. The evolution of such protocols has been enormous, due to the fact that in such protocols users should hold a password only and no other external infrastructure.

Although the practitioners have made incredible contribution to reach a great state of security, up to this date problems still exists but in a different dimension. The research in this field was performed in a narrow minded fashion. Looking at the forest through the trees would require us to step out of the circle and look at the problem from a broader point of view. Therefore, we propose to use UserID as another security measure that will complement the role of the password as the only line of defense without hindering the convenience of the authentication process.

In the next chapter, we will introduce our protocol that implements our main idea in a practical setting.

Chapter 5: Mutual Authentication and Key Exchange using Usernames and Passwords (MAKE-UP)

In the previous chapter we discussed PAKE protocols and all related issues. We noticed that the evolution of such protocols have started to address this issue successfully, but there is much more to do.

In this chapter we introduce our protocol, Mutual Authentication and Key Exchange using Usernames and Passwords (MAKE-UP) that will apply our ideology in a practical setting to solve numerous problems in existing PAKE protocols resulting in a more secure environment.

5.1 Preliminaries

We assume that the adversary have full control over the network. In particular, the adversary can read the messages produced by the parties, provide messages of his or her own to them, modify messages before they reach their destination, and delay messages or replay them. Most importantly, the adversary can start up entirely new instances of any of the parties, modeling the ability of communicating agents to simultaneously engage in many sessions at once.

5.2 MAKE-UP ideology

This thesis suggests to keep the userID secret and to use it as an equivalent second password, thus making the password-based authentication more secure than ever. The idea presented is generic enough to be integrated with other password based authentication protocols.

To exemplify the ideology, an Automatic Teller Machine (ATM) verifies two factors: Something a user has such as (ATM card), and something a user knows such as a Personal Identification Number (PIN). Making Usernames public is somewhat similar to the situation where a thief had stolen an ATM card, but does not know the secret PIN number, although he knows it is a number in the range 0000-9999. It is important to note that userID represents a factor of what users know rather than what users have. Nevertheless, having two factors of what a user knows is certainly better than one.

The proposed scheme will not only act as doubling the security measures of password authentication, but will certainly strengthen the existing protocols. If the userID is augmented as a second security measure in any of the on hand protocols, a noticeable result of additional complications for attackers is obvious. Attackers guess at user names and passwords, thus if they do obtain the user name of a legitimate user then impersonating him is a simpler matter and requires less effort.

Therefore, we introduce a protocol Mutual Authentication and Key Exchange using Usernames and Passwords (MAKE-UP), to implement our ideology of using the userID as a complement to the password, and building on the general idea of two factors better than one. Noting that the two factors used in the authentication process stem from the "what a user know" authentication technique that achieves the best proper balance of all the authentication requirements discussed in the previous chapters.

5.3 Desirable protocol characteristics

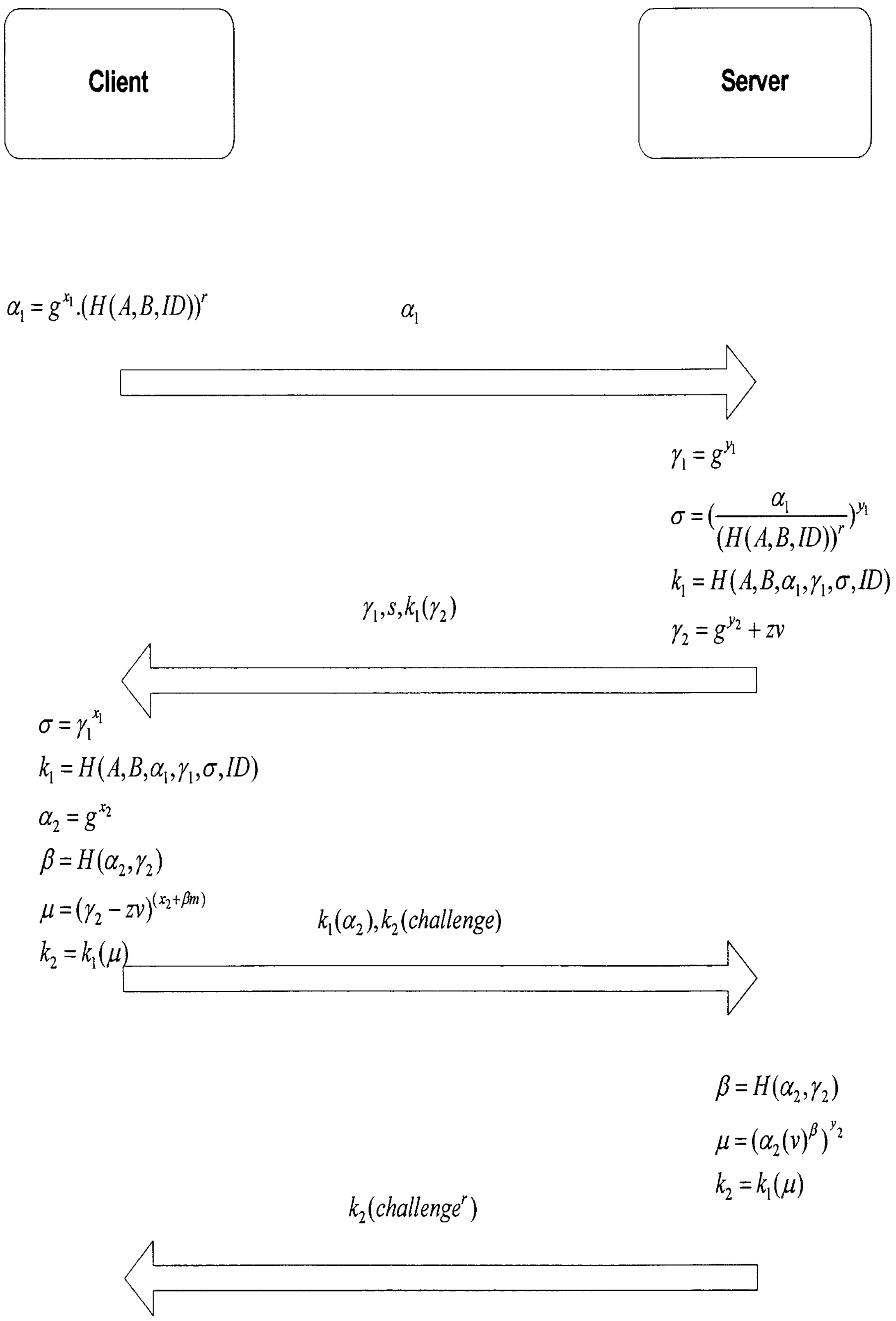
In the previous chapter, we introduced PAKE goals and characteristics. Whereas, in this section we will list MAKE-UP desirable characteristics that are derived from the strict requirements for secure authentication protocols initiated by the internet Engineering Task Force (IETF).

1. No useful information about the password *pw* or userID *ID* or their associated private information is revealed during a successful run. Attackers are unable to guess and verify passwords and userIDs based on exchanged messages (resiliency to eavesdropping attack).
2. No useful information about the session key *K* is revealed to an eavesdropper during a successful run.
3. Even if an intruder is able to alter or create his own messages and make them appear to originate from the server or the client, the protocol should prevent the intruder from gaining access to the host or learning any information about usernames, passwords, or session keys.
4. If the host's password file is captured and the intruder learns the verifier value *v*, the intruder should not be able to impersonate the user (resiliency to file compromise).
5. If the session key *k* of any past session is compromised, the intruder should not be able to impersonate the user directly or indirectly by deducing the user's password and/or username (resiliency to Denning-Sacco attack).
6. If the user's password *pw* itself is compromised, it should not allow the intruder to determine the user's *ID*.

7. If the user's *ID* itself is compromised, it should not allow the intruder to determine the user's password *pw*.
8. If the user's *ID* and/or user's password *pw* are compromised by a passive adversary (eavesdropper), he or she can not ascertain anything about session keys.
9. If the user's *ID* and/or user's password *pw* are compromised, it should not allow the intruder to determine the session key for past sessions and decrypt them (perfect forward secrecy).

Now, let us look at the design of the MAKE-UP protocol to make sure that MAKE-UP fulfills the requirement of a secure authentication protocol, satisfies our desirable protocol goals and characteristics, and makes up for the past years of insecurity and vulnerability on the price of slight added complication in regard to the extra number of computations that it requires in comparison with other PAKE protocol.

The following figure shows the MAKE-UP protocol.



Client
Server
$\alpha_1 = g^{x_1}.(H(A,B,ID))^r$
α_1
$\gamma_1 = g^{y_1}$
$\sigma = (\frac{\alpha_1}{(H(A,B,ID))^r})^{y_1}$
$k_1 = H(A,B,\alpha_1,\gamma_1,\sigma,ID)$
$\gamma_2 = g^{y_2} + zv$
$\gamma_1, s, k_1(\gamma_2)$
$\sigma = \gamma_1^{x_1}$
$k_1 = H(A,B,\alpha_1,\gamma_1,\sigma,ID)$
$\alpha_2 = g^{x_2}$
$\beta = H(\alpha_2,\gamma_2)$
$\mu = (\gamma_2 - zv)^{(x_2+\beta m)}$
$k_2 = k_1(\mu)$
$k_1(\alpha_2), k_2(challenge)$
$\beta = H(\alpha_2,\gamma_2)$
$\mu = (\alpha_2(v)^\beta)^{y_2}$
$k_2 = k_1(\mu)$
$k_2(challenge^r)$

Figure 5: MAKE-UP protocol

5.4 Implementation Details

In this section we will go through the steps of the protocol and describe how the protocol works. Let us first define the symbols and terms used by our protocol. Table 3 shows these terms with their definitions.

A	Client's name
B	Server's name
ID	Client's *ID*
pw	Client's password
n	A large safe prime ($n=2p+1$, where p is a prime)
g	A generator modulo n
z	Multiplier parameter ($z = H(n, g), z = 3$)
r	Random number used as a nonce
s	Salt
v	Password verifier ($v = g^m$, $m = H(s,pw)$)
m	A private key derived from the password and the salt
H()	One way hash function
σ	Common exponential value
B	Random scrambling parameter
μ	Hash exponential value
x_1, x_2, y_1, y_2	Secret ephemeral values
α_1	Client public ephemeral value
γ_1	Server public ephemeral value
α_2	Client secret ephemeral value
γ_2	Server secret ephemeral value
k_1	Premier session key
k_2	Session key

Table 3: Mathematical notations for MAKE-UP

Further detailed description of the symbols use will follow later on. All computations are performed in a finite field *GF(n)*. In other words, a large prime number *n* is chosen ahead of time, and all additions, multiplications, and exponentiations are performed modulo *n*. To establish a password *pw* with the Server, the Client picks a random salt *s*, and computes

$v = g^{H(s, pw)}$. The Server stores v and s as Client's password verifier and salt, associated with the Client's *ID*. Note that the value $H(s, pw)$ is discarded because it is equivalent to the plaintext password pw.

The server keeps (ID, s, v) in its password database. Then authentication proceeds as follows:

Step 1: The Client sends his ephemeral public key α_1.

Step 2: The Server sends the salt s, ephemeral public key γ_1, and secret ephemeral key γ_2 encrypted with the premier session key k_1.

Step 3: The Client sends his secret ephemeral key α_2 encrypted with the premier session key k_1, accompanied with a random challenge encrypted with the resulting session key k_2.

Step 4: The server sends the challenge raised to the nonce encrypted with the resulting session key k_2.

Hereafter, each of these steps is explained with more details.

5.4.1 Protocol step 1

When the client wishes to authenticate himself or herself, he or she generates Client public ephemeral value α_1 and sends it to the server as a proof that he or she knows the Client ID *ID*. While generating α_1, the Client ID *ID* is kept secret and is hashed with the Client's name A and the Server's name B by a one way hash function H. The output of the hash function is raised to the power of a random number r that is used as a nonce to insure freshness and prevent replay attacks. Then, the result is multiplied by g^{x_1}, where x_1 is nothing but a secret ephemeral value. It should be noted that the use of secret ephemeral values x_1, x_2, y_1 and y_2 guarantees that a compromised session key in a given session does not cause the compromise of any earlier session.

5.4.2 Protocol step 2

The server checks if the Client ID *ID* is valid, then computes the session group element σ and premier session key k_1 using available values and generates Server public ephemeral value γ_1 and Server secret ephemeral value γ_2.

To compute the session group parameter σ, the Server divides Client public ephemeral value α_1 by the hash of the Client name A, Server name B, and the ClientID ID raised to the power of the nonce r; then raising the result to the power of his or her secret ephemeral value y_1.

To compute k_1, the Client name A, Server name B, Client public ephemeral value α_1, Server public ephemeral value γ_1, session group element σ, and the ClientID *ID* are all hashed using a one way hash function H.

While generating γ_2 we added the verifier v that allows Server authentication while preventing an active dictionary attack, carried out by an attacker who pretends to be a legitimate host. However, the way in which the residues g^{y_2} and v are combined must be selected carefully. If we denote the combining function used to compute γ_2 as $\gamma_2 = f(v, g^{y_2})$, then we have to avoid using functions $f()$ that have the property that $f(g^x, g^y) = f(x;y)$ for some easily derived $f()$. The attack described above can be extended to situations where $f()$ has this undesirable property. This rules out, for example, $f(x,y) = xy$. In addition, we also need the value of γ_2 to leak as little information about v as possible, which rules out $f(x,y) = x \oplus y$ or $f(x,y) = E_y(x)$, where $E_k()$ is a symmetric encryption algorithm. In either of these cases, an attacker can carry out a partition attack.

Modular addition appears to be the simplest operation that leaks no information about v while at the same time enabling MAKE-UP to resist a dictionary attack by a fake host.

Additionally, g must be a primitive root of $GF(n)$ in order to make all values of γ_2 equiprobable for any v. Because of the symmetry of the equation for the server's value, it is possible for an attacker to insert a password guess. One simple way to remove the symmetry in the server's key exchange value is to multiply by some value z agreed by both sides.

Server's public ephemeral value γ_1 will help the Client but no one else to generate the primary session key k_1 so that the Client and only the Client will be able to decrypt the server's ephemeral value γ_2. The salt s is sent to allow the Client to compute his or her private key m using s and his or her real password pw.

5.4.3 Protocol step 3

The client generates his or her secret ephemeral value α_2, and computes the session group element σ and the primary session key k_1.

The session group element σ is computed by raising the Client public ephemeral value γ_1 to his or her secret ephemeral value x_1 resulting in the same session group element σ as the one the server possesses.

The primary session key k_1 is computed in a similar way which the server used by utilizing the same available values that they both share.

Then the client computes the random scrambling parameter β by hashing the secret ephemeral value of the client and the server α_2 and γ_2. By computing the hash exponential value μ, the clients authenticate that real identity of the server because no one else know the password verifier and then he or she authenticates himself by including m that is derived from the salt s received from the server and from using his own real password. The Client encrypts the hash exponential value μ by the primary session key k_1 to result in the session key k_2. Then the client chooses a random challenge and encrypt it with the resulting session key k_2 as a proof of its knowledge, and at the same time to challenge the server to prove his knowledge of it. This action is the first step in the key confirmation phase.

Then the client encrypt his secret ephemeral value α_2 by the session key k_1 as a proof of his or her knowledge of the premier session key k_1 and to provide the server with his secret ephemeral value so that he can compute the resulting session key k_2 to be able to decrypt the challenge.

5.4.4 Protocol step 4

The server decrypts the client secret ephemeral value α_2, while verifying that the client actually possesses the correct premier session key k_1.

Then, the server computes the random scrambling parameter β by hashing the secret ephemeral value of the client and the server, then computes hash exponential value μ and checks if he have had received the same hash exponential value μ from the client, then encrypt the hash exponential value μ by the primary session key k_1 to result in the cryptographically strong session key k_2.

The server decrypts the challenge, raise it to the nonce, and encrypt it with the computed value of the session key k_2 as a response to the challenge, a proof that he or she has knowledge of the resulting session key, and to complete the key confirmation phase.

The key confirmation phase of this protocol rests behind the challenge-response methodology in steps 3 and 4. The methodology is based on the fact that if one of the communicating entities sends challenge c encrypted by k, where c was never used before, and receives response that contains c, it means that the originator of the message has the ability to encrypt message with k. It is possible to replace this mechanism for validating k by other means. For example, if the clocks between communicating parties are monotonic and synchronized then the time could be exchanged encrypted by k.

5.4.5 Constraint checks

The following list of constraints must be performed by both sides to ensure the security of the MAKE-UP Protocol.

- **n is a large prime number**: The client must ensure that n is large enough to resist attacks. A safe prime n can be expressed as $2p + 1$, where p is a prime.
- **g is a primitive root of $GF(n)$**: g must be a generator modulo n, which means that for any x where $0 < x < n$, there exists a value x for which $g^x \ (mod\ n) = x$.
- **x_1,x_2,y_1,y_2 > log g(n)**: Otherwise the operation (mod n) is not used and an attacker could take the algebraic logarithm of g^{x_1} to recover x_1.

5.5 Security and analysis

The security of MAKE-UP is based on its mathematical structure that relies on two familiar problems, which are believed to be infeasible to solve in polynomial time.

The first one is, Discrete Logarithm Problem: Assume $f(x) = g^x$ where the base g and the implied modulus n are publicly known and agreed-upon values. Computing $f(x)$ is known as discrete exponentiation, and its inverse is known as a discrete logarithm. Finding discrete logarithms is a problem long believed to be computationally difficult for large values of n (512 bits or longer). n must be a non-smooth prime; in other words $n-1$ must not consist entirely of small factors; this will harden the calculations of discrete logarithms in $GF(n)$. To avoid confinement attacks, where an attacker forces the session key used by either party to be confined to a small subgroup of $GF(n)$, MAKE-UP chooses n as a safe prime $n=2p+1$; where p is also prime.

The second one is, Deffie-Hellman Problem: The simplest, and original, implementation of the protocol uses the multiplicative group of integers modulo p, where p is prime and g is a primitive mod p. Modulo (or mod) simply means that the integers between 1 and $p-1$ are used with normal multiplication, exponentiation and division, except that after each operation the result keeps only the remainder after dividing by p. Both parties agree on a finite cyclic group G and a generating element g in G. This is usually done long before the rest of the protocol; g is assumed to be known by all attackers.

1. Client picks a random natural number a and sends g^a to Server.
2. Server picks a random natural number b and sends g^b to Client.
3. Client computes $(g^b)^a$.
4. Server computes $(g^a)^b$.

Both Client and Server are now in possession of the group element g^{ab} which can serve as the shared secret key. The values of $(g^b)^a$ and $(g^a)^b$ are the same because groups are power associative. The security of the technique depends crucially on the difficulty of computing logarithms mod p .If large values are used for a, b, and p then even the best known algorithms for finding a given only g, p, and g^a mod p (known as the discrete logarithm problem) would take longer than the lifetime of the universe to run. However, The Diffie-Hellman protocol neither provides any form of authentication nor protects against man in the middle attacks.

The mathematical structure of MAKE-UP uses these two problems efficiently to provide authentication and to protect against a number of security attacks. MAKE-UP is proved secure in the random oracle model. Our security is based on shoup [40] that is based on [39]. In [40], the security is defined using an ideal model, which defines the service to be provided, and a real system, which describes the world in which the involved parties and the adversaries work. The security of the protocol is proven by demonstrating that anything the adversary can do in the real system can also be done in the ideal system. This can be done by simulating the realistic settings in the real world to that of the ideal world.

We define an ideal world model in which a sleeping adversary exists, which is unable to do anything. We also define a real world model that describes what an adversary can and can not do in real life.

Then for both real world and ideal world adversaries, a transcript is generated that logs all important events as they happen. Security is provided by ensuring that for every real world adversary there exists an ideal world adversary, such that the transcripts that both of them generate are computationally indistinguishable.

The next figure clarifies the ideology.

Ideal
world

Figure 5: Realization of the ideal world and the real world

5.6 Summary

In this chapter we introduced MAKE-UP, as a verifier-based password protocol that provides mutual authentication of both parties. We showed that it met the severe requirements laid down in (RFC 1704) for a non-disclosing authentication protocol and satisfied all our desirable protocol characteristics.

MAKE-UP provides perfect forward secrecy, and is classified as zero-knowledge protocols since the protocol do not even leak any information about the password to the legitimate host. It protects against a number of attacks, such as: active, passive, on-line, off-line, small group confinement, Denning-Sacco, and others.

Nonetheless, MAKE-UP has light constraint and is easy to generalize and implement, and is ideal for a wide range of real-world applications in which strong secure password authentication is required.

The next Chapter is devoted to the conclusion of this thesis. It will summarize our contributions and present some possible extensions of our work.

Chapter 9: Conclusion

The importance of computer security is continuously increasing with the rapid growth of computer technology, and becomes nowadays the most essential aspect in the computing world.

Fifty years ago, computer crimes were not heard of. New technologies bring new problems, as the computer technology progresses dramatically; the phenomena of computer crimes threaten computer security. Nowadays, more and more computer crimes are committed; to cope with the current crime rate, computer security is becoming a fundamental need. Effective computer security is now more crucial than ever, and the need to increase awareness is compelling.

Their appreciated findings in this domain have led to incredible levels of security that contributed to magnificent advancements in the computing world. In order to advance more, we have to accept the fact that security is not static, and the risks are always there and therefore do our utmost to minimize those risks as far as is humanly and technologically possible.

In this thesis, we have addressed authentication which is one of the most important factor in the computer security world. Then we focused on passwords and password authentication in an intensive manner and provided some useful key points to reinforce the weakest link in the authentication process. Our results in this thesis were enriched with remarkable statistics gathered from a questionnaire that was distributed to over 500 participants with different backgrounds.

We then focused on password authentication protocols that are deployed without depending on an expensive external infrastructure. Only recently, significant attention has been paid to such protocols, whereas traditionally password authentication protocols used symmetric encryption, public-key encryption, or a combination of the two approaches to resist common attacks. In this thesis, we have demonstrated that several of the earlier methods were flawed, the surviving and enhanced forms have started to address this problem with some success, but that there was still room for improvement.

9.1 Achievements

This thesis has presented a summary on passwords and authentication, and then listed the requirements of password authentication systems and the possible attacks against passwords. We then introduced solutions for strengthening password authentication systems.

As we proved the importance to measure password strength using entropy, we suggested that password policies should encourage users to use password-based mnemonics as they are easy to remember as naively selected passwords and as hard to crack as randomly chosen ones, by combining the best of both while minimizing their drawbacks.

Then, we enhanced the proactive password checkers, by modifying its dictionary based checking by the inclusion of entropy based checking. Using both methodologies jointly solved the drawback of proactive password checkers and weak password choices with low entropy are no longer selected. Thus, secure password systems are strengthened by providing positive feedback to the user on his or her password strength.

We also introduced a solution to resist DOS attacks, by firstly applying a premier challenge quiz that could avoid machine guessing and delay a human being in its guessing attack. Nonetheless, we used an incremental approach for the time locking period to minimize the drawbacks of such a technique.

Then we researched Password-only Authentication and Key Exchange (PAKE) protocols that have received growing attention in recent years because such protocols are much easier to use, simpler to implement, and less expensive to deploy than any other authentication mechanism, due to the fact that in such protocols users should hold a password only and no other external infrastructure. This thesis provided a summary on all existing password-only based authentication and key exchange protocols, their enormous evolution, their goals and characteristics, and their formal definitions and proofs.

Our main contribution is the introduction of a new protocol Mutual Authentication and Key Exchange using Usernames and passwords as a password- based key exchange protocol that provided mutual authentication and amplified a shared password and userID into a shared secret session key, to be used for protecting subsequent communication over insecure networks by following the various notable predecessors. The key idea behind the MAKE-UP is to keep the userID secret and to use it as a compulsory form of user authentication. Our idea is general and easy to implement and overcomes most of the difficulties of previously suggested methods of improving the security of user authentication schemes to result in a more secure environment.

In our analysis, we put forth a series of requirements for a secure password protocol, emphasizing those that existing protocols failed to meet. We have verified that MAKE - UP , is a verifier-based, zero-knowledge protocol resistant to multiple attacks, which offers a number of security advantages that surely improves the authentication process, and certainly help solve numerous problems within the computer security field.

9.2 Future works

We are planning to put into practice the suggested theoretical solutions into existing password authentication systems, and are looking forward to seeing our protocol in practical use. Our main idea is so general; we expect it to be integrated in existing password authentication protocols and to be implemented in anticipated future ones.

References

[1] M. Albataineh and A. En-nouaary, “Strengthening Password Authentication Systems,” Sixth International Network Conference, Plymouth, U.K., July 2006.

[2] Hailer, "The 13 (TM) one-time password system," in Proceedings of the Internet Society Symposium on Network and Distributed System Security, pp. 151-158, 1994.

[3] M. Albataineh and A. En-nouaary, “Password-only based Authentication and Key Exchange Protocols: A Survey,” Submitted to International Journal of Security and Networks (IJSN), 2006.

[5] M. Lomas, L. Gong, J. Saltzer, and R. Needham, “Reducing risks from poorly chosen keys," ACM Symposium on Operating System Principles, pp.14-18, 1989.

[6] S.M. Bellovin and M. Merritt, "Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks," IEEE Symposium on Research in Security and Privacy, pp. 72-84, 1992.

[7] R. L. Rivest, A. Shamir, and L. Adleman, "A method of obtaining digital signatures and public-key cryptosystems," Communications of the ACM, Vol. 21, pp. 120-126, 1978.

[8] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Transactions on Information Theory, vol. IT-31, pp. 469-472, 1985.

[9] S.M. Bellovin and M. Merritt, "Augmented Encrypted Key Exchange: A Password Based Protocols Secure Against Dictionary Attacks and Password file Compromise," 1st ACM Conference on Computer and Communications Security, pp.244-250, 1993.

[10] R. Morris and K. Thompson, "Password Security: A Case History," Communications of the ACM, 22(11), pp. 594-597, 1979

[11] M. Stenier, G. Tsudik, and M. Waidner, “Refinement and Extension of Encrypted Key Exchange,” ACM SIGOPS Operating Systems Review, 1994.

[12] D. Jablon, "Strong Password-Only Authentication Key Exchange," ACM Computer Communication Review, Vol. 26, No. 5, pp. 5-26, 1996.

[13] D. Jablon, "Extended Password key Exchange Protocols Immune to Dictionary Attack. Proceedings of the WETICE Workshop on Enterprise Security, Cambridge, MA, June 1997.

[14] T. Wu, “The Secure Remote Password Protocol,” Proceedings of the Internet Society Symposium on Network and Distributed System Security, pp. 97-111, 1998.

[15] T. Wu, "SRP6: Improvements and refinements to the secure remote password protocol," October 2002.

[16] T. Kwon and J. Song, "Secure agreement scheme for g^{xy} via password authentication," Electronics Letters, vol.35, no.11, pp.892-893,1999.

[17] T. Kwon "Authentication and Key Agreement via Memorable Password," 2000.

[18] Y. Hwang, D. Yum, and P. Lee, "EPA: an efficient password-based protocol for authenticated key exchange," ACISP, Lecture Notes in Computer Science (2727), Springer-Veralg, Wollongong, Austrialia, July 2003.

[19] T.Kwon, Y. Park, and J. Lee, "Security Analysis and Improvement of the Efficient Password-based Authentication Protocol, " IEEE Communication Letters, vol. 9, no. 1, January 2005.

[20] V. Boyko, P. MacKenzie and S. Patel, "Provably secure Password Authenticated Key Exchange using Diffie-Hellman," Proceedings of Eurocrypt 2000, Springer-Verlag, pp 156-171, 2000.

[21] P. MacKenzie, S. Patel and R. Swaminathan, "Password-Authenticated Key Exchange based on RSA," Advances in Cryptology-ASIACRYPT'2000, pp. 599-613, 2000.

[22] P. MacKenzie. "The PAK suite: Protocols for password-authenticated key exchange". Contributions to IEEE P1363.2, 2002

[23] T. Kwon, "Practical Authentication Key Agreement using Passwords," School of Computer Engineering, Sejong University, Seoul 143-747, Korea, 2004.

[24] M. Bellare, P. Rogaway. The AuthA protocol for password-based authenticated key exchange," Contributions to IEEE P1363, March 2000.

[25] M. Scott, "Efficient Short-Password Key Exchange and Logging Protocols," School of Computer Applications, Dublin City University, Ireland, 2001.

[26] K. Kobara and H. Imai, "Pretty-simple password-authenticated key-exchange under standard assumptions," IEICE Transactions, E85-A(10):2229-2237, Oct. 2002.

[27] J. Katz, R. Ostrovsky, and M. Yung, "Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords," Adv. in Cryptology, Eurocrypt 2001, LNCS vol. 2045, Springer-Verlag, pp. 475-494, 2001.

[28] S. Lusks, "Open key exchange: How to defeat dictionary attacks without encrypting public keys," The Security Protocol Workshop '97, 1997.

[29] P. MacKenzie and R. Swaminathan, "Secure Network Authentication with Password Identification," Presented to IEEE P1363a, August 1999.

[30] F. Zhu, A. H. Chan, D. S. Wong, and R. Ye, "Password Authenticated Key Exchange based on RSA for Imbalanced Wireless Network," In Proc. of ISC '02, LNCS 2433, pages 150-161, Springer-Verlag, Berlin, 2002

[31] M. Zhang. "New Approaches to Password Authenticated Key Exchange based on RSA," Advances in Cryptology - ASIACRYPT 2004, pp. 230-244, 2004.

[32] L. Gong, M. Lomas, R. Needham and J. Saltzer, "Protecting Poorly Chosen Secrets from Guessing Attacks," IEEE Journal on Selected Areas in Communications, 11(5), pp. 648-656, June 1993.

[33] M. Bellare, and P. Rogaway, "Entity Authentication and Key Distribution," Advances in Cryptology, 1993.

[34] M. Bellare and P. Rogaway, "Provably Secure Session Key Distribution - The Three Party Case," Proceedings of the 27th ACM Symposium on the Theory of Computing, pp. 57-66, 1995.

[35] S. Blake-Wilson, D. Johnson and A. Menezes, "Key agreement protocols and their security analysis," Sixth IMA Intl. Conf. on Cryptography and Coding, No. 1355, pp. 30-45, 1997.

[36] S. Halevi and H. Krawczyk, "Public-Key Cryptography and Password Protocols," ACM Transactions on Information and System Security, 2(3), pp. 25-60,1999.

[37] M. Boyarsky, "Public-key Cryptography and Password Protocols: The Multi-User Case," 5th ACM Conference on Computer and Communications Security, pp. 63-72, 1999.

[38] M. Bellare, D. Pointcheval and P. Rogaway, "Authenticated Key Exchange Secure Against Dictionary Attacks," Advances in Cryptology-EUROCRYPT'2000, pp. 139-155, 2000.

[39] M. Bellare, R. Canetti, and H. Krawczyk, "A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols, " Proceedings of the 30th STOC, ACM Press, New York, 1998.

[40] V. Shoup, "On Formal Models for Secure Key Exchange," Cryptology ePrint Archive, Report 1999/012 , 1999.

[41] O. Goldreich and Y. Lindell, "Session-Key Generation using Human Passwords Only," Advances in Cryptology – Crypto 2001, LNCS vol. 2139, Springer-Verlag, pp. 408–432, 2001.

[42] M. Nguyen and S. Vadhan, " Simpler Session-Key Generation from Short Random Passwords," Proceedings of the 1st Theory of Cryptography Conference (TCC'04), LNCS vol. 2951, Springer-Verlag, pp. 442-445, 2004.

[43] R. Gennaro, and Y. Lindell, "A Framework for Password-Based Authenticated Key Exchange," Advances in Cryptology, 2004.

[44] R. Cramer and V. Shoup. "A Practical Public Key Cryptosystem provably secure against Chosen Ciphertext Attack". Proceedings of Crypto '98, Springer-Verlag,1999.

[45] R. Canetti, "Universally Composable Security: A New Paradigm for Cryptographic Protocols," 42nd IEEE Symposium on Foundations of Computer Science (FOCS), pp. 136–145, 2001.

[46] R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. MacKenzie, "Universally Composable Password-Based Key Exchange," In Proceedings of Eurocrypt 2005, pp. 404-421, 2005.

[47] R. Canetti, "Universally Composable Security: A New Paradigm for Cryptographic Protocols," 42nd IEEE Symposium on Foundations of Computer Science (FOCS),IEEE, pp. 136–145, 2001.

[48] K. Lauter and A. Mityagin, "Security Analysis of KEA Authenticated Key Exchange," August 2005.

[49] W. Ford and B.S. Kaliski, "Server-Assisted Generation of a Strong Secret from a Password," Proc. 5th IEEE Intl. Workshop on Enterprise Security, 2000.

[50] D. Jablon, "Password Authentication Using Multiple Servers", RSA Cryptographers' Track 2001, LNCS vol. 2020, Springer-Verlag, pp. 344–360, 2001.

[51] P. MacKenzie, T. Shrimpton, M. Jakobsson, "Threshold Password-Authenticated Key Exchange," 2002.

[52] M. Di Raimondo and R. Gennaro, "Provably Secure Threshold Password-Authenticated Key Exchange," Advances in Cryptology, Eurocrypt 2003, LNCS vol. 2656, Springer-Verlag, pp. 507-523, 2003.

[53] J. Brainard, A. Juels, B. Kaliski, and M. Szydlo. Nightingale, "A New Two-Server Approach for Authentication with Short Secrets," 12th USENIX Security Symposium, pp. 201–213, 2003.

[54] J. Katz, P. MacKenzie, G. Taban, and V. Gligor, "Two-Server Password-Only Authenticated Key Exchange," Applied Cryptography and Network Security (ACNS) Springer-Verlag, pp. 1-16, 2005.