3-PARTY KEY AGREEMENT PROTOCOL SECURE AGAINST ONLINE AND DICTIONARY ATTACKS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
In
Computer Science & Engineering

By
SAIRAM KULKARNI

Department of Computer Science & Engineering
National Institute of Technology
Rourkela
2007

3-PARTY KEY AGREEMENT PROTOCOL SECURE AGAINST ONLINE AND DICTIONARY ATTACKS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
In
Computer Science & Engineering

By
SAIRAM KULKARNI

Under the Guidance of
Prof.S.K.Jena

Department of Computer Science & Engineering
National Institute of Technology
Rourkela
2007

National Institute of Technology

Rourkela

CERTIFICATE

This is to certify the thesis entitled, **3-PARTY KEY AGREEMENT PROTOCOL SECURE AGAINST ONLINE AND DICTIONARY ATTACKS** submitted by **Sri.Sairam Kulkarni** in partial fulfillment of the requirements for the award of Master of Technology Degree in Computer Science and Engineering at the National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date: 21-05-2007

Prof. S. K. Jena
Professor&Head
Dept. of Computer Science and Engg.
National Institute of Technology
Rourkela – 760008

Acknowledgement

I am grateful to numerous local and global peers who have contributed towards shaping this thesis. At the outset, I would like to express my sincere thanks to Professor S.K.Jena for his advice during my thesis work. As my supervisor, he has constantly encouraged me to remain focused on achieving my goal. His observations and comments helped me to establish the overall direction of the research and to move forward with investigation in depth. He has helped me greatly and been a source of knowledge.

I thank Prof.S.K.Rath(Dean-Academics) for his constant encouragement and support during my postgraduate studies. I would like to express my special thanks to Prof.B.Majhi for being a great soul. He is always ready to help with a smile. I would like to thank Prof. R. Baliarsingh, Prof. A. K. Turuk, Prof. D. P. Mohapatra, Prof. B. D. Sahoo, Prof. S. Chinara, Prof. Sabuj K. Jena for their valuable suggestions.

I express my gratitude to Prof. P. K. Sa for generously sharing his time and knowledge and for making life fun while working.

I would like to thank administrative and technical staff members of the Department who have been kind enough to advise and help in their respective roles. Throughout Post graduate studies I have been fortunate to have wonderful support structure among the post graduate students.Iam really thankful to my friends. My sincere thanks to everyone who has provided me with kind words, a welcome ear, new ideas, useful criticism, or their invaluable time, I am truly indebted. Last, but not least, I would like to dedicate this thesis to my family,for their love, patience, and understanding.

(Sairam Kulkarni)

Contents

1 INTRODUCTION **1**

1.1 ELEMENTS OF INFORMATION SECURITY 2

1.2 LAYOUT OF THESIS 3

2 OVERVIEW OF KEY AGREEMENT PROTOCOLS **5**

2.1 DEFINITIONS 6

2.2 MOTIVATION FOR USE OF SESSION KEYS 7

2.3 DESIGN GOALS 8

2.4 ADVERSARIES IN KEY ESTABLISHMENT PROTOCOLS 10

3 TRADITIONAL KEY DISTRIBUTION PROTOCOLS **12**

3.1 INITIAL ASSUMPTIONS 12

3.2 2-PARTY KEY DISTRIBUTION PROTOCOL (2PKDP) 14

3.2.1 Security Analysis 14

3.3 3-PARTY KEY DISTRIBUTION PROTOCOL (3PKDP) 17

3.3.1 The Protocol 17

3.3.2 Insider Attacks 18

4 2-PARTY KEY AGREEMENT PROTOCOLS **19**

4.1 DIFFIE-HELLMAN PROTOCOL 19

4.1.1 The Protocol 23

4.1.2 Attacks on Diffie-Hellman 24

4.1.3 Results And Discussion 27

4.2 ENCRYPTED KEY EXCHANGE PROTOCOL 29

4.2.1 Generic EKE 29
4.2.2 EKE with Diffie-Hellman key exchange 30
4.2.3 Results And Discussion 31
4.3 SAKA PROTOCOL 35
4.3.1 The Protocol 35
4.3.2 Security Analysis 36
4.3.3 Results And Discussion 37

5 3-PARTY KEY AGREEMENT PROTOCOLS 40
5.1 STW PROTOCOL 40
5.1.1 The Protocol 41
5.1.2 Undetectable on-line guessing attacks 42
5.2 LSH 3-PEKE PROTOCOL 43
5.2.1 The Protocol 44
5.2.2 Results And Discussion 45

6 MODIFIED 3-PARTY KEY AGREEMENT PROTOCOL 51
6.1 THE PROTOCOL 51
6.2 COMMUTATIVE ONE-WAY HASH FUNCTIONS 54
6.3 SECURITY ANALYSIS 54
6.4 RESULTS AND DISCUSSION 55

7 CONCLUSION AND FUTURE WORK 64

List of Tables

3.1 Terminology used 12

Abstract

Frequent key changes are must in order to limit the amount of data compromised. Cryptography simply can not get off the ground without effective key distribution mechanism.Several key agreement protocols are proposed on password based mechanism. These protocols are vulnerable to dictionary attacks. Traditional 3-party key agreement protcols are vulnerable to insider attacks and server becomes a monitoring centre which we dont want in most of the applications.EKE protocol is vulnerable to Denning -Sacco attacks. EKE demands storing clear text version of password on server which is always not possible. STW protocol was proved to be vulnerable to on-line and off-line guessing attacks as it lacks server authentication to hosts. LSH 3-PEKE uses server public keys but its not an optimistic solution.the approach of using server public keys is not always a satisfactory solution and is impractical for some environments. Communication parties have to obtain and verify the public key of the server, a task which puts a high burden on the user. SAKA protocol has got limited applications as it is a 2-party protocol.

In proposed protocol trusted third party (key Distribution server) mediates in key distribution. Rather than storing clear text version of password one way hash of the password is stored at the server. Every host and server agree upon family of commutative hash functions using which host authenticates itself to server when it applies for session key . During this protocol run host establishes one time key with server using which server also authenticates to host. This defeats man-in-the middle attacks.Diffie-Hellman protocol serves as basis for this protocol. It is secure against dictionary attacks as we use one time keys with server. It is also secure against malaicious insider attacks (host misuses the information in one protocol run to another)since we use one time keys. It also provides perfect forward secrecy i.e. even if one key is disclosed future session keys will not be disclosed. Moreover we don't use any public key infrastructure which needs large computational power. In this protocol server acts just like a authentication server not like a monitoring server. This protocol is also immune to off-line and on-line guessing attacks as there is no verifiable information is present.

Chapter 1

INTRODUCTION

"A hundred ounces of silver spent for information may save ten thousand spent on war".

By Sun-Tzu[Chinese general fourth century B.C]

Information is a "quality" of a message that is sent from a sender to one or more receivers. Information is the state of a system of interest (curiosity). Message is the information Materialized.Information security deals with several different "trust" aspects of information. Another common term is information assurance. Information security is not confined to computer systems, or to information in an electronic or machine-readable form. It applies to all aspects of safeguarding or protecting information or data, in whatever form. The concepts, techniques, technical measures, and administrative measures used to protect information assets from deliberate or inadvertent unauthorized acquisition, damage, disclosure, manipulation, modification, loss, or use.

Information security is defined as the protection of information systems against unauthorized access to or modification of information, whether in storage, processing or transit, and against the denial of service to authorized users or the provision of service to unauthorized users, including those measures necessary to detect, document, and counter such threats. Because today's economy depends on the secure flow of information within and across an organization, information security is an issue of vital importance. A secure and trusted Environment for stored and shared information greatly enhances consumer benefits, business performance and productivity, and national security. Conversely, an

insecure environment creates the potential for serious damage to governments and corporations that could significantly undermine consumers and citizens.

For firms engaged in critical activities, such as electrical power generation, banking and finance, or healthcare, the stakes are particularly high. Integral to all security programs whether for an asset or an entire agency is a risk assessment process that includes determining the level of sensitivity of information and systems. As the technological revolution has progressed throughout society, an increasing amount of 'valuable' data is stored or transported in the form of electronic binary sequences. This has lead to an extremely efficient and sophisticated environment, with the rate of information exchange reaching incomprehensible limits. Organizations have finally grasped the concept of Information Technology, and now see that by effectively applying it, they are increasing business prosperity. Inevitably, this has lead to greater dependence on IT on the part of the firms.

As we all know, we do not live in a perfect environment, and with access to such informational power via the networks, there is bound to be an increasing amount of hacking. So, with a variety of organizations depending on their computer systems, the cost of this type of crime could be extremely high, both in terms of time and money. The other main disadvantage is the need for extremely expensive hardware leading to an increased number of physical thefts. It is these fundamental problems that have brought about the need for security. It is an important point that information security is, inherently and necessarily, neither hermetic nor watertight nor perfectible. No one can ever eradicate all risk of improper or capricious use of any information.

1.1 ELEMENTS OF INFORMATION SECURITY

Three widely accepted elements (aims, principles, qualities, characteristics, attributes) of information security are:

Confidentiality :

Confidentiality has been defined by the International Organization for Standardization (ISO) as "ensuring that information is accessible only to those authorized to have access" and is one of the cornerstones of Information security. Confidentiality is one of the design goals for many crypto systems, made possible in practice by the techniques of modern

cryptography.

Integrity :

In computer science and telecommunications, the term data integrity has the following meanings:

- The condition in which data are identically maintained during any operation, such as transfer, storage, and retrieval.
- The preservation of data for their intended use.
- Relative to specified operations, the apriori expectation of data quality.

Another aspect of data integrity is the assurance that data can only be accessed and altered by those authorized to do so. Integrity can be compromised in two main ways:

1. Malicious altering
2. Accidental altering

Availability :

Availability has the following meaning:

- The degree to which a system, subsystem, or equipment is operable and in a committable state at the start of a mission, when the mission is called for at an unknown, i.e., a random, time. Simply put, availability is the proportion of time a system is in a functioning condition. The conditions determining operability and commutability must be specified.availability is Expressed mathematically, as one minus the unavailability. A simple way to express this is "the right information to the right people at the right time".

1.2 LAYOUT OF THESIS

As part of this thesis in Chapter-1 need for Information security,Elements of Information security are discussed. In Chapter-2, key distribution,Design goals of key agreement protocols,possible attacks are discussed. In Chapter-3 , traditional 2-Party authentication,2-Party key distribution and 3-party key distribution protocols[7] are discussed. 2-party key

agreement protocols like Diffie-Hellman protocol [10], Encrypted key exchange protocol (EKE)[5] ,Simple Authenticated Key Agreement protocol [1] are discussed in Chapter-4. 3-party Key agreement protocols , STW protocol [13], LSH-3 Party EKE [2] are discussed in Chapter-5. In Chapter-6, new 3-party key agreement protocol which is secure against masquerading and dictionary attacks is proposed. In Chapter-7 conclusion and scope of future work in this area is discussed.

Chapter 2

OVERVIEW OF KEY AGREEMENT PROTOCOLS

The main goal of cryptography is to enable secure communication in a hostile environment. Two parties, P_i and P_j ,want to safely communicate over a network occupied by an active adversary. Usually, P_i and P_j will want to ensure the privacy and authenticity of the data they send to each other. They will encrypt and authenticate their transmissions. But before P_i and P_j can use these tools they will need to have keys. Indeed, without keys, cryptography simply cannot get off the ground. Key agreement is one of the fundamental cryptographic primitive after encryption and digital signature. Such protocols allow two or more parties to exchange information among themselves over an adversarially controlled insecure network and agree upon a common session key, which may be used for later secure communication among the parties. Thus, secure key agreement protocols serve as basic building block for constructing secure, complex, higher-level protocols. Key establishment may be broadly subdivided into key transport and key agreement.

Secret communications with secret keys implies that only trusted parties should have copies of the secret key. Although secret keys can assure us of confidentiality,authentication of users, and message integrity, in a global world we must be able to securely distribute keys at a distance in a timely manner [19]. If security is to be maintained, key distribution must be as solid as the cryptographic method and must be able to ensure that only trusted parties have copies of the keys. Obviously, key distribution is a significant problem. Key establishment protocols involving authentication typically require a set-up phase whereby authentic and possibly secret initial keying material is distributed. Most

protocols have as an objective the creation of distinct keys on each protocol execution. In some cases,the initial keying material pre-defines a fixed key which will result every time the protocol is executed by a given pair or group of users. Systems involving such static keys are insecure under known-key attacks.

Key pre-distribution schemes are key establishment protocols whereby the resulting established keys are completely determined a priori by initial keying material. In contrast, dynamic key establishment schemes are those whereby the key established by a Fixed pair (or group) of users varies on subsequent executions. Dynamic key establishment is also referred to as session key establishment. In this case the session keys are dynamic, and it is usually intended that the protocols are immune to known-key attacks. Many key establishment protocols involve a centralized or trusted party, for either or both initial system setup and on-line actions (i.e., involving real-time participation). This party is referred to by a variety of names depending on the role played, including: trusted third party, trusted server, authentication server, key distribution center (KDC), key translation center (KTC), and certification authority.

It is generally desired that each party in a key establishment protocol be able to determine the true identity of the other(s) which could possibly gain access to the resulting key, implying preclusion of any unauthorized additional parties from deducing the same key. In this case, the technique is said (informally) to provide secure key establishment. This requires both secrecy of the key and identification of those parties with access to it.

2.1 DEFINITIONS

- **Protocol :**
  protocol is a multi-party algorithm, defined by a sequence of steps precisely specifying the actions required of two or more parties in order to achieve a specified objective.

- **Key establishment :**
  Key establishment is a process or protocol where by a shared secret becomes available to two or more parties, for subsequent cryptographic use.

- **Key transport protocol :**

  key transport protocol or mechanism is a key establishment technique where one party creates or otherwise obtains a secret value, and securely transfers it to the other(s).

- **Key agreement protocol :**

  key agreement protocol or mechanism is a key establishment technique in which a shared secret is derived by two (or more) parties as a function of information contributed by, or associated with, each of these, (ideally) such that no party can predetermine the resulting value.

- **Key distribution system :**

  KDS is a method whereby, during an initialization stage, a trusted server generates and distributes secret data values (pieces) to users, such that any pair of users may subsequently compute a shared key unknown to all others (aside from the server).

2.2 MOTIVATION FOR USE OF SESSION KEYS

Key establishment protocols result in shared secrets which are typically called, or used to derive, session keys. Secret keys are not secure forever. They can be stolen, lost, forgotten, destroyed,stored in insecure ways, or copied without authorization [16]. A secret key that has been used many times probably hides more secrets than a secret key that has been used only once. An adversary is more likely to go after the secret key that has been used many times. Ideally, a session key is an ephemeral secret, i.e., one whose use is restricted to a short time period such as a single telecommunications connection (or session), after which all trace of it is eliminated [17].

Motivation for ephemeral keys includes the following:

- To limit available cipher text (under a fixed key) for cryptanalytic attack.
- To limit exposure, with respect to both time period and quantity of data, in the event of (session) key compromise.
- To avoid long-term storage of a large number of distinct secret keys (in the case where one terminal communicates with a large number of others), by creating keys

only when actually required

- To create independence across communication sessions or applications. It is also desirable in practice to avoid the requirement of maintaining state information across sessions.

2.3 DESIGN GOALS

Before turning to the construction of the actual protocols,we emphasize the goals of design and desired properties of the resulting protocols:

- **Simplicity**

  Simplicity is the major theme in the design and its foremost intended feature. The simpler the protocol, the easier it is to spot vulnerabilities and to demonstrate security features.

- **No timetamps**

  Use of timestamps in authentication and key distribution protocols has been debated ad nauseum for a number of years. Since our concern is with simplicity, timestamps are unacceptable because of the inherent requirement for (even loose) clock synchronization [19].

- **Small number of cryptographic operations**

  Cryptography is essential but must be used sparingly. Minimizing the use of cryptography makes protocols simpler and more efficient.

- **Small message sizes**

  Small message sizes can make a protocol suitable for implementation in space-conscious environments, e.g., in a network layer" or in a boot service. Another incentive is to eliminate unnecessary redundancy which can otherwise make a protocol less secure and/or less efficient.

- **Small number of messages**

  Similarly, too many messages make an awkward protocol. Few messages make the protocol simpler to implement, and less prone to timing constraints(i.e., fewer delays).

- **Conventional cryptography**

  The merits of public key cryptography are many and well known. However, it is still quite inefficient. Furthermore its use sometimes presents a problem because of the associated patent issues. Finally, most public key methods impose a fairly large basic encryption block size (e.g., 512 bits is a recommended minimum for RSA.). Nonetheless, for the sake of generality, the resulting protocols must not have any features that rule out the use of public key cryptography.

- **No decryption**

  A typical cryptosystem has two components: encryption and decryption. While the use of encryption is necessary there are reasons to avoid using decryption. First, decryption makes the implementation more complex. Second, it rules out the use of strong one-way hash functions in place of traditional encryption techniques (and where only encryption is needed, strong one-way hash functions can be used instead at much lower cost).

- **Minimal overhead**

  The key distribution protocol we intend to construct should impose minimal additional overhead on the existing authentication protocol it is based on. Since the authentication protocol used as building block is a generic one, we abstract out the specifics of the underlying encryption function.

- **Key Non-Disclosure**

  A third party cannot discover a key being distributed without explicit collaboration of a legitimate party. Legitimate party is one of the two protocol participants.

- **Key Non-Modification**

  a third party cannot modify a key being distributed to any value known to this third party.

- **Key Non-Reuse**

  A third party cannot distribute a previously used key, i.e., it cannot fool the initiating party into using an old key(This can be considered a subset of the Key Non-Modification property).

- **Key Independence**

  Knowledge of one key cannot be used to compute other keys, i.e., a key distributed

in one protocol run does not open the door to discovering keys distributed in other protocol runs.

2.4 ADVERSARIES IN KEY ESTABLISHMENT PROTOCOLS

To clarify the threats protocols may be subject to, and to motivate the need for specific protocol characteristics, one requires (as a minimum) an informal model for key establishment protocols, including an understanding of underlying assumptions. Attention here is restricted to two-party protocols, although the definitions and models may be generalized. Communicating parties or entities in key establishment protocols are formally called principals,and assumed to have unique names. In addition to legitimate parties, the presence of an unauthorized third party is hypothesized, which is given many names under various circumstances, including: adversary, intruder, opponent, enemy, attacker, eavesdropper,and impersonator. When examining the security of protocols, it is assumed that the underlying cryptographic mechanisms used, such as encryption algorithms and digital signatures schemes,are secure. If otherwise, then there is no hope of a secure protocol. An adversary is hypothesized to be not a cryptanalyst attacking the underlying mechanisms directly, but rather one attempting to subvert the protocol objectives by defeating the manner in which such mechanisms are combined, i.e., attacking the protocol itself.

A passive attack involves an adversary who attempts to defeat a cryptographic technique by simply recording data and thereafter analyzing it (e.g., in key establishment, to determine the session key). An active attack involves an adversary who modifies or injects messages. It is typically assumed that protocol messages are transmitted over unprotected (open) networks, modeled by an adversary able to completely control the data therein, with the ability to record, alter, delete, insert, redirect, re-order, and reuse past or current messages,and inject new messages. An active adversary attacks the network. Adversary can start up entirely new instances of players. Adversary may acquire session keys and corrupt players themselves. In the face of such a powerful adversary secure session key distribution is only possible when P_i and P_j have some information advantage over the adversary. To emphasize this, legitimate parties are modeled as receiving

messages exclusively via intervening adversaries (on every communication path, or on some subset of **t** of **n** paths), which have the option of either relaying messages unaltered to the intended recipients, or carrying out (with no noticeable delay) any of the above actions.An adversary may also be assumed capable of engaging unsuspecting authorized parties by initiating new protocol executions.An adversary in a key establishment protocol may pursue many strategies, including attempting to:

- Deduce a session key using information gained by eavesdropping.
- Participate covertly in a protocol initiated by one party with another, and influence it,e.g., by altering messages so as to be able to deduce the key.
- Initiate one or more protocol executions (possibly simultaneously), and combine (interleave) messages from one with another, so as to masquerade as some party or carry out one of the above attacks.
- Without being able to deduce the session key itself, deceive a legitimate party regarding the identity of the party with which it shares a key. A protocol susceptible to such an attack is not resilient.

Chapter 3

TRADITIONAL KEY DISTRIBUTION PROTOCOLS

In this protocol an existing secure two-party authentication protocol [7]is used as a stepping stone for constructing a series of simple and secure key distribution protocols. The protocols are shown to satisfy desired security requirements, using the security properties of the underlying authentication protocol. This protocol is modular and simple.

A, B, P, Q	Full principal names
S	Trusted Third Party
$E_k(X)$	Encryption of plaintext block "X" under key "K"
$MAC_K(X)$	Message Authentication Code
K_{ab}	A and B share Key "K"
N_{ab}	Nonce genrated by A and received by B
$A \Rightarrow B \;\; M$	A sends message "M" to B

Table 3.1: Terminology used

3.1 INITIAL ASSUMPTIONS

Here it is assumed that there exists a secure two-party authentication protocol (2PA P)[7].However, any secure nonce based 2PAP will suffice this purposes. Informally speaking, a 2PAP is considered secure if and only if It is computationally difficult for an intruder to impersonate either party. The difficulty should be equal to the strength of the underlying cryptosystem or a strong one-way function. For example, if DES is used

with the 2PAP in , the computational difficulty of defeating the protocol equals that of breaking DES by brute force,which is generally believed to require on the order of 2^{56} trials. 2PAP is as given below :

$$P \Rightarrow Q \qquad P, N_{pq} \quad (1)$$

$$Q \Rightarrow P \qquad AUTH_{K_{pq}}(N_{pq}, N_{qp}, Q), N_{qp} \quad (2)$$

$$P \Rightarrow Q \qquad ACK_{K_{pq}}(Npq, Nqp, P) \quad (3)$$

$AUTH_{K_{pq}}$ denotes an authentication expression based on the shared key K_{pq} and generated by the responding party (Q in this case). This expression is computed over three inputs, two nonces (one generated by each party) and the name of the message originator.One example of AUTH is: $E(Q \oplus E(N_{qp} \oplus E(N_{pq})))$ Similarly, $ACK_{K_{pq}}$ is the authentication expression that the initiating party sends in order to complete two-way authentication. It is computed over the same inputs except for the message originator's name (which Is P in this case, or can evenbe omitted). An example of ACK is: $E(N_{qp} \oplus E(N_{pq}))$.

In flow-1 "P" genrates a nonce, which is assumed to be a good random number choosed from a uniform distribution and assumed to be different in every protocol run. "P" sends N_{pq}. "Q" genrates its nonce N_{qp}. "Q" computes authentication expression and sends along with N_{qp} to "P" in flow-2. After receving, "P" computes authentication expression and verifies "Q" s authenticity. Since authentication expression is assumed to be secret only legitimate parties can compute it. "P" computes acknowledgement, N_{pp} and N_{qp} as seeds and sends to "Q". "Q" recomputes acknowledgement and authenticates "P". Since N_{pq} and N_{qp} are coming in plain there is no need of decryption. Authentication expression and acknowledgment functions are oneway hash functions. This protocol is compact(requires minimum no.of messages). Stronger one way hash function suffices decryption.

3.2 2-PARTY KEY DISTRIBUTION PROTOCOL (2PKDP)

The model for two-party key distribution [7] is such that one of the parties initiates the protocol by requesting a new key. The other party responds by generating a new key and shipping it back to the requester. The protocol may include a confirmation flow whereby the initiator acknowledges the receipt of the new key. The 2PKDP is as specified below:

$$P \Rightarrow Q \qquad P, N_{pq} \quad (1)$$

$$Q \Rightarrow P \qquad AUTH_{K_{pq}}(N_{pq}, N_{qp}, Q) \oplus K_{new}, N_{qp} \quad (2)$$

It is a simple protocol consisting of only two messages. This protocol is much more similar to 2PAP. "Q" generates K_{new} (session key) sends to "P". After flow-2, "P" extracts the key XOR-ing the re-computed mask expression with the corresponding field in the message. K_{new} is also assumed to be good random number which varies in every protocol run. Subsequently 2PKDP can be used as a stepping stone for obtaining three-party KDPs. However, one should not conclude that two-party key distribution, by itself, has no applications. A 2PKDP can be used, for example, to refresh a short-term session key between two parties (while retaining a more long-term pairwise key). Given a secure 2PAP, it is computationally difficult for an intruder (not knowing K_{pq}) to obtain an AUTH expression when at least one of the nonce N_{pq} or N_{qp} is selected by a legitimate party. The AUTH expression scavenged from the secure 2PAP. Both plaintext and key are random and unpredictable values, and therefore this can be considered a strong encryption function. It resembles a truncated 2PAP except that the authentication expression in the second message is used as a one-time mask for the key being distributed.

3.2.1 Security Analysis

Key Disclosure in 2PKDP can take place only if the attacker is able to obtain an AUTH expression that safeguards a new key. The attacker has two sources of information that can help in "breaking" the protocol. First of all, the attacker may record any number of legitimate executions of 2PKDP between P and Q. In this case,N_{pq} is always under control of "P" and N_{qp} is always under control of "Q". Alternatively, he may try to impersonate "P" by changing or composing a first message of the protocol and intercepting Q's reply;

in which case N_{pq} is under the attacker's control, and N_{qp} is selected by Q. In both cases, at least one of the nonces: N_{pq}, N_{qp}, is always under the control of a legitimate party, i.e.,P or Q. Therefore, the ability to compute $AUTH_{K_{pq}}(N_{pq}, N_{qp}, Q)$ is equivalent to breaking 2PAP.

Key Modification to a value known by the attacker is essentially equivalent to key disclosure. If the attacker is able to modify the key to a selected value then the corresponding AUTH expression simultaneously becomes known. However,since the attacker cannot know the key apriori, he must first know the AUTH expression. This is clearly impossible.

Key Reuse entails the attacker "feeding" an old key to the initiating party. Note that the attacker does not have to know the old key in order to try this attack (the simplest attack is to use pre-recorded replies from previous 2PKDP runs). Since the attacker does not know any old key K^0, the only pieces of knowledge available to him are the recorded messages from previous protocol runs of the form: $AUTH_{K_{pq}}(N^0_{pq}, N^0_{qp}, Q) \oplus K^0, N^0_{qp}$
In order to be fooled into accepting K^0, P has to receive a message of the form:
$AUTH_{K_{pq}}(N_{pq}, N^x_{qp}, Q) \oplus K^0, N^x_{qp}$
where N_{pq} is the fresh nonce generated by P in the current, protoco run and N^x_{qp} is a nonce which is either generated by the attacker or by Q. Then, the following relationship must hold:
$AUTH_{K_{pq}}(N_{pq}, N^x_{qp}, Q) = AUTH_{K_{pq}}(N^0_{pq}, N^0_{qp}, Q)$
Assuming that AUTH is based on a strong one-way function, this condition can hold only if: $N_{pq} = N^0_{pq}$ and $N^x_{qp} = N^0_{qp}$ which is impossible since P is assumed to generate bonafide nonces (i.e., nonces, by definition, are never reused). Alternatively, we can analyze the issue of replay by considering what happens if an attacker re-sends an old protocol message to P, replacing the nonce N^0_{qp} by some value N^x_{qp}: $AUTH_{K_{pq}}(N^0_{pq}, N^0_{qp}, Q) \oplus K^0, N^x_{qp}$ where either $N_{pq} \neq N^0_{pq}$ or $N^x_{qp} \neq N^0_{qp}$ P will extract a key $\bar{K}$ instead of K^0, satisfying following equation:
$AUTH_{K_{pq}}(N_{pq}, N^x_{qp}, Q) \oplus \bar{K} = AUTH_{K_{pq}}(N^0_{pq}, N^0_{qp}, Q) \oplus K^0$

$\bar{K}$ is,not known to the attacker, since he can't compute the AUTH expression masking it. One important consequence of this result is that the attacker can, in effect, "fool"

the protocol initiator P into accepting just about any tuple of the form $< G^X, N^X >$ as valid reply in the second flow of the protocol. However, still the protocol achieves its goal of distributing a random,one-time key which can only be discovered by the legitimate protocol initiator. That is, even though the attacker can convince P to accept any value G^X as an expression masking the key, the protocol retains its strength since the key extracted from G^X remains secret.

Key Independence requires that protocol runs be unrelated. If the attacker discovers K^i from some protocol run (possibly via a brute force attack or some other means external to the protocol) then he simultaneously gains the knowledge of the corresponding AUTH expression,$AUTH_{K_{pq}}(N^i_{pq}, N^i_{qp}, Q)$ that conceals the said key. As there is no relationship between keys distributed in different protocol runs, knowledge of a K_i by itself, offers no advantages to the attacker. Even knowledge of $AUTH_{K_{pq}}(N^i_{pq}, N^i_{qp}, Q)$ is only marginally useful since we assume that the probability of:
$AUTH_{K_{pq}}(N^i_{pq}, N^i_{qp}, Q) = AUTH_{K_{pq}}(N^j_{pq}, N^j_{qp}, Q)$.
is negligible when $\ulcorner N^i_{pq}, N^i_{qp} \urcorner \neq \ulcorner N^j_{pq}, N^j_{qp} \urcorner$ (i, j denote the different protocol runs). Therefore, knowledge of a single session key cannot lead to the discovery of other session keys.

Key Integrity : This protocol does not provides key integrity. Key integrity is not necessarily considered a foremost property of a secure key distribution protocol. Failure to assure key integrity may result in the distribution to the requesting party of a key different from the one originally issued. However, under some circumstances,this is not problematic. The integrity of the key does not matter as long as its value does not become known to an unauthorized party. There are also scenarios where key integrity is needed. So far, we have made an implicit assumption that the new key is chosen uniformly by its issuer. By uniformly we mean that, supposing that a key is an n bits long, then every possible n-bit quantity is equally likely to be selected as a key. On the other hand, if keys are selected in a non-uniform manner whereby each key must satisfy some particular requirements (e.g., the RSA cryptosystem), the uniformness can not be acheived.

3.3 3-PARTY KEY DISTRIBUTION PROTOCOL (3PKDP)

The properties of the 2PKDP discussed so far appear reassuring. However, two party key distribution is not a particularly useful application. A much more common scenario is that of three-party key distribution[7]. The model for three party key distribution is that two parties having no shared secret key enlist the assistance of a mutually trusted third party performs the actual key distribution. This trusted third party is frequently referred to as Authentication Server (AS) or Key Distribution Center (KDC). Each of the two parties are assumed to share a long term key with the AS. As with 2PKDP, the goal is to design a secure 3PKDP. The conditions for a secure 3PKDP are essentially similar to that of 2PKDP. The only additional requirement is that a 3PKDP must be secure against a malicious insider, i.e., a legitimate party that, by participating in legitimate runs of the protocol,can gather enough information to impersonate other parties or otherwise abuse the protocol(e.g., a malicious insider disclosing a key shared with another party).

3.3.1 The Protocol

A naive version of a secure 3PKDP is illustrated here. It is constructed by simply putting together two runs of 2PKDP.

$$A \Rightarrow S \qquad A, B, N_{as} \quad (1)$$

$$S \Rightarrow A \qquad AUTH_{K_{as}}(N_{as}, N_{sa}, B) \oplus K_{ab}, N_{sa} \quad (2)$$

$$B \Rightarrow S \qquad B, A, N_{bs} \quad (3)$$

$$S \Rightarrow B \qquad AUTH_{K_{bs}}(N_{bs}, N_{sb}, A) \oplus K_{ab}, N_{sb} \quad (4)$$

One notable aspect is that the key being distributed in messages 2 and 4 is one and the same-K_{ab}. The names of the parties involved are changed to emphasize the difference with respect to previously discussed two-party protocols. A and B are the two principals and S is the mutually trusted AS. The only other aspect where the present protocol differs from 2PKDP is in the way principal names are used within AUTH tokens. Whereas before, a name denoted the originator of a token , it now refers to the thrid party in the protocol,e.g., the AUTH token sent from S to A includes B's name. Similarly, the AUTH token sent from S to B includes A's name. This feature is necessary to prevent

masquerading attacks whereby a malicious party tampers with the principals' names in message 1 of the protocol. The protocol is secure with respect to outsider attacks,i.e., a non-participating party (i.e., not A, B or S) cannot subvert the protocol. This follows directly from the established security of 2PKDP.

3.3.2 Insider Attacks

The new danger introduced in this 3PKDP as a result of using the same key in messages 2 and 4 are the so called insider attacks by either A or B. Both A and B, being privy to K_{ab} can discover each other's AUTH expressions and try to use this new knowledge in some malicious fashion. Knowing Kab, A (or B) can now alter B's (or A's)key distribution token to any desired value. Whether or not this is a real threat depends on the requirements specific to the local environment. The present protocol certainly fulfills the requirements of nondisclosure, non-modification, non-reuse and independence. As long as the adversary is an outsider. In its current state, the protocol is vulnerable to modification of K_{ab} by an insider (A or B). In other words, neither of the two parties can be sure that theK_{ab} was actually issued by the AS. This exposure cannot be addressed without changing the original 2PKDP. However, not knowing either the key or the masking expression, the attacker can only try to play XOR-ing "games" and factor out K_{ab} by computing:
$AUTH_{K_{as}}(N_{as}, N_{sa}, B) \oplus K_{ab} \oplus AUTH_{K_{bs}}(N_{bs}, N_{sb}, A) \oplus K_{ab}$
$= AUTH_{K_{as}}(N_{as}, N_{sa}, B) \oplus AUTH_{K_{bs}}(N_{bs}, N_{sb}, A)$
This expression cannot be of any value since its components remain unknown. Since the AUTH expression is computed using K_{as} the only way "X" (another insider) could try to misuse this information is in an attempt to modify or find a key distributed to A in a 3PKDP execution between A and some other party, say B. For such an attempt to succeed, X needs to compute $AUTH_{K_{as}}(N_{as}, N_{sa}, B)$ for a random value of N_{as} or N_{sa}. Since X only knows expressions of the form $AUTH_{K_{ax}}(..., ..., X)$ and finding an expression of the form $AUTH_{K_{as}}(..., ..., B)$ is equivalent to breaking 2PAP.

Chapter 4

2-PARTY KEY AGREEMENT PROTOCOLS

2-party key agreement protocols generally function by sharing a predistributed secret between both parties for authentication.

4.1 DIFFIE-HELLMAN PROTOCOL

The Diffie-Hellman key agreement protocol [10] (also called exponential key agreement) was developed by Diffie and Hellman in 1976 and published in the ground-breaking paper New Directions in Cryptography. The protocol allows two users to exchange a secret key over an insecure medium without any prior secrets. A number of commercial products employ this key exchange technique. The algorithm itself is limited to the exchange of keys. The Diffie-Hellman algorithm [10] depends for its effectiveness on the difficulty of computing discrete logarithms. This Protocol is based on Group theory.

- **Group :**

  In abstract algebra, a group is a set with a binary operation that satisfies certain axioms. For example, the set of integers with addition is a group [20]. Many of the structures investigated in mathematics turn out to be groups. These include familiar number systems, such as the integers, the rational numbers, the real numbers, and the complex numbers under addition, as well as the non-zero rationals, reals, and complex numbers, under multiplication. A group G,sometimes denoted by {G,•},is a set of elements with a binary operation •,that associates to each ordered pair (a,b) in G,such that following axioms are obeyed.

(A1) Closure : If a and b belong to G,then a •b is also in G.

(A2) Associative : a •(b •c) =(a •b) •c for all a,b,c in G.

(A3) Identity Element : There is an element "e" in G such that

a •e=e •a=a for all a in G.

(A4) Inverse Element : For each "a" in G there is an element a 'in G such that a •a '=a '•a=e

- **Finite Group :**

  If a group has a finite number of elements,it is referred to as **Finite Group** ,and the **Order** of the group is equal to the number of elments in the group. Otherwise the group is infinite group.

- **Abelian Group :**

  A group is said to **Abelian** if it satisfies all the properties of group and the fallowing additional condition:

  (A5) Commutative : a •b=b •a for all a,b in G.

  The set of integers(positive,negative and zero) under addition is an abelian group.

- **Cyclic Group :**

  A group is **Cyclic** if every element of G is power a^k(k is an integer) of a fixed element $a \in G$. The element "a" is said to generate the group G, or to be a **generator** of G. A cyclic group is always abelian,and may be finite or infinite. Exponentiation with in a group as repeated application of the group operator,so that a^3=a •a •a. Furthur, a^0=e,the identity element;and $a^{-n}=(a^{'})^n$. The additive group of positive integers is an infinite cyclic group generated by the element 1.

- **Rings :**

  A ring R,sometimes denoted by {R,+,X}, is a set of elements with two binary operations,called addition and multiplication, such that for all a,b,c in the R following axioms are obeyed:

  (A1-A5) : R is an abelian group with respect to addition;that is,R satisfies axioms A1 to A5.For this case of an additive group we denote the identity element as 0 and the inverse of a as -a.

  (M1) Closure under multiplication : If a and b belong to R then "ab" is also in R.

  (M2) Associativity of Multiplication : a(bc)=(ab)c for all a,b,c in R

(M3) Distributive Laws: a(b+c)=ab+ac ,(a+b)c=ac+bc for all a,b, c in R

with respect to addition and multiplication ,the set of all n-Square matrices over the real numbers is a ring R.

A ring is said to be **commutative** if it satisfies the fallowing additional condition:

(M4) Commutativity of multiplication : ab=ba for all a,b in R.

Let S be the set of even integers(positive,negative and 0) under the usual operations of addition and multiplication. S is a commutative ring.

An **Integral domain** ,which is a commutative ring that obeys following axioms:

(M5) Multiplicative identity : There is an element 1 in R such that a*1=1*a=a for all a in R.

(M6) No Zero Divisors : If a,b in R and ab=0,then either a=0 or b=0.

Let S be the set of integers,positive,negative,and 0 ,under usual operations of addition and multiplication. S is an integral domain.

- **Fields :**

  A field F, sometimes denoted by {F,+,x} ,is a set of elements with two binary operations,called addition and multiplication,such that for all a,b,c in F the following axioms are obeyed:

  (A1-M6) F is an integral domain;that is, F satisfies A1 to M6.

  (M7) Multiplicative inverse: For each a in F,except 0,there is an element a^{-1} in F such that a(a^{-1})=(a^{-1})a=1.

  In essence, a field is a set in which we can do addition, subtraction,multiplication,and division without leaving the set. Division is defined with following rule:

  a/b=a(b^{-1}). Familiar examples of fields are the rational numbers,the real numbers,and the complex numbers. Set of all integers is not a field,because not every element of the set has a multiplicative inverse. Infact only 1 and -1 have the multiplicative inverse in the integers.

- **Galois Field :**

  Infinite fields are not of particular interest in the context of cryptography. However finite fields play vital role in many cryptographic algorithms. Galois field is a finite field. The finite field of order p^n,where p is a prime and n is a positive integer is generally written as GF(p^n);GF stands for Galois field,in the honour of the mathematician who first studied finite fileds. For every prime number p and

integer $n \geq 1$, there exists a finite field with p^n elements. The simplest case is when the order of the field is prime, i.e., n = 1. This finite field, GF(p). It is a finite field with p elements, usually labelled 0, 1, 2, ... p-1, where arithmetic is performed modulo p. It is also sometimes denoted by Z_p. The simplest finite field is GF(2). Addition in GF(2) is equivalent to the Exclusive-OR(XOR) operation,and multiplication is equivalent to the logical AND operation.

A primitive root of a prime P is an integer g such that g(mod P) has modulo order P-1 . More generally, if GCD(g,n)=1 (g and n are relatively prime) and g is of modulo order $\phi(n)$ modulo n. Where $\phi(n)$ is the totient function, then g is a primitive root of n . The first definition is a special case of the second since $\phi(n)$=P-1 for P a prime.

for example,P=7. We have to choose a number which is relatively prime to P(i.e. gcd(P,x)=1).Taking x=3 ,genrate allthe powers of "x" mod P.

$(3^1, 3^2, 3^3, 3^4, 3^5, 3^6)\text{mod}(7)\equiv$(3,2,6,4,5,1) .

powers of 3 generate all the integers from 1 to 6(i.e P-1) hence 3 is a primitive root of 7.Taking x=5

$(5^1, 5^2, 5^3, 5^4, 5^5, 5^6)\text{mod}(7)\equiv$(5,4,6,2,3,1) .

powers of 5 generate all the integers from 1 to 6(i.e P-1) hence 5 isalso a primitive root of 7.Taking x=2

$(2^1, 2^2, 2^3, 2^4, 2^5, 2^6)\text{mod}(7)\equiv$(2,4,1,2,4,1) .

since 2 does not genrates all the integers from 1 to 6(i.e P-1) it is not a primitive root.

Algorithm for computing Primitive root:

Input : A cyclic group G of order n, and the prime factorization

n= $p_1^{e_1}.\ p_2^{e_2}.\ p_3^{e_3}.....p_k^{e_k}$.

OUTPUT: a generator "g" of G.

1. Choose a random element g of G.
2. Choose i from 1 to k.
   2.1 compute $b \leftarrow g^{n/p_i}$
   2.2 if b = 1 then go to step 1.
3. Return (g).

4.1.1 The Protocol

Two parties who wish to establish a key agree upon "global public elements" ,"P" a large prime no and "g" a generator over a finite (Galois) field. Let G be a cyclic group generated by g(primitive root of P) which is of order P. Then every element of G can be expressed as g^n, $n\epsilon[1...p-1]$. That is if g is a primitive root , powers of g generate all the integers from 1 to p-1. g mod(P) ,$g^2 mod(P)$,$g^3 mod(P)$ .....$g^{p-1} mod(P)$ are distinct and consist of the integers from 1 through P-1 in some permutation we can find a unique exponent "i" such that $b \equiv g^i (mod\ P)$ where $0 \leq i \leq (P-1)$.

The exponent "i" is refered to as the discrete logarithm, or index, of b for the base g, mod P.

Step-1 :

$A \Rightarrow B$ $Y_A = g^{X_A} mod\, P.$

User A selects a secret X_A (private) and $X_A \leq P-1$.

User A computes $Y_A = g^{X_A} (mod\ P)$.

A sends Y_A TO B.

Step-2 :

$B \Rightarrow A$ $Y_B = g^{X_B} mod\, P.$

User B selects a secret X_B (private) and $X_B \leq P-1$.

User B computes $Y_B = g^{X_B} (mod\ P)$.

B sends Y_B TO A.

B computes session key as K=$(Y_A)^{X_B} (mod\ P)$

Step-3 :

A computes session key as K=$(Y_B)^{X_A} (mod\ P)$

These two calculations of "K" produce identical results:

$K = Y_B^{X_A} (mod\ P)$

$= (g^{X_B} mod\, P)^{XA} (mod\ P)$

$= (g^{X_B})^{XA} (mod\ P)$

$= (g^{X_A})^{XB} (mod\ P)$

$= (g^{X_A} mod\, P)^{XB} (mod\ P)$

$= Y_A^{X_B} (mod\ P)$

The result is that the two sides have exchanged a secret key. Furthurmore, because X_A and X_B are private,an opponent only has following ingredients to work with : P,g,Y_A and Y_B. Thus the opponent is forced to take discrete logarithm to determine the key. The security of Diffie-Hellman key exchange [10] lies in the fact that,while it is relatively easy to caliculate exponentials(by repeatedly taking squares of a number) it is very difficult to calculate discrete logarithms. For larger primes,the latter task is considered infeasible.

4.1.2 Attacks on Diffie-Hellman

Attacks against the Diffie-Hellman protocol [10] come in a few flavors. The plausibility of these attacks depends on what assumptions we make about the adversary.

- **Man in the Middle Attacks :**

  An active attacker (Oscar), capable of removing and adding messages, can easily break the core DH protocol presented above. By intercepting g^x and g^y and replacing them with g^{x^1} and g^{y^1} respectively, Oscar (O) can fool A and B into thinking that they share a secret key. In fact, A will think that the secret key is $g^{x(y^1)}$ and B will believe that it is $g^{(x^1)y}$. This is a man in the middle attack.As an example of what can be done with such an attack, consider the case where A and B use a shared secret key obtained in a DH protocol for symmetric encryption. Suppose A sends a message "m" to B and that $ENC_K(m)$ represents the symmetric encryption (e.g. DES) of "m" using the secret key K.

  1. "A" sends $ENC_{g^{x(y^1)}}(m)$. to "B"
  2. "O" intercepts $ENC_{g^{x(y^1)}}(m)$ and decrypts it (which he can do since he knows $g^{x(y^1)}$).
  3. "O" replaces this message with $ENC_{g^{(x^1)y}}(m^1)$. which he sends to B. Note that m^1 can be set to any message.

  The encryption scheme is thus clearly compromised as message privacy is violated. In the next section, we study attacks that can be mounted by a less powerful adversary.

- **Degenerate Message Attacks :**

  There are degenerate cases in which the protocol does not work (i.e. it can be

broken). For example when g^x or g^y equals one, the shared secret key becomes one. Since the communication channel is public anybody can detect this anomaly. Fortunately, this situation is impossible in a properly carried out protocol run because both x and y are chosen from {1, . . . , p - 2} . However, an insider attack is possible and so DH protocol participants should make sure that their key agreement peer does not send $g^x = 1$.

- **Simple Exponents :**

  If one of x and y can be easily determined, the protocol can be broken. For example, if x equals 1 then g^x = g which any observant attacker will be able to detect. It is very hard to determine where to draw the line here, that is, determining for which values of g^i, "i" is hard to determine, since this depends entirely on the strategy of the attacker. Any set of "i" values could be vulnerable, depending on which values of g^i are precomputed, where the search starts, and how it proceeds. In any case, it seems very reasonable to insist that x and y not equal 1.

- **Simple Substitution Attacks :**

  The following attack is very interesting, as it is extremely easy to mount and normally would not come up in theoretical proofs of security. The attacker can force the secret key to be an impossible value. If the DH protocol would only be executed by sentient beings this would not be interesting as the anomalies would be easily detected. However in practice DH protocols are carried out by computers and careless implementations might not spot the following attack.

  1. O intercepts g^x and g^y and replaces them with 1.
  2. Both A and B compute the same shared secret key which equals one.

  So it is safe practice to always verify that g^x and g^y are positive integers smaller than p - 1 and greater than 1.

- **Identity Mis-binding Attacks :**

  Diffie-Hellman protocol [10] does not provides authenticity. In this model any entity can pretend like any other entity and can establish a secret a key. For example an entity A^1 may pretend like A and send a request to B. B do not have any information to verify that whether requested party is Genuine or not. If B establishes a secret key with A^1 ,he will reveal all the secrets to him.

- **Subgroup Confinement Attack :**

  The generator g in the Diffie-Hellman protocol [10] is a primitive root of the prime p, i.e. the order of the group generated by g is equal to p-1. If the selected prime p is such that p-1 has several small prime factors, then some values between 1 and p-1 do not generate groups of order p-1,but of subgroups of smaller orders. Hence, within the group of order p - 1 there are subgroups of smaller orders. If the public parameter of either A or B lies within one of these small subgroups, then the shared secret key would be confined to that subgroup. If the order of the subgroup is small enough, the intruder may launch a brute force attack to determine the exact value of the shared secret key.

  Example :

  Let p = 19 and $g = 2$.

  Then the group generated by g is

  (2, 4, 8, 16, 13, 7, 14, 9, 18, 17, 15, 11, 3, 6, 12, 5, 10, 1)

  Now, Let k = 2(secret key of first party), A = $2^2 = 4$

  Subgroup generated by A = S_A = (4, 16, 7, 9, 17, 11, 6, 5, 1)

  Let l = 3(secret key of second party), B = $2^3 = 8$

  Sub-group generated by B = S_B = (8, 7, 18, 11, 12, 1)

  $K_{ab} = 26(mod\,19) = 7$

  It can be clearly seen from the example above that the shared secret key K_{ab} lies in the intersection of the subgroups generated by k and l. The Solution to counter this kind of an attack is to choose a Safe Prime. Safe primes are prime numbers of the form $p = 2q + 1$ where q is prime. Such primes have various cryptographic advantages.

4.1.3 Results And Discussion

SOURCE PROGRAM

DIFFIE -HELLMAN PARAMETER SET:

P:

119940246058424581467543369193520986783355240855921263255870336586969827366219870923411129002678668583127831556541616810090991878861881894137317819653787 83

g:

675056857616279441879613198962387746584230538955484228729839182062472425430740773279202778206203590181251617980664074463164554780377228365002547148448497 7

****

G[xa]modP:

278793203010932988933657623969293858893394419817713224043639234360621315986240514818687607170113612002790965470407269180728681196015517535321284931679850 6

G[xa]mod P is forwarded to destination

****

Received G[xb] mod P from destn.

G[xb] mod P:

112246321678327609948354317011535317327278656937169714746441258386688031075863444155224926094940898845353388914127691642944035081781561468646078212250446113

****

Final Key agreed Upon:G[x,y]:

665273203027529716793293276791247147787728922226977987772185820379995675635809620728799913664919589436596523938772074250872881327899498722056862076349531 1

DESTINATION PROGRAM

DIFFIE -HELLMAN PARAMETER SET:

P:

11994024605842458146754336919352098678335524085592126325587033658696982736621

198709234111290026786685831278315565416168100909918788618818941373178196537 8

783

g:

6750568576162794418796131989623877465842305389554842287298391820624724254307 4

0773279202778206203590181251617980664074463164554780377228365002547148448497 7

****

Received G[xa] mod P from source.

G[xa] mod P:

2787932030109329889336576239692938588933944198177132240436392343606213159862 4

0514818687607170113612002790965470407269180728681196015517535321284931679850 6

****

G[xb] modP:

11224632167832760994835431701153531732727865693716971474644125838668 8

0310758634441552249260949408988453533889141276916429440350817815614 68

64607212250446113

G[xb]mod P is forwarded to source

****

Final Key agreed Upon:G[x,y]

665273203027529716793293276791247147787728922226977987772185820379995 67

563580962072879991366491958943659652393877207425087288132789949872205 68

620763495311

In this protocol one can easily observe from results that neither of the two parties can alone decide the key completely. Both parties collaborate and constitute session key without revealing their secret information. This protocol lacks entity authentication.

4.2 ENCRYPTED KEY EXCHANGE PROTOCOL

This protocol allows two parties sharing a password to establish a secret key. The Encrypted Key Exchange(hereafter referred to as simply EKE)[5] presents a novel and elegant method of key establishment.

4.2.1 Generic EKE

The 'generic' version of EKE is illustrated below:

$A \Rightarrow B$	$A, P(E_a)$	(1)
$B \Rightarrow A$	$P(E_a(K))$	(2)
$A \Rightarrow B$	$K(C_a)$	(3)
$B \Rightarrow A$	$K(C_a, C_b)$	(4)
$A \Rightarrow B$	$K(C_b)$	(5)

The protocol begins with A generating a random key-pair (E_a, D_a) of some public key encryption scheme. Then, A sends to B the encryption of E_a under the password P (a weak shared secret). B generates a new session key, K, encrypts it with E_a, super-encrypts the result with P, and forwards it back to A. The remainder of the protocol - flows 3, 4, and 5 - represent standard hand-shaking that follows key distribution. A generates a challenge C_a and encrypts it with K and send to B. B decrypts it and generates its own challenge C_b . B encrypts and sends back both C_a and C_b to A. A decrypts it and verifies received value of C_a with sent value and confirms that B posses same key as A. A encrypts C_b and sends back to B. B decrypts it and verifies received value of C_b with sent value and confirms that A posses same key as B.

Attacks on Generic EKE:

The generic EKE [5]protocol is susceptible to the type of attack that, for lack of better term, we shall call Denning-Sacco Attack [11] or DS for short. The attack proceeds as follows:

The attacker manages to obtain one of the session keys used in one run of a key distribution protocol. Armed with that knowledge, the attacker is then able to impersonate one of the parties indefinitely often.

The attacker somehow obtains one of the session keys distributed in one (recorded) run of EKE. Armed with that knowledge, the attacker mounts a dictionary attack on the

password and, upon breaking the password, is able to impersonate one of the parties indefinitely.
In more detail, the DS attack is as follows:

- The attacker records one run of generic EKE and somehow obtains the key K.
- Iterating upon all possible choices of "P" :
  1. Pick a candidate $\overline{P}$
  2. Compute$\overline{E_a} = \overline{P}^{-1}(P(E_a))where P(E_a)$ is taken from flow 1 of the recorded run.
  3. Compute $\overline{E_a}(K)$ (only if $\overline{E_a}$ is a valid key)
  4. Compute $P(\overline{E_a}(K))$ and compare it to $P(E_a(K))$ from recorded flow 2.

  A match in the last step indicates correct guess of the password and earns the attacker *carte blanche* with respect to impersonating A.

4.2.2 EKE with Diffie-Hellman key exchange

The Exponential Key Exchange (EKE) variant (referred to as EKE-DH from here on) is illustrated below. EKE-DH appears to be the most practical EKE variant because of the relative simplicity of the Diffie-Hellman key exchange[10]. Here it is assumed that both A and B agree upon Diffie-Hellman parameters, g(genrator of cyclic group) and p(large prime no.)

$A \Rightarrow B$	$A, P(R_a)$	(1)
$B \Rightarrow A$	$P(R_b), K(C_b)$	(2)
$A \Rightarrow B$	$K(C_a, C_b)$	(3)
$B \Rightarrow A$	$K(C_a)$	(4)

In step-1, A picks a random number "ra" and calculates $R_a = g^{ra}(mod p)$. Note that name is sent in the clear. R_a is encrypted with shared password(P) between A and B. A,P(R_a) is sent to B

In step-2 B picks a random number "rb" and calculates $R_b = g^{rb}(mod\, p)$. B also uses the shared password P to decrypt P[R_a] and calculates $g^{(ra)(rb)}(mod\, p)$ like in Diffie-Hellman protocol[10]. The session key K is derived from this value, perhaps by selecting certain bits. Finally, a random challenge C_b is generated.C_b is encrypted with K. P(R_b),K(C_b) is sent to A.

In step-3 A uses P to decrypt P[R_b] and calculates $g^{(rb)(ra)}(mod\, p)$. From this, K is calculated; it in turn is used to decrypt K [Cb]. A then generates her own random challenge C_a. Both C_a and C_b are encrypted with K and sent to B.

In step-4 B decrypts and verifies that C_b is proper. B encrypts C_a with K and sends to A. A verifies genuineness of C_a.

4.2.3 Results And Discussion

SOURCE PROGRAM

Diffie-Hellman Parameter set:

P:

109579776835328642071509181023663069353658528879893587273529656697 3

243831286957344450457139556768953702304263640847683010091300628516 3

226374528497081783723

g:

716531661610066108061824510714311075432644242743931596309980267357 1

068234800249357351983247378825844688304154648543439356945565757647 3

06712866661624492784

****

g.[ra]mod P

448990968423904912541156063339395799757226648544441982142063964061 33

228892651412927072339589062666276924341550326979951415909634488039 37

850164961594530685

****

AFTER ENCRYPTION:

448990968423904912541156063339395799757226648544441982142063964061332

288452730278398725869178187504249881099953604967056148359247954306254
0323906262386719

g[ra]mod P is forwarded to destination

****

Received g[rb] mod P from destn.

Received challenge-B:

644784991818454152722819191298327885028844551312946016155120225718673
120274098256439110844911931459122497950191945026571093610511107061816
086198882386049

Encrypted g[rb] mod P:

698875332881071785105248881631044812329816206093439758836925026495866
033265858310260265602550706012623805381407585844656283927476039982692
6078895563696082

****

Decrypted g[rb] mod P:

698875332881071785105248881631044812329816206093439758836925026495866
032221556510929853625617989707514912610569600843191903000096853728706
8031381424829616

****

Key value:

644784991818454152722819191298327885028844551312946016155120225718673
120274098256439110844911931459122497950191945026571093610511107061816
086198922769717

****

Decrypted challenge-B : 45626804

challenge- A: 1174

****

K[Ca]:

644784991818454152722819191298327885028844551312946016155120225718673120
274098256439110844911931459122497950191945026571093610511107061816086
98922768803

K[Cb]:

644784991818454152722819191298327885028844551312946016155120225718673120

027409825643911084491193145912249795019194502657109361051110706181608619888238604 9

k[ca,cb] is forwarded to Destination

****

Received K[Ca] from destn.

Received K[Ca] :

644784991818454152722819191298327885028844551312946016155120225718673120274098256439110844911931459122497950191945026571093610511107061816086198922768803

****

Received [Ca] after decryption: 1174

RECEIVED CHALLENGE-A IS PROPER

DESTINATION PROGRAM

Diffie-Hellman Parameter set:

P:

109579776835328642071509181023663069353658528879893587273529656697324383128695734445045713955676895370230426364084768301009130062851632263745284970817837 23

g:

716531661610066108061824510714311075432644242743931596309980267357106823480024935735198324737882584468830415464854343935694556575764730671286666162449278 4

****

Received g.[ra] mod P from source

Received Value g.[ra] mod P From Source[Encrypted]:

448990968423904912541156063339395799757226648544441982142063964061332288452730278398725869178187504249881099953604967056148359247954306254032390626238671 9

G[xa] mod P[Decrypted]:

448990968423904912541156063339395799757226648544441982142063964061332288926514129270723395890626662769243415503269799514159096344880393785016496

1594530685

****

g[rb] mod P[Before Encryption] :

6988753328810717851052488816310448123298162060934397588369250264958660322
2155651092985362561798970751491261056960084319190300009685372870680313814
24829616

****

Key value :

6447849918184541527228191912983278850288445513129460161551202257186731202
7409825643911084491193145912249795019194502657109361051110706181608619892
2769717

****

challenge B: 45626804

Challenge B[encrypted] :

6447849918184541527228191912983278850288445513129460161551202257186731202
7409825643911084491193145912249795019194502657109361051110706181608619888
2386049

g[xb] mod P[After Encryption] :

6988753328810717851052488816310448123298162060934397588369250264958660332
6585831026026560255070601262380538140758584465628392747603998269260788955 6
3696082

g[xb]mod P,K[Cb] is forwarded to source

****

Received K[Ca,Cb]

K[Ca]:

6447849918184541527228191912983278850288445513129460161551202257186731202
7409825643911084491193145912249795019194502657109361051110706181608619888
922768803

K[Cb]:

6447849918184541527228191912983278850288445513129460161551202257186731202
7409825643911084491193145912249795019194502657109361051110706181608619888
2386049

****

Decrypted [Cb]:45626804

RECEIVED CHALLENGE-B IS PROPER

Decrypted [Ca]:1174

****

Encrypted Ca:K[Ca]:

64478499181845415272281919129832788502884455131294601615512022571867312027409825643911084491193145912249795019194502657109361051110706181608619892276880 3

K[Ca] is forwarded to source

****

This protocol provides entity authentication. Both parties encrypt their data using shared password. This protocol also provides key confirmation. Sharing a secret between every two parties is almost impossible in large environment. Unlike generic EKE, DH-EKE does not suffers from dictionary attacks.

4.3 SAKA PROTOCOL

Simple Autheticated Key Agreement Protocol [1] called SAKA is simple and cost effective. SAKA has less number of steps and less computation cost. Password based mechanism is used for user autentication. This is a 2-party key agreement protocol. This protocol is based on Diffie-Hellman key agreement[10],Easy generalization.

4.3.1 The Protocol

A and B(system pricipals) are assumed to share the weak secret(password) pw in a secure way. They agree upon the generator g and its group Z_p^*. x and y are selected in Z_p^* for a uniform distribution, and $X = g^x(mod\ P)$ and $Y = g^y(mod\ P)$ are also in Z_p^* for a uniform distribution. The session key is made by$h(g^{xy} mod\ P)$. The protocol run as follows.

1.$A \Rightarrow B \qquad X \oplus pw$

A chooses a random number x, computes $X = g^x(mod\ P)$, and encrypts it with pw send to B. After receiving message 1, B recovers X by using the password pw. Then, B chooses a random number y, computes $Y = g^y(mod\ P)$ and $Key_2 = X^y(mod\ P) = g^{xy}(mod\ P)$

like in Diffie-Hellma protocol[10]. B encrypts Y with pw. B also computes one way hash $h(\ )$ using X,Key_2 as parameters. B sends $Y \oplus pw||h(Key_2, X)$ to A.

2.$B \Rightarrow A$ $\quad Y \oplus pw||h(Key_2, X)$

After receiving message- 2, A recovers Y by using the password pw and computes $Key_1 = Y^x (mod\ P) = g^{xy} (mod\ P)$. A also computes one way hash $h(\ )$ using X,Key_1 as parameters and verifies that $h(Key_1, X) = h(Key_2, X)$. If they match each other, A confirms that Key_2 is valid and both parties posses same key. It also suggests that X is not tampered in transit and Y is from valid source(B) i.e. it authenticates B. Then, A computes the response data $h(Key_1, Y)$ and sends it to Bob.

3.$A \Rightarrow B$ $\quad h(Key_1, Y)$

B computes $h(Key_2, Y)$ and verifies $h(Key_1, Y) = h(Key_2, Y)$. If they match each other, B confirms that Key_1 is valid and both parties posses same key. It also suggests that Y is not tampered in transit and X is from valid source(A) i.e. it authenticates A.

Finally, A and B agree on the common session key $K = h(Key_1) = h(Key_2) = h(g^{xy} mod\ p)$.

4.3.2 Security Analysis

SAKA protocol [1] is based on computational Diffie-Hellman (CDH) [10] problem, which states that computing $g^{xy} (mod\ P)$ giving $g^x (mod\ P)$ and $g^y (mod\ P)$ is hard. SAKA [1] also satisfies completeness property i.e. if each party's messages are faithfully relayed to one another, then the parties succeed in authentication and key agreement, at least with overwhelming probability. Adversaries cannot not be accepted by the principals without knowing the password. Off-line password guessing attack succeeds when there are pieces of information in communications which can be used to verify the correctness of the guessed passwords. In the SAKA protocol, a passive attacker , all he receives from the protocol is as follows: $X \oplus pw$,$Y \oplus pw$,$h(Key_1, Y)$,$h(Key_2, X)$. He first guesses a password pw^1 and finds $g^{x^1} = X \oplus pw \oplus pw^1$and $g^{y^1} = X \oplus pw \oplus pw^1$. If he wants to verify his guess, he has to find Key_1 or Key_2 which is impossible.

Since x and y are selected in the cyclic group for a uniform distribution, we can see that X and Y remain on the cyclic group under uniform distribution, and $X \oplus pw$ and $Y \oplus pw$ also remain on the cyclic group under uniform distribution. There is no way to find the relationship between the rejected password and the remaining password.on-line

trial on password cannot partition out the possible set. The partitioning implies that the possible set decreases logarithmically. If an adversary tries to masquerade B and defraud A, she can know $g^x \oplus pw$ sent from A and y, g^y,$g^y \oplus pw^1$ by herself where pw^1 is a guessed password. It is helpless since there is not any verifiable data. That means, he can not carry out the off-line guessing attacks. To continue, he has to reply $h(Key_2, X)$ data. Which is not possible since he cannot find out Key_2.

4.3.3 Results And Discussion

SOURCE PROGRAM

P :

10163730290118433019669854391759641550877131600343545635428272118766324512
92160915193928533212202860148651474922288894298730439908593907300992684837
9534113

g :

79107858144553357499004354010774603642381692475568793029591610234869803517
01132320936286130733271761100015930547819036820304088822415812780082043052
443176

Pw:

98439849821793940494843739830

****

G(xa)modP :

52335914079605760131781045401304740492987987578769754811105561739728654544059
10158847411022613636628665523734836433474265955186439441014919747711606832317

****

AFTERENCRYPTION:

52335914079605760131781045401304740492987987578769754811105561739728654591427
22822414672641837294947578583333489793234954450831656745438703033320515645407

G[xa]mod P is forwarded to destination

****

Received Hash Code : 380258010

Received G(xb) mod P :

89878914849557570277047852664036400385943662322031549774967736017094093807865

139146485297999160722609840266625670277865831446753369500505820321201832177 14

Decrypted G(xb) mod P :

898789148495575702770478526640364003859436623220315497749677360170940936844 54

844628006839852239230676365176086325132289719249336007274109980481881280218 40

****

Received hash value from DestinationH(GXY,GX) :380258010

Computed Hash value:H(GXY,GX) 380258010

Received Hash value is proper.

****

HASH:H(GXY,GY) : 352721330

H(GXY,GY) is forwarded to Destination

****

Final Key agreed Upon:H(G(x,y)) 1592155219

DESTINATION PROGRAM

DIFFIE-HELLMAN PARAMETER SET :

P :

101637302901184330196698543917596415508771316003435456354282721187663245 12

921609151939285332122028601486514749222888942987304399085939073009926848 37

9534113

g :

791078581445533574990043540107746036423816924755687930295916102348698035 17

011323209362861307332717611000159305478190368203040888224158127800820430 52

443176

Pw:

98439849821793940494843739830

****

Received Value G(xa) mod P From Source

523359140796057601317810454013047404929879875787697548111055617397286545 91

427228224146726418372949475785833334897932349544508316567454387030333205 15

645407

G(xa) mod P(DECRYPTED) :

523359140796057601317810454013047404929879875787697548111055617397286545440591015884741102261363662866552373483643347426595518643944101491974771160683231 7

****

G(xb) mod P(Before Encryption) :

898789148495575702770478526640364003859436623220315497749677360170940936844548446280068398522392306763651760863251322897192493360072741099804818812802184 0

****

NEW HASHH(GXY,GX) : 380258010

G(xb) mod P(After Encryption) :

898789148495575702770478526640364003859436623220315497749677360170940938078651391464852979991607226098402666256702778658314467533695005058203212018321771 4

G(xb)mod P is forwarded to source

****

COMPUTED HASH:H(GXY,GY) 352721330

Received HashH(Gxy,Gy) Value : 352721330

Final Key agreed Upon:H(G(x,y)) 1592155219

SAKA provides entity authentication by shared passwords. This protocol is optimal 2-party key agreement protocol as it takes minimum no.of steps and random nubers. Both the parties encrypt the data using shared secret. It is secure against dictionary attacks as there is no verifiable information present. One can observe from results that one way hash is not at all useful for cryptanalysis purpose as it is irrversable.

Chapter 5

3-PARTY KEY AGREEMENT PROTOCOLS

5.1 STW PROTOCOL

This protocol was proposed by Steiner, Tsudik and Waidners [13]. Password-based mechanism is the widely used method for authentication since it allows people to choose their own passwords without any assistant device to generate or store. However,people are used to choose easy-to-remember passwords such that guessing attacks could succeed. This is a 3-party key agreement protocol. All parties (clients) share their secrets with a trusted server only. This protocol is more suitable for large communication environments. From the form of passwords stored in the second party (B), there are two types of protocols,plaintext-equivalent protocols in which the clear form of the first party's (A) password is stored in B, and verifier-based protocols in which the verifier that is easily computed from the password, yet deriving the password from the verifier is computationally infeasible, is stored in B. The verifier-based protocol has the advantage that a compromised verifier does not reveal the password directly. However, a compromised verifier of a weak password also suffers from the guessing attack. Additionally, the verifier-based protocol has more computational overheads than the plaintext-equivalent protocol. Thus, a secure plaintext-equivalent protocol is suitable and necessary if we can not confine people to choose and remember strong passwords.

From the session key creation point, such protocols can be classified into two types: key transport protocols in which the session key is created by one party and securely

transmitted to the other party, and key agreement protocols in which both parties contribute information for creating the resultant session key. The latter is fairer and more secure than the former since in the latter no one can fully control the session key, while the former is suitable for some special environments. In key transport type of three-party protocols,the session key is created by the server S instead of one of the two communication parties. This will result in the worry that a malicious server can get all transaction contents. In most applications we only need the server to be an authentication server but not to be a monitor center (except some special needs, e.g. national defense). In key agreement type of three-party protocols, the session key contributed from A, B and S needs more computational cost than it contributed from A and B, moreover, it is still unknown to the server S.

Steiner, Tsudik, and Waidner proposed a three-party EKE protocol (hereafter referred to as STW 3-Party EKE) [13] and declared that it performed the following tasks:

- Secure distribution of session key K to A and B.
- Mutual authentication of A and B.
- (Indirect) authentication of S to A and S to B.

5.1.1 The Protocol

Every host shares a secret(password) with trusted third party denoted by P. They agree upon Diffie-Hellman [10] parameter set g(generator),p(large prime number)
STW 3-party EKE is as given below:
1.$A \Rightarrow B$ $\quad [R_A \oplus B]_{P_A}$
A chooses a random exponent N_A, keeps it secret and computes R_A=$g^{N_A}(mod\, p)$. Then,A encrypts $[R_A \oplus B]$ with his password P_A and sends the encrypted message as a request to B. After receiving A's request, B also chooses a random exponent N_B, keeps it secret, computes R_B=$g^{N_B}(mod\, p)$, then encrypts $[R_B \oplus A]$ with P_B . B forwards A's request with the encrypted message to S.

2.$B \Rightarrow S$ $\quad$ A, $[R_A \oplus B]_{P_A}, [R_B \oplus A]_{P_B}$
S decrypts $[R_A \oplus B]_{P_A}, [R_B \oplus A]_{P_B}$ with P_A and P_B respectively and responses $R_A^{N_S}$, $R_B^{N_S}$

to B, in which N_S is a random exponent S chose.

3.$S \Rightarrow B$ $\quad R_A^{N_S}$, $R_B^{N_S}$

B computes the session key $K = (R_A^{N_S})^{N_B} = g^{N_A.N_B.N_S}(mod\, p)$ and sends $R_B^{N_S}$ with a key confirmation message $[flow1]_K$ to A.

4.$B \Rightarrow A$ $\quad R_B^{N_S}$,$[flow1]_K$

A computes the session key $K = (R_B^{N_S})^{N_A} = g^{N_A.N_B.N_S}(mod\, p)$. A decrypts $[Flow1]_K$ with session key "K" and checks Flow-1 equals $[R_A \oplus B]_{P_A}$ to confirm that B posses the same session key K. A re-encrypts $[Flow1]_K$ with the session key K and responses it to B for key confirmation.

5.$A \Rightarrow B$ $\quad [[Flow1]_K]_K$

B decrypts $[[flow1]_K]_K$ with the session key K and checks $[Flow1]_K$ to confirm that A posses the same session key K.

5.1.2 Undetectable on-line guessing attacks

An attacker attempts to use a guessed password in an online transaction. He verifies the correctness of his guess using responses of S. If his guess fails he must start a new transaction with S using another guessed password. A failed guess can not be detected and logged by S, as S is not able to depart an honest request from a malicious request. This can be demonstrated in two scenarios. In these two scenarios, the attacker B, who is valid but malicious, completes the protocol with S and no participation of A is required.

Scenario 1 :

1. B: records $[R_A \oplus B]_{P_A}$

   The attacker B records $[R_A \oplus B]_{P_A}$ of an arbitrary run of the protocol. He guesses a password $\overline{P}_A$ and computes the value $\overline{R}_A$. He sets $R_B = \overline{R}_A$ and encrypts $\overline{R}_A \oplus A$ with his password P_B. Then, he sends S the message A,$[R_A \oplus B]_{P_A}$,$[\overline{R}_A \oplus A]_{P_B}$.

2. $B \Rightarrow S$ $\quad$ A, $[R_A \oplus B]_{P_A}, [\overline{R}_A \oplus A]_{P_B}$

   S decrypts $[R_A \oplus B]_{P_A}, [\overline{R}_A \oplus A]_{P_B}$ with P_A and P_B respectively and responses $R_A^{N_S}$,$\overline{R}_A^{N_S}$ to B, in which N_s is a random exponent S chose.

3. $S \Rightarrow B \qquad R_A^{N_S}, \overline{R}_A^{N_S}$

   The attacker B compares the two values. If $R_A^{N_S} = \overline{R}_A^{N_S}$ and so he has guessed the correct password $\overline{P}_A = P_A$

Scenario 2 :

1. B: $[\overline{R}_A \oplus B]_{\overline{P}_A}$

   The attacker B guesses a password $\overline{P}_A$ generates on behalf of A a random exponent $\overline{N}_A$ and computes $\overline{R}_A = g^{\overline{N}_A}(mod\, P)$. Then, B encrypts $[\overline{R}_A \oplus B]$ with the guessed password $\overline{P}_A$. Additionally, B chooses a random exponent N_B computes $R_B = g^{N_B}(mod\, P)$ and encrypts $[R_B \oplus A]$ with his password P_B. Then, he sends S the message A,$[\overline{R}_A \oplus B]_{\overline{P}_A}$,$[R_B \oplus A]_{P_B}$

2. $B \Rightarrow S \qquad$ A, $[\overline{R}_A \oplus B]_{\overline{P}_A}, [R_B \oplus A]_{P_B}$

   S decrypts $[\overline{R}_A \oplus B]_{\overline{P}_A}, [R_B \oplus A]_{P_B}$ with P_A and P_B respectively and responses $R_A^{N_S}$, $R_B^{N_S}$ to B. in which N_S is a random exponent S chose.

3. $S \Rightarrow B \qquad R_A^{N_S}, R_B^{N_S}$

   The attacker B computes the two values $(R_A^{N_S})^{N_B}$ and $(R_B^{N_S})^{\overline{N}_A}$. If they are equal, it follows that he has guessed the correct password $\overline{P}_A = P_A$

5.2 LSH 3-PEKE PROTOCOL

This protocol was proposed by Chun-Li Lin, Hung-Min Sun and Tzonelih Hwang [2]. This protocol is secure against both the off-line guessing attack and undetectable on-line guessing attacks [14] but also satisfies the security properties of perfect forward secrecy . The most important requirement to prevent undetectable on-line guessing attacks is to provide authentication of A and B to S(server). In step 2 of the STW 3-Party EKE, the message $[R_A \oplus B]_{P_A}, [R_B \oplus A]_{P_B}$ doesn't contain any verifiable information for S to authenticate A and B eventhough S uses correct passwords to decrypt that message. On the contrary, if there is any verifiable information for S combined with password P_A and P_B, it will result in off-line guessing attacks. One solution for this problem is by means of the help of server S's public key. Assuming that S's public key is a cryptographic

parameter which is secure against guessing and exhaustive attacks, and is well-known for all parties. Therefore, any verifiable information with a confounder encrypted with S's public key is able to withstand off-line guessing attacks. Based on such idea, a new three-party EKE protocol was proposed which is secure against both off-line guessing attacks and undetectable on-line guessing attacks [14].

5.2.1 The Protocol

The LSH 3-PEKE is described below:

1. $A \Rightarrow B$ $\quad A, \{ra, R_A, P_A\}_{K_s}$

   A chooses a confounder "ra" and a random exponent N_A, keeps them secret and computes $R_A = g^{N_A}(mod\ P)$. Then, A encrypts ra, R_A, P_A with server's public key K_s and sends the encrypted message as a request to B. After receiving A's request, B also chooses a confounder "rb" and a random exponent N_B, keeps them secret, computes $R_B = g^{N_B}(mod\ P)$ then encrypts rb, R_B, P_B with server's public key K_s. B forwards A's request with the encrypted message to S.

2. $B \Rightarrow S$ $\quad A, \{ra, R_A, P_A\}_{K_s}, \{rb, R_B, P_B\}_{K_s}$

   S decrypts $\{ra, R_A, P_A\}_{K_s}, \{rb, R_B, P_B\}_{K_s}$ with his private key and authenticates A and B by verifying their passwords P_A and P_B respectively. Then, S encrypts B, R_B with "ra", encrypts A, R_A with "rb" and responses them to B. Notice that the values "ra" and "rb" also act as one-time keys.

3. $S \Rightarrow B$ $\quad [B, R_B]_{ra}, [A, R_A]_{rb}$

   B decrypts $[A, R_A]_{rb}$ with "rb" and authenticates S by verifying the integrity of ID A. B computes the session key K=$R_A^{N_B}(mod\ P)$ and sends $[B, R_B]_{ra}$ with a key-confirmation message $[f(flow1), C_B]_K$ to A.

4. $B \Rightarrow A$ $\quad [B, R_B]_{ra}, [f(flow1), C_B]_K$

   A decrypts $[B, R_B]_{ra}$ with "ra" and authenticates S by verifying the integrity of ID B. A computes the session key K=$R_B^{N_A}(mod\ P)$ decrypts $[f(flow1), C_B]_K$ with the session key K and checks $f(flow1)? = f(A, \{ra, R_A, P_A\}_{K_s})$ to confirm that B possessed the same session key K. Then, A responses C_B to B for key confirmation.

5. $A \Rightarrow B$ C_B

B checks C_B to confirm that A possessed the same session key K.

5.2.2 Results And Discussion

SERVER PROGRAM

SERVER RUNNING.....

****

DIFFIE -HELLMEN PARAMETER SET

P :

894422361934331941320266039107881006695547901278516704146171564246280 2

983362976438648566265641032430444084355694406582175947026806420693338 2

85846500843697

g :

501397352772342931285583598718496400715622791724363912772790574725796 7

508114194569171423793180963728450529566697205851182871375034450811980 8

32852300901492

****

Public key Pair of server : 349427 , 732349

Private key Pair of server :23 ,732349

****

Received PKT- II $B \longrightarrow S$ [A,ra,RA,PAKS,rb,RB,PBKS]

SOURCE ID : A

Confounder[Encrypted] : 254460

RA[Encrypted] :

237034808456000989935402916779019874555093262165390085218274703814160 71

958588885255056886117422104148207215274136590323926723516402484255396 53

750799433526

PA[Encrypted] : 593246

DESTINATION ID : B

Confounder[Encrypted] : 417399

RB[Encrypted] :

889418517184999520689375294271323051723852105406388092480002432475854722517588037052597412556297994772787758614586679338622906804002575496287257219269280 4

PB[Encrypted] : 235976

****

DATA AFTER DECRYPTION[WITH SERVER PRIVATE KEY

CONFOUNDER[ra] : 7499

PASSWORD[PA] : 1007

CONFOUNDER[rb] : 2543

PASSWORD[PB] : 1109

****

RA :

237034808456000989935402916779019874555093262165390085218274703814160719585888852550568861174221041482072152741365903239267235164024842553965375079870145 1

RB :

889418517184999520689375294271323051723852105406388092480002432475854722517588037052597412556297994772787758614586679338622906804002575496287257219197925 7

****

PASSWORD OF SOURCE A : 1007

PASSWORD OF DESTINATION B : 1109

SOURCE PASS WORD MATCHED

DESTN. PASS WORD MATCHED

****

(RB)ra :

889418517184999520689375294271323051723852105406388092480002432475854722517588037052597412556297994772787758614586679338622906804002575496287257219197637 0

(RA)rb :

237034808456000989935402916779019874555093262165390085218274703814160719585888852550568861174221041482072152741365903239267235164024842553965375079870320 4

PKT-3 $S \longrightarrow B$: [B,RB]ra,[A,RA]rb TO DESTINATION DISPATCH

****

SOURCE PROGRAM

DIFFIE -HELLMEN PARAMETER SET

P :

894422361934331941320266039107881006695547901278516704146171564246280

298336297643864856626564103243044408435569440658217594702680642069333

8285846500843697

g :

501397352772342931285583598718496400715622791724363912772790574725796

750811419456917142379318096372845052956669720585118287137503445081198

0832852300901492

****

ENTER YOUR IDENTITY : A

CONFOUNDER GENARATED : 7499

ENTER PUBLIC KEY [e,n] OF SERVER : 349427 732349

ENCRYPTED CONFOUNDER[era] : 254460

****

RA :

237034808456000989935402916779019874555093262165390085218274703814160 7

195858888525505688611742210414820721527413659032392672351640248425539 6

53750798701451

ENTER UR PWD : 1007

Encrypted PWD : 593246

$A, \{ra, R_A, P_A\}_{K_S}$ FORWARDED TO DESTINATION

****

Received PKT- IV $B \longrightarrow A$ [B,RB]ra,[f(flow1),CB]

(B,RB)ra :

889418517184999520689375294271323051723852105406388092480002432475854722

517588037052597412556297994772787758614586679338622906804002575496287257

2191976370

(f(flow1),CB) :

13784811410543185516492913129443273087286611425694641104530447228315324387235562457079466084119200613801554669940005656262498473517756133173243283183493719DECRYPTED [RB] :889418517184999520689375294271323051723852105406388092480002432475854722517588037052597412556297994772787758614586679338622906804002575496287257219197925719**********FINAL KEY :13784811410543185516492913129443273087286611425694641104530447228315324387235562457079466084119200613801554669940005656262498473517756133173243283136519**********Flow1 : 31131269CB[Decrypted] : 26584557CB .. FORWARDED TO DESTINATION

DESTINATION PROGRAM

DIFFIE-HELLMAN PARAMETER SET:P :894422361934331941320266039107881006695547901278516704146171564246280298336297643864856626564103243044408435569440658217594702680642069333382858465006436975013973527723429312855835987184964007156227917243639127727905747257967508114194569171423793180963728450529566697205851182871375034450811980832852300901492**********Received Pkt1.[$A, \{ra, R_A, P_A\}K_S$] from source**********CONFOUNDER GENARATED : 2543ENTER UR ID : BENTER PUBLIC KEY [e,n] OF SERVER : 349427 732349

ENCRYPTED CONFOUNDER[erb] : 417399

****

RB :

889418517184999520689375294271323051723852105406388092480002432475854722517588037052597412556297994772787758614586679338622906804002575496287257219197925 7

Encrypted RB:

889418517184999520689375294271323051723852105406388092480002432475854722517588037052597412556297994772787758614586679338622906804002575496287257219269280 4

****

ENTER UR PWD : 1109

Encrypted PWD : 235976

PKT- II $B \longrightarrow S$ $[A, \{ra, R_A, P_A\}K_S, \{rb, R_B, P_B\}K_S]$ DISPATCHED

****

Received Pkt3. $S \longrightarrow B$: $[B, R_B]_{ra}, [A, R_A]_{rb}$ from server

(RB)ra :

889418517184999520689375294271323051723852105406388092480002432475854722517588037052597412556297994772787758614586679338622906804002575496287257219197637 0

(RA)rb :

237034808456000989935402916779019874555093262165390085218274703814160719558588885255056886117422104148207215274136590323926723516402484255396537507987032 04

RA [Decrypted] :

237034808456000989935402916779019874555093262165390085218274703814160719585888852550568861174221041482072152741365903239267235164024842553965375079870145 1

****

FINAL KEY VALUE :

137848114105431855164929131294432730872866114256946411045304472283153243872355624570794660841192006138015546699400056562624984735177561331732432831457365 1

****

Flow1 : 31131269

CB : 26584557

ENCRYPTED[f(flow1),CB] :

13784811410543185516492913129443273087286611425694641104530447228315324387235562457079466084119200613801554669940005656262498473517756133173243283

18349371

****

PKT- IV $B \longrightarrow A$ [B,RB]ra,[f(flow1),CB] DISPATCHED

Received CB:26584557

Received CB frm source is proper

This protocol uses public key infrastructure. Here RSA algorithm is used for this purpose. In the first two steps of protocol all the data are encrypted using server public key using RSA algorithm. This protocol provides key confirmation also in last two steps. As all the data are encrypted using server public key it is not vulnerable to dictionary attacks.

Chapter 6

MODIFIED 3-PARTY KEY AGREEMENT PROTOCOL

A new 3-party key agreement protocol is proposed which withstands all online and off-line guessing attacks ,which does not makes use of public key infrastructure. Several key agreement protocols are proposed on password based mechanism. These protocols are vulnerable to dictionary attacks. Storing clear text version of password on server is always not possible. In this protocol we use a trusted third party (key Distribution server) which mediates in key distribution. Rather than storing clear text version of password we store one way hash of the password at the server. Every host and server agree upon family of commutative hash functions using which host authenticates itself to server when it applies for session key . During this protocol run host establishes one time key with server using which server also authenticates to host. Moreover we don't use any public key infrastructure which needs large computational power. Since this is 3-party key agreement protocol every host need not share secret information with every other host.

6.1 THE PROTOCOL

Before we proceed we define one way hash function:
A one-way function is a function f such that for each x in the domain of f, it is easy to compute $f(x)$, but for essentially all y in the range of f, it is computationally infeasible to find any x such that $y = f(x)$.
The protocol is as described below:

1. $A \Rightarrow S$ $\quad A\ , B\ , H(P_A)[R_A]$

   A chooses a random no. "ra" and generates $R_A = g^{ra}(mod\ P)$ like in Diffie-Hellman protocol [10] where "g" is a generator of cyclic group and "P" is a large prime. It also generates one way hash of its password $H(P_A)$. A encrypts R_A with $H(P_A)$ and sends it to server along with ID s of participating entities. Server stores one way hash of password of every host (assumed to be pre distributed) using which it decrypts the above packet to get $R_A = g^{ra}(mod\ P)$.

2. $S \Rightarrow A$ $\quad H(P_A)(g^{rs_1} mod\ P),\ H(P_B)(g^{rs_2} mod\ P)$

   Server chooses random nos. rs_1 and rs_2. "S" generates $g^{rs_1}(mod\ P)$ and $g^{rs_2}(mod\ P)$ respectively. Using these quantities server establishes ephemeral keys with A and B respectively. Using this ephemeral keys "S" authenticates itself to "A" and "B". "S" computes these ephemeral keys as specified below:

   $K_{AS} = (g^{ra})^{rs_1} mod\ P$

   $K_{BS} = (g^{rb})^{rs_2} mod\ P$

   Note that K_{BS} can be computed only after fourth step of the protocol.

   $g^{rs_1}(mod\ P)$ and $g^{rs_2}(mod\ P)$ are encrypted with $H(P_A)$ and $H(P_B)$ respectively and dispatched to "A". A decrypts this packet with $H(P_A)$

3. $A \Rightarrow B$ $\quad F_A(P_A, K_{AS})\ , H(P_B)(g^{rs_2} mod\ P)$

   A decrypts this packet with $H(P_A)$ to get $g^{rs_1}(mod\ P)$. "A" establishes ephemeral key with "S" as $K_{AS} = (g^{rs_1})^{ra} mod\ P$. "A" calculates its predicate function $F_A(P_A, K_{AS})$ using which it authenticates itself to server. Since only "A" knows P_A only it can compute this predicate function. This is a commutative one way hash function .

   This value along with $H(P_B)(g^{rs_2} mod\ P)$ is forwarded to "B". B decrypts with $H(P_B)$ to get $(g^{rs_2} mod\ P)$

4. $B \Rightarrow S$ $\quad F_A(P_A, K_{AS}),\ F_B(P_B, K_{BS}), H(P_B)[R_B]$

   "B"chooses a random no. "rb" and generates $R_B = g^{rb}(mod\ P)$. It also generates one way hash of its password $H(P_B)$. "B" computes ephemeral key for authenticating server as $K_{BS} = (g^{rs_2})^{rb} mod\ P$. "B" calculates its predicate function $F_B(P_B, K_{BS})$ using which it authenticates itself to server. This predicate function

is a commutative one way hash function. Password of B and ephemeral session key K_{BS} are seeds for this function. Since only "B" knows P_B only it can compute this predicate function. After receiving this packet server decrypts with $H(P_B)$ to get R_B. Server computes ephemeral one time session key $K_{BS} = (g^{rb})^{rs_2} mod\ P$. using this K_{AS},K_{BS} server authenticates itself to hosts. K_{AS},K_{BS} changes with every protocol run. Server recomputes predicate functions of A and B i.e. $F_A(\ )$,$F_B(\ )$ and authenticates A and B respectively.

Server need not know P_A to compute this predicate function. Since this is a commutative hash function and $H(P_A)$ is pre distributed, using $K_{AS}, H(P_A)$ server can calculate the predicate function $F_A(\ )$ and authenticates A. Similarly using $K_{BS}, H(P_B)$ server can calculate the predicate function $F_B(\ )$ and authenticates B.

5. $S \Rightarrow B$ $\quad f_{K_{AS}}(R_B), f_{K_{BS}}(R_A), H_{K_{AS}}(R_A, R_B), H_{K_{BS}}(R_A, R_B)$
"S" encrypts R_B and R_A with K_{AS} and K_{BS} respectively. This defeats identity mis-binding attacks.$f(\ )$is a cryptographic transformation function."S" computes one way hash function$H_{K_{AS}}(R_A, R_B)$ using K_{AS}(one time key shared between A and server)using this host-A authenticates the server. Similarly "S" computes one way hash function $H_{K_{BS}}(R_A, R_B)$ using K_{BS}(one time key shared between B and server)using this host-B authenticates the server.

$f_{K_{AS}}(R_B), f_{K_{BS}}(R_A), H_{K_{AS}}(R_A, R_B), H_{K_{BS}}(R_A, R_B)$ are sent to B. After receiving this B decrypts $f_{K_{BS}}(R_A)$ with K_{BS} and gets R_A. Since K_{BS} is shared between server and B,it ensures B that R_A value is from authentic source. B recomputes one way hash $H_{K_{BS}}(R_A, R_B)$ using K_{BS} as key and authenticates server. B computes session key with A as $K_{AB} = (R_A)^{rb}(mod\ P)$ like in Diffie-Hellman protocol [10].

6. $B \Rightarrow A$ $\quad f_{K_{AS}}(R_B), H_{K_{AS}}(R_A, R_B), H_{K_{AB}}(N_{AB}), N_{AB}$
B forwards $f_{K_{AS}}(R_B), H_{K_{AS}}$ to A. A decrypts $f_{K_{AS}}(R_B)$ using K_{AS} to get R_B. Since K_{AS} is shared between server and A, it ensures A that R_B value is from authentic source. A computes session key with B as $K_{AB} = (R_B)^{ra}(mod\ P)$ like in Diffie-Hellman protocol [10]. B also computes a one way $H_{K_{AB}}(N_{AB})$ using K_{AB} and N_{AB} as seeds. Where N_{AB} is a random number. This one way hash is used for key confirmation(assures that both parties posses same session key). Since

N_{AB} is transmitted in plain there is no need of decryption. One way hash suffices decryption.

7. $A \Rightarrow B$ $\quad H_{K_{AB}}(H_{K_{AB}}(N_{AB}))$

   Using K_{AB} and N_{AB} A recomputes one way hash $H_{K_{AB}}(N_{AB})$ and verifies that B posses same key, K_{AB} as A. Using K_{AB} A once again calculates one way hash $H_{K_{AB}}(H_{K_{AB}}(N_{AB}))$ and sends to B. After receiving this B recomputes this one way hash using K_{AB} and verifies that A posses same session key(K_{AB}) as B.

6.2 COMMUTATIVE ONE-WAY HASH FUNCTIONS

Both host and server agree upon family of commutative hash functions $\{H_0, H_1, H_2, ...H_n\}$. Let H (P) be defined as $H_0(P)$, a member of a family of Commutative one-way hash functions. Host A calculates one way hash of its password as $H_0(P_A) = P_A^{h_0}(mod\ P)$,where h_0 is a random number(which it kepps as secret). We assume that one way hash of password $H_0(P)$ of every host is distributed to server. Since one way hash is irreversable nobody can compute P from $H_0(P)$. Host A calculates its predicate function $F_A(\ )$ as :
$H_0(H_{K_{AS}}(P_A)) = (P_A^{K_{AS}})^{h_0}(mod\ P)$.
Server Knows only one way hash of password P_A i.e. $H_0(P_A)$ using which it calculates predicate function of A as $H_{K_{AS}}(H_0(P_A))$.
$H_{K_{AS}}(H_0(P_A)) = (P_A^{h_0})^{K_{AS}}(mod\ P)$
Here K_{AS} is one time (ephemeral) key established between server and A. Since server knows K_{AS} and $H_0(P_A)$ it can compute this predicate function and authenticate A. Similarly it computes predicate function for B and authenticates B. Since these are commutative hash functions $H_{K_{AS}}(H_0(P_A)) = H_0(H_{K_{AS}}(P_A))$ i.e. $(P_A^{K_{AS}})^{h_0}(mod\ P) = (P_A^{h_0})^{K_{AS}}(mod\ P)$.

6.3 SECURITY ANALYSIS

Proposed protocol extends Two party version SAKA [1] to 3-party protocol. Every host need not share a secret with every other host. Since all the transactions in the protocol are done using one way hash of the password, host need not store clear text version of its password at server and it defeats dictionary attacks. Since this is a one way hash function there is no way to recover P from H(P). Every time host and server establish a

one time(ephemeral) key using which host authenticates server. Unlike traditional 3-party key agreement protocol we need not use long term(master) keys which leads to malacious insider attacks. In malicious insider attacks one of the participating parties turns hostile and misuse the information it has acquired in previous protocol runs and breaks the system. Server authenticates hosts using predicate functions. For calculating predicate functions server need not know password of hosts. Server can calculate predicate function using one way hash of the password of the hosts.

Even though $H(P)$ is compromised (under some rare circumstances) nobody can mimic the host to server as only legitimate hosts can compute predicate functions . Because host only knows its password which is a seed for predicate function. Nobody can mimic the server to the host even if $H(P)$ is compromised. It is equivalent to breaking Diffie-Hellman problem[10]. "S" encrypts R_B and R_A with K_{AS} and K_{BS} (one time ephemeral keys) respectively. This defeats identity mis-binding or masquerading attacks. Here the server acts just like authentication server not as monitoring server. This prevents malicious server from knowing session key and subsequently knowing all transactions. This protocol also ensures key integrity, key non disclosure, and key confirmation. Proposed protocol also ensures perfect forward key secrecy. Even if one of the session keys are compromised it will not lead to disclose of future keys. This protocol sustains online and off-line guessing attacks as there is no verifiable information present.

6.4 RESULTS AND DISCUSSION

SERVER PROGRAM

SERVER RUNNING.....

****

DIFFIE -HELLMEN PARAMETER SET

P :

12138636106431059491621290358412286839576116231165531425234532007 1

65568312340044675579825709081 489157457158870 77474420238844614265

19390093776802191225110 59

g :

11907729199496466390886678489775691771535007142983954218503418967 0

272358636182731452203365635033722738175924780216490096446074023516
3701209875835435676725 7
****
RECEIVED PKT-1
H(P)RA :
450860617222406300419888774521161902095161857482069331035135354615535027181887162402144360923991761194969085540527558077046356487655570668732280012756286 4
Decrypted RA:
669419147849771618786915150987434294812825385160818613273754632849101341094657781303794293016492796449969196322054637296528300259965271951809790735081847 0
(g.pow(rs1)) mod P :
105896702879490168015028198187385056008076767331840430517588848917274625865098583030127886347591505247051209102537491239263773837870126231572612421991 8095
H(Pa) (g.pow(rs1)) mod P :
119389058286104025522286403735371242270491581118942181313974069942960593334427499234110945744879375402999008955472013523403469256001602023401012700469937 21
(g.pow(rs2)) mod P :
394591762559215207764913352309990521462254459706570827371789841246741793602856957596134845456404096921490808804474960732322523746245718924359278866329 84
H(Pb)(g.pow(rs2)) mod P :
498186284965139789172312899773082736504000530936371831719371818800454797521658880855816161525038129298390490690071424549997612083703138171761177452252754 6
Secret key(one time) with A (K_{AS})=
617598085324987784237709316240330128442068968408230344688448073119062557888498208689510503457594784337426328223663504053770048999062787025936917548599675 2

Pkt-2 Dispatched.....

****

RECEIVED PKT-4 FA(PA,KAs), Fb(Pb,Kbs) ,H(PB)(RB) FROM DESTN

RECEIVED VALUES :

SOURCE PREDICATE :

156506744624926129356976652625782787911968245256182578123214850719876270620088543648285627817522889877556254698686078839113046871830108088363926183434648 9

DESTN PREDICATE :

109479135814411459918174743302117894419552835146608453067113214234301047162754040174103630173912019429160414994740171158660190816025527410658156172669660 34

H(PB)(RB) :

498186284965139789172312899773082736504000530936371831719371818800454797521658880855816161525038129298390490690071424549997612083703138171761774522527546

RB :

394591762559215207764913352309990521462254459706570827371789841246741793602856957596134845456404096921490808804474960732322523746245718924359278866329 84

Secret key(one time) with B (K_{BS})=

512212498053090802815885427610053396658517871012825054829099037487458314009316246512792839089917262644214684819576941501374597908475310037772519892730027

****

Predicate function of source Hk(H0(P)) :(Caliculated at server side)

156506744624926129356976652625782787911968245256182578123214850719876270620088543648285627817522889877556254698686078839113046871830108088363926183434648 9

Predicate function of Destination Hk(H0(P)) : (Caliculated at server side)

109479135814411459918174743302117894419552835146608453067113214234301047162740401741036301739120194291604149947401711586601908160255274106581561726966039 4

****

PREDICATE VERIFIED

VALID SOURCE : A

VALID DESTINATION : B

****

Fkas(RB) :

613664808286064512159743058568566886834062938841957086880633058236925902470262026494924368391854686922603689755409552306540551769857458861685332837464647246

Fkbs(RA) :

157714321354857921334035691826169479431608080082860024777604434473443309045757127224501306518441696331019179759360972868363455125015397916145325639955610 9

HKas(RA,RB) :7237724

Hkbs(RB,RA) :234806968

PKT-5 (Fkas(RB),Hkas,Fkbs(RA),Hkbs) DISPATCHED

SOURCE PROGRAM

DIFFIE -HELLMEN PARAMETER SET:

P :

121386361064310594916212903584122868395761162311655314252345320071655683123404467557982570908148915745715887077474420238844614265193900937768021912251105 9

g :

119077291994964663908866784897756917715350071429839542185034189670272358636182731452203365635033722738175924780216490096446040235163701209875835435676725 7

****

ENTER YOUR IDENTITY : A

ENTER DESTINATION IDENTITY : B

ENTER YOUR PASSWORD : 454545435234354545

****

Hash of password :

2187671133980143754901405612596440220308509820482974824804448396751

2652166763554602516526365450730233397829845723871185967442857653909

63261130156439870166

RA:

6694191478497716187869151509874342948128253851608186132737546328491 0

1341094657781303794293016492796449969196322054637296528300259965271 9

518097907350818470

H(P)RA:

4508606172224063004198887745211619020951618574820693310351353546155 3

5027181887162402144360923991761194969085540527558077046356487655570 6

687322800127562864

Dispatched Pkt-1 to Server

****

Received packet -2 H(Pa) (g.pow(rs1)),H(Pb) (g.pow(rs2)) from Server

H(Pa)(g.pow(rs1) mod p) :

1193890582861040255222864037353712422704915811189421813139740699429

6059333447499234110945744879375402999008955472013523403469256001602

02340101270046993721

H(Pb)[g.pow(rs2) mod p] :

4981862849651397891723128997730827365040005309363718317193718188004 5

4797521658880855816161525038129298390490690071424549997612083703138 1

717611774522527546

****

Secret key(one time) with server KAS=

6175980853249877842377093162403301284420689684082303446884480731190 62

5578884982086895105034575947843374263282236635040537700489990627870 25

9369175485996752

****

PREDICATE FUNCTION VALUE :

1565067446249261293569766526257827879119682452561825781232148507198762

7062008854364828562781752288987755625469868607883911304687183010808836

39261834346489

PKT-3 H(Pa)(g.pow(rs2)),Fa(Pa,Kas) DISPATCHED TO DESTN

****

Received (Fkas(RB),Hkas(RA,RB), H(KAB,RA)) from DESTINATION

RB : (DECRYPTED)

394591762559215207764913352309990521462254459706570827371789841246741793602856957596134845456404096921490808804474960732322523746245718924359278866329 84

Nonce (NAB): 5256

SERVER VERIFIED ::::VALID SERVER::::::

****

SESSION KEY (KAB) :

218249043683412182355254285217552589403899844291028940954026449506207468500523477114221141194422065320959420173534748182106912388247936981578906751618716 1

****

key conformation message Received from Destn. H(KAB,NAB) : 858699223

key conformation message computed by source H(KAB,NAB) : 858699223

KEY CONFIRMATION FROM SOURCE SIDE ENDS:::::VALID KEY

key conformation message to Destn. H(H(KAB,NAB)) : 602588964

PKT-7 key conformatio message H(H(KAB,NAB)): DISPATCHED TO DESTN

KEY ESTABLISHMENT ACCOMPLISHED

DESTINATION PROGRAM

DIFFIE -HELLMEN PARAMETER SET

P :

121386361064310594916212903584122868395761162311655314252345320071655683123404467557982570908148915745715887077474420238844614265193900937768021912251105 9

g :

119077291994964663908866784897756917715350071429839542185034189670272358636127314522033656350337227381759247802164900964460740235163701209875835435676725 7

****

ENTER UR ID : B

ENTER DESTINATION IDENTITY : A

ENTER YOUR PASSWORD : 3434343324434

Hash of password :

5020979838561340280569369167844073853109775816593486487161599054378521515605612818780174933335382193374929853827673208714769599214693953111107462855333666

PKT-3 H(Pa)(g.pow(rs2)),Fa(Pa,Kas) RECEIVED FROM SOURCE

****

H(Pb)(g.pow(rs2)) :

4981862849651397891723128997730827365040005309363718317193718188004547975216588808558161615250381292983904906900714245499976120837031381717611774522527546

Predicate of source :

1565067446249261293569766526257827879119682452561825781232148507198762706200885436482856278175228898775562546986860788391130468718301080883639261834346489

RB :

3945917625592152077649133523099905214622544597065708273717898412467417936028569575961348454564040969214908088044749607323225237462457189243592788632984

H(P)RB :

4981862849651397891723128997730827365040005309363718317193718188004547975216588808558161615250381292983904906900714245499976120837031381717611774522527546

****

Secret key(one time) with server KBS :

5122124980530908028158854276100533966585178710128250548290990374874583140093916246512792839089917262644214684819576941501374597908475310037772519892730027

PREDICATE FUNCTION VALUE (Destination) :

1094791358144114599181747433021178944195528351466084530671132142343010471

627404017410363017391201942916041499474017115866019081602552741065815617269660394

PKT-4 FA(PA,KAs), Fb(Pb,Kbs) ,H(PB)(RB) DISPATCHED TO SERVER.

****

PKT-5 Fkas(RB),Hkas,Fkbs(RA),Hkbs RECEIVED FROM SERVER

Fkbs(RA) :

157714321354857921334035691826169479431608080082860024777604434473443309045757127224501306518441696331019179759360972868363455125015397916145325639955610 9

Hkbs(RB,RA) : 234806968

RA :

669419147849771618786915150987434294812825385160818613273754632849101341094657781303794293016492796449969196322054637296528300259965271951809790735081847 0

SERVER VERIFIED ::::VALID SERVER::::::

****

SESSION KEY (KAB) :

218249043683412182355254285217552589403899844291028940954026449506207468500523477114221141194422065320959420173534748182106912388247936981578906751618716 1

****

GENRATED NONCE(NAB) : 5256

key conformation message to source H(KAB,NAB) : 858699223

PKT-6 Fkas(RB),Hkbs(RB,RA), H(KAB,RA) DISPATCHED TO SOURCE

****

key conformation message computed by Destn. H(H(KAB,NAB)) : 602588964

PKT-7 H(H(KAB,NAB)) RECEIVED FROM SOURCE

key conformation message from source H(H(KAB,NAB)) : 602588964

Received H(H(KAB,NAB)): is proper

KEY CONFIRMATION FROM DESTINATION SIDE ENDS:::::VALID KEY

One way hash of the password of host is stored at server. The data sent to server is encrypted using this one way hash. Predicate function is calculated using commutative hash functions using password as seed at client side and one way hash of the password at server. Host authenticates Server using one way hash computed using R_B and R_A as seeds. Key confirmation is provided through one way hash using session key and a random no. as seeds. Inclusion of random no. widens the key space in case of dictionary attacks. Encryption of R_B and R_A using one time keys provides user authentication. It also prvents malicious insider attacks.

Chapter 7

CONCLUSION AND FUTURE WORK

Strength of any crypto system relies upon strength of the encryption/decryption algorithm and strength of Key distribution mechanism. Frequent key changes are must inorder to minimize the amount of data compromised. Several applications demand throw away session keys especially financial applications. In traditional 2-party key distribution protocol every host should share a secret with every other host. 2-party key distribution protocol is not a particularly useful application but it can be used as a stepping stone for 3-party key distribution protocol . In 3PKDP every host need not share a secret with every other host. This greately simplifies the number of masters keys to be distributed. Every host shares a master key with Key Distribution centre(KDC). But 3PKDP(traditional) is vulnerable to insider attacks. In 3PKDP server genrates a session key and distributes to both parties,i.e. Server also knows the session key and it can monitor every transaction between two hosts. Which is a big problem in case of malacious servers.In most of the applications we want authentication server but not a monitoring server(except very few applicatons like national defense).

Diffe-Hellma Protocol [10]solves these problems. One of the advantages of using Diffie-Hellman key exchange [10] is its inherent "democracy" - both parties contribute equally to the resultant key without revealing their secrets. This and the relative simplicity of implementation make it quite attractive and practical to implement, especially in low end environments such as smartcards. Diffie-Hellman protocol [10] is vulnerable to several

attacks. It is obviously important to choose a group (i.e. P) large enough so that the best known algorithms for computing discrete logs are intractable. To defeat Man -in-the middle attack secret keys can be used in conjunction with message authentication codes.

Generic EKE [5] version is proved to be vulnerable to cryptanalysis . Unlike generic EKE, EKE-based on Diife-Hellman appears to be resistant to the dictionary attacks. Its resistance is due to the fact that the key is never communicated in any way. Instead, only residues are communicated in encrypted form. Even if the attacker obtains both residues, he does not come closer to discovering Key. Conversely, if the attacker somehow discovers Key, he cannot validate correct guesses of R_B and R_A thus making a dictionary attack impossible. Users sole means of authentication and sole long-term storage is a simple Password, rather than a bulky private key. EKE, required that both parties have clear text versions of the shared password, a constraint that cannot (or ought not) always be met. In particular, consider the problem of a user logging in to a computer that does not rely on a secure key server for authentication. It is inadvisable for most hosts to store passwords in either clear form or in a reversibly-encrypted form. EKE is 2-party key agreement protocol. Since every pair of hosts have to share a password,this limits its practical applications. In large communication environments, it is inconvenient in key management that every two communication parties mutually share a secret.

STW protocol is three party EKE protocol. Password based mechanism is the widely used method for authentication. Since STW protocol is 3-party protocol every host shares a password only with server. It is also based on Diffe-Hellman protocol. It is vulnerable to undetectable on-line password guessing attacks and off-line password guessing attacks [14]. Among the two classes of attacks, the off-line password guessing attack is the most comfortable and promising one for an attacker. It is also not noticed and has no communication cost. Comparing with off-line guessing attacks, undetectable on-line guessing attacks are much expensive due to the communication cost. Furthermore, undetectable on-line guessing attacks will probably fail if too many attempts notice a sensitive server.

LSH 3PEKE protocol [2] is immune to on-line guessing attacks, the off-line guessing attacks [14]. The off-line guessing attack will not work because nothing is encrypted by passwords. The only appearance of the password is the message encrypted with S's public key. Since the password is confounded with a confounder that is a sufficiently large ran-

dom number, the amount of guessing to verify the ciphertext is a half of the multiplication of the confounder space and the password guessing space on average. It is computational infeasible. Another way to get the password is direct to crack S's private key. It is also computational infeasible. In step 2 of protocol, S decrypts $\{ra, R_A, P_A\}_{K_s}, \{rb, R_B, P_B\}_{K_s}$ by his private key and verifies the correctness of passwords P_A and P_B to authenticates A and B. Thus, undetectable online guessing attacks also will not work. Although at the moment S can not confirm the freshness of the request, the following responses encrypted by the one-time keys ra and rb are able to guarantee the mutual authentication and the freshness to A and B. This protocol satisfies the property of perfect forward secrecy . It also satisfies the property of known-key security because the ephemeral random exponents N_A and N_B are independent among every protocol run.LSH 3-Party EKE protocol [2] uses server public keys for key exchange. Using public key infrastructure is inadvisable in key agreement as it needs extensive computaional power.However, the approach of using server public-keys is not always a satisfactory solution and is impractical for some environments. Communication parties have to obtain and verify the public-key of the server, a task which puts a high burden on the user. In fact, key distribution services without public-keys are quite often superior in practice than PKIs or are at least widely deployed.

User authentication is one of the most important security services in secure communications. It is necessary to verify the identities of the communication parties before they start a new connection. SAKA [1] protocol is a 2-party key agreement protocol which provides user authentication based on shared secret(password). This protocol is simple and efficient. It is secure against on-line and off-line guessing attacks. Since it is a 2-party protocol its applications are limited.

In proposed protocol public key infrastructure is not used. Since this is a 3-party protocol every host need not share a secret with other host. Proposed protocol provides host authentication and server authentication as a result man-in-the middle attacks are averted. It also spoils online and off-line guessing attacks as it uses one time keys,commutative hash functions for authentication. Hosts are not forced to store clear text version of password. Proposed protocol ensures perfect forward key secrecy ,key integrity. Proposed protocol also sustains malacious insider attacks as we use one time

keys for authentication. Server acts just like authentication server not like a monitoring server. Proposed protocol provides key confirmation also.

Future work

Proposed protocol can be extended to group key agreement protocol in distributed systems. Another possible optimization is to encrypt predicate functions $F_A(\ )$,$F_B(\ )$ with one time ephemeral keys K_{AS}, K_{BS} respectively. This will furthur strengthen the protocol against dictionary attacks [12]. Key confirmation(last two messages in protocol) can also be acheived through blind signature. But for that we have to use again public key infrastructure. During key confirmation step we may omit the random number but it opens up scope for known plain text attacks.

Bibliography

[1] Yeh Her-Tyan and Sun Hung-Min ,"Simple Authenticated Key Agreement Protocol Resistant to Password Guessing Attacks. "ACM SIGOPS Operating Systems Review.Volume 36,(October 2002):Issue 4.

[2] Lin C.L.,Sun H.M. and Hwang T.,"Three-party encrypted key exchange :Attacks and a solution." ACM Operating System Review.volume 36(2000):pp.12-20.

[3] Aiello William,Bellovin Steven M.,Blaze Matt."Efficient, DoS Resistant,Secure Key Exchange for Internet Protocols."Proceedings of the 9th ACM conference on Computer and communications security CCS '02, (November 2002):pp.78-85.

[4] Aiello William,Bellovin Steven M.,Blaze Matt,"Just Fast Keying: Key Agreement in a Hostile Internet."ACM Transactions on Information and System Security (TISSEC) Volume 7,Issue 2,(May 2004):pp.112-118.

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

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

[7] Tsudik Gene and Herreweghen Els Van."On simple and Secure key Distribution."Proceedings of the 1st ACM conference on Computer and communications security,(December 1993):pp.263-269.

[8] Parnerkar Amit,Guster Dennis,Herath Jayantha,"secret key Distribution protocol using public key cryptography." Journal of Computing Sciences in Colleges.Volume 19,Issue 1,(October 2003):pp134-140.

[9] A.M Barmawi,S Takada and N Doi."Augmented encrypted key exchange using RSA encryption." The 8th IEEE International Symposium on Personal Indoor and Mobile Radio Communications Volume2,(1-4Sept.1997):Pp:490-494.

[10] Diffie W. and Hellman M.E.,"New directions in cryptography." IEEE Transactions on Information Theory IT-11.(November 1976):pp.644-654.

[11] D. Denning and G. Sacco,"Timestamps in key distributed protocols."Communication of the ACM.Volume no.24, Issue 8,(1981):pp.533-535

[12] L. Lomas, M.A. Needham and R.M. Saltzer, "Protecting poorly chosen secrets from guessing attacks."IEEE Journal on Selected Areas in Communications.Volume 11,Issue 5,(June 1993):Page(s):648 - 656.

[13] M. Steiner, G. Tsudik and M. Waidner,"Refinement and extension of encrypted key exchange."ACM Operating Syst. Rev. vol.29,no.3, (1995):pp.22-30.

[14] Y. Ding and P. Horster, "Undetectable on-line password guessing attacks."ACM Operating Syst. Rev.vol.29,no.4,(1995):pp.77-86.

[15] Wilson S. Blake,Johason D. and Menezes A."Key Agreement Protocols and Their Security Analysis." In proceedings of the sixth IMA International Conference on Cryptography and Coding,Springer-Verlag LNCS 1355,(1997):pp.30-45.

[16] Maher David Paul,"CryptoBackup and Key Escrow."communications of the ACM. Vol.39,No.3,(March 1996):Page no. 48-53.

[17] bellare mihir,Rogaway Phillip."Provably secure session key distribution-the three party case."Proceedings of the twenty-seventh annual ACM symposium on Theory of computing STOC '95,(May 1995):ACM Press.

[18] Schneier Bruce.Applied Cryptography: Protocols and Algorithms.John Wiley and Sons.,1994.

[19] Menezes A.,Oorschot P.van and Vanstone S."Handbook of Applied Cryptography. CRC Press, 1996.

[20] Stallings Williams.cryptography and network security.3rd edition, Pearson education,2004.

[21] Mel H.X.,Baker Doris.M. and Burnett Steve. Cryptography Decrypted.Addison-Wesley,2004.

[22] Strangio M.A ."An optimal round two-party password-authenticated key agreement protocol. "The First International Conference on Availability, Reliability and Security, (20-22 April 2006):Page(s):8 pp.

[23] Al Sultan K., Saeb M., El-Raouf Badawi." A new two-pass key agreement protocol."46th IEEE International Midwest Symposium on Circuits and Systems, 2003, Volume 1,(27-30 Dec):Page(s):509 - 511.

[24] Harn L., Hsin W J., Mehta M."Authenticated Diffie-Hellman key agreement protocol using a single cryptographic assumption." Communications, IEE Proceedings,Volume 152, Issue 4,(Aug 2005):Page(s):404 - 410.

[25] Popescu C."A secure authenticated key agreement protocol." Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference, Volume 2,(12-15 May 2004):Page(s):783 - 786.