A Novel Secure Key Agreement Protocol using Trusted Third Party

Sairam Kulkarni kulkarnisairam@gmail.com
Department of Computer Science & Engg.
National Institute of Technology Rourkela
Rourkela, 769008, India

Debasish Jena debasishjena@hotmail.com
Assistant Professor
Centre for IT Education
Biju Patnaik University of Technology
Bhubaneswar, 751010, India

Sanjay Kumar Jena skjena@nitrkl.ac.in
Professor
Department of Computer Science & Engg.
National Institute of Technology Rourkela
Rourkela, 769008, India

Abstract

In the past, several key agreement protocols are proposed on password based mechanism. These protocols are vulnerable to dictionary attacks. Storing plain text version of password on server is not secure always. In this paper we utilize the service of a trusted third party, i.e., the Key Distribution server (KDS) for key agreement between the hosts. Now-a-days in large working environments two party key agreement protocols are being rarely used. In this proposed scheme, instead of storing plain text version of password we store one way hash of the password at the server. Every host and server agree upon family of commutative one-way hash functions, using which host authentication is done when a host applies for session key with KDS. Host establishes one time key with server using which server authentication is done. Due to this man-in-the middle attacks are defeated. The proposed protocol is based on Diffie-Hellman key exchange protocol.

Keywords: Key Agreement, Diffie-Hellman, Online guessing attacks, Dictionary attacks.

1. INTRODUCTION

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 [1].

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 [2]. 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 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 [3] [4].

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 [5].

In a secure system, passwords can be easily guessed if user chooses their own password in plain text [6]. Storing plain text version of password on server is not secure. This weakness exists in practically all widely used systems. The proposed protocol is secure against dictionary attacks as we use one time keys with server. This protocol is also secure against malicious insider attacks, where a host misuses the information in one protocol run to another. Proposed protocol also provides perfect forward secrecy i.e. even if one key is disclosed future session keys will not be disclosed. As we don’t use any Public Key Infrastructure (PKI), large computational power is not required. Since this is a third-party key agreement protocol every host need not share secret information with other host.

In this paper in Section 2, we review short comings of existing protocols. In section 3 we discuss our new third-party Key Agreement Protocol. Formal security analysis of proposed protocol is done in Section 4. Finally concluding remarks is done in Section 5.

2. RELATED WORK

DH-BPAKE [7] is a two party key agreement protocol based on Diffie-Hellman [8] and Encrypted key exchange protocols which were proposed by Strangio [9]. This protocol is not suitable for large networks where we cannot assume that every party shares a secret (password) with every other party. Simple Authenticated Key Agreement (SAKA) protocol proposed by Her-Tyan Yeh et al. [10] is also a two party key agreement protocol which is based on password based authentication and Diffie-Hellman key agreement. User authentication is one of the most important security services in secure communications. It is necessary to verify the identities of the communicating parties before they start a new connection. Password-based mechanism is the most widely used method for user authentication since it allows people to choose and remember their own password without any assistant device. This protocol is simple and cost effective, but is being rarely used in large networks.

STW protocol is a three party Encrypted key exchange protocol proposed by Steiner et al. [11]. Since this is a three party key agreement protocol, both the hosts share a secret key only with trusted third party. Ding et al [12] have proved that this protocol is vulnerable to undetectable online guessing attacks. According to Lin C.L. et al [13], this protocol is also vulnerable to offline guessing attacks. An attacker attempts to use a guessed password in an online transaction. Host verifies the correctness of his guess using responses from server. If his guess fails he must start a new transaction with server using another guessed password. A failed guess can not be detected and logged by server, as server is not able to depart an honest request from a malicious request. In off-line guessing attacks an attacker guesses a password and verifies his guess off-line. No participation of server is required, so server does not notice the attack. If his guess fails, the attacker tries again with another password, until he finds the proper one. Among these classes of attacks, the off-line password guessing attack is the most comfortable and promising one for an attacker. It is not noticeable and has no communication cost. Storing a plain text version of the shared password at the server is 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 plain form or in a reversibly encrypted form.

LSH 3-PEKE protocol was proposed by Chun-Li Lin et al [13]. This protocol is secure against both the offline guessing attack and undetectable on-line guessing attacks 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 host to server. In the STW 3-PEKE, there is no verifiable information for server to authenticate host. On the contrary, if there is any verifiable information for server combined with password will result in offline guessing attacks. LSH 3-PEKE uses server public keys for this purpose. But this is not a satisfactory solution all the times 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 PKI.

3. PROPOSED 3-PARTY KEY AGREEMENT PROTOCOL

Our proposed protocol withstands all online [12] and offline guessing attacks [13], and does not makes use of PKI. Every host and server agree upon family of commutative one-way hash functions using which host authentication is done when it applies for session key. Host establishes one time key with server using which server authentication is done. Rather than storing a plain text version of password we store one way hash of password at server. A one-way function is a function f such that for each x in the domain of f, it is easy to compute $y = f(x)$, but it is computationally infeasible to find any x given $f(x)$.

3.1 Notations

In this paper, we use the following notations

A, B	Full principal names
S	Trusted Third Party
$E_K(X)$	Encryption of plaintext block X under key K
$D_K(X)$	Decryption of plaintext block X under key K
K_{AB}	A and B share Key K
$H_{K_{AB}}(X)$	One way hash of X using key KAB
N_{AB}	Nonce generated by A and received by B
P_A	Password of A
$H(P_A)$	One way hash of password of A
g	Generator of cyclic group
P	Large prime number
$A \rightarrow B \quad M$	A sends message "M" to B

3.2 Proposed Protocol

In this subsection, we describe the steps involved in detail.

i. *A* chooses a random number ra and generates $R_A = g^{ra} (\bmod \quad p)$ then encrypts R_A with $H(P_A)$. After calculating the values sends it to server along with IDs of participating entities.

$$A \rightarrow S \qquad ID_A, ID_B, H(P_A)[R_A]$$

ii. After receiving the values sent by *A*, server *S* decrypts the packet to get R_A by previously distributed one way hash of password of *A*. server randomly chooses $rs1$ and $rs2$ and computes ephemeral key with *A* as follows

$$K_{AS} = (R_A)^{rs1} (\bmod p) = (g^{ra})^{rs1} \bmod p$$

S generates $g^{rs1} (\bmod p)$ and $g^{rs2} (\bmod p)$ and encrypts with $H(P_A)$ and $H(P_B)$ respectively. Using these quantities server establishes ephemeral keys with *A* and *B* respectively and server authentication is done. *S* sends the values to *A*

$$S \rightarrow A \qquad H(P_A)(g^{rs1} \bmod p), H(P_B)(g^{rs2} \bmod p)$$

iii. *A* decrypts this packet with $H(P_A)$ to get $g^{rs1} (\bmod p)$ and establishes ephemeral key with *S* as $K_{AS} = (g^{rs1})^{ra} \bmod p$.*A* calculates one way function $F_A(P_A, K_{AS})$ using which server authenticates *A*, since only *A* knows P_A it can compute this function. As this is a commutative one way hash function [14], server need not know host password to evaluate this function. Using one way hash of host password server can calculate predicate function and authenticate host. *A* sends the following values to *B*

$$A \rightarrow B \qquad F_A(P_A, K_{AS}), H(P_B)(g^{rs2} \bmod p)$$

iv. After receiving the values *B* decrypts it with $H(P_B)$ to get $(g^{rs2} \bmod p)$.*B* chooses randomly rb and generates $R_B = g^{rb} (\bmod p)$ .Then computes ephemeral key for authenticating server as $K_{BS} = (g^{rs2})^{rb} \bmod p$. *B* calculates one way function $F_B(P_B, K_{BS})$, using which server authenticates *B*. Password of *B* and ephemeral session key K_{BS} are seeds for this function. Since only *B* knows P_B it can compute this function and sends the values to *S*.

$$B \rightarrow S \qquad F_A(P_A, K_{AS}), F_B(P_B, K_{BS}), H(P_B)[R_B]$$

v. server decrypts it with $H(P_B)$ to get R_B and computes ephemeral key $K_{BS} = (g^{rb})^{rs2} \bmod p$. For authentication of *A* and *B* server evaluates one way functions $F_A(...), F_B(...)$. server need not know host passwords to evaluate these functions. Using one way hash of host password it can evaluate this function as it is a commutative one way hash function. If it results into true then it confirms that host is genuine. It defines a predicate as $T(H(P), F(P,K), K)$. This evaluates to true if and only if the genuine password *P* was used to create both $H(P)$ and $F(P,K)$. *K* can be K_{AS}, K_{BS} for A and B respectively. *S* encrypts R_B and R_A with K_{AS}, K_{BS} respectively and 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) and authenticates *B* and sends the values to *B*.

$$S \rightarrow B \qquad E_{K_{AS}}(R_B), E_{K_{BS}}(R_A), H_{K_{AS}}(R_A, R_B), H_{K_{BS}}(R_A, R_B)$$

vi. After receiving this *B* decrypts $E_{K_{BS}}(R_A)$ 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* computes 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 $K_{AB} = (R_A)^{rb} (\bmod p)$. *B* computes a one way hash $H_{K_{AB}}(N_{AB})$ 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. After computing all the values it sends to *A.*

$$B \rightarrow A \qquad E_{K_{AS}}(R_B), H_{K_{AS}}(R_A, R_B), H_{K_{AB}}(N_{AB}), N_{AB}$$

vii. *A* decrypts $E_{K_{AS}}(R_B)$ 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 $K_{AB} = (R_B)^{ra} (\bmod p)$ .Using K_{AB} and N_{AB} *A* computes 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*.

$$A \rightarrow B \quad H_{K_{AB}}(H_{K_{AB}}(N_{AB}))$$

viii. Finally, after receiving this *B* computes this one way hash using K_{AB} and verifies that *A* posses same session key (K_{AB}) as *B*.

The detail is explained in Fig-1.

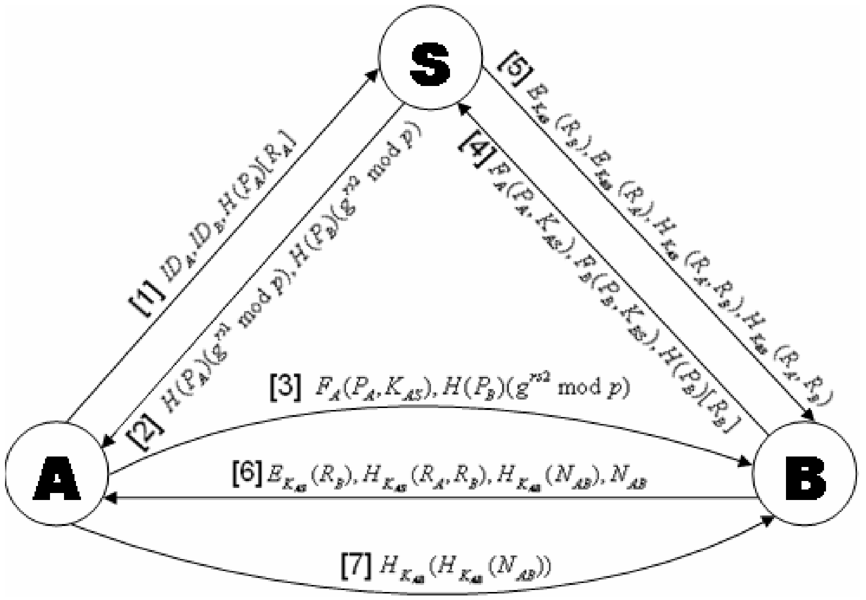
S
A
B
[1] $ID_A, ID_B, H(P_A)[R_A]$
[2] $H(P_A)(g^{rs1} \bmod p), H(P_B)(g^{rs2} \bmod p)$
[3] $F_A(P_A, K_{AS}), H(P_B)(g^{rs2} \bmod p)$
[4] $F_A(P_A, K_{AS}), F_B(P_B, K_{BS}), H(P_B)[R_B]$
[5] $E_{K_{AS}}(R_S), E_{K_{BS}}(R_A), H_{K_{AS}}(R_A, R_B), H_{K_{BS}}(R_A, R_B)$
[6] $E_{K_{AS}}(R_B), H_{K_{AS}}(R_A, R_B), H_{K_{AB}}(N_{AB}), N_{AB}$
[7] $H_{K_{AB}}(H_{K_{AB}}(N_{AB}))$

FIGURE 1: Proposed Protocol.

3.3 Commutative one way hash functions

Both host and server agree upon family of commutative one-way hash functions $\{H_0, H_1, H_2 ....H_N\}$ [14]. 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 $H_0(P_A) = (P_A)^{h_0} (\bmod p)$, where h_0 is a random number. We assume that one way hash of password $H_0(P)$ of every host is distributed to server. Since one way hash is irreversible nobody can compute P from $H_0(P)$. Host *A* calculates its one way function as $F_A(P_A, K_{AS}) = H_{K_{AS}}(P_A) = (P_A^{K_{AS}})(\bmod p)$ and sends to server. 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 $H_{K_{AS}}(H_0(P_A)) = (P_A^{h0})^{K_{AS}} (\bmod p)$. Server computes $H_0(H_{K_{AS}}(P_A)) = (P_A^{K_{AS}})^{h_0} (\bmod p)$. Here $H_{K_{AS}}(P_A) = (P_A^{K_{AS}})(\bmod p)$ is sent by the host. Now server checks $H_{K_{AS}}(H_0(P_A))$ equals $H_0(H_{K_{AS}}(P_A))$ or not. If these two are equal it confirms server that host is genuine. Much better implementation of commutative one way hash functions can be found.

4. SECURITY ANALYSES

In this section, we provide formal security analysis of our protocol. Hosts are not forced to store plain text version of password at server as a result this protocol is not vulnerable to password file compromise attacks [14]. Though $H(P)$ is compromised there is no way to recover P from $H(P)$. Even $H(P)$ is compromised no body can mimic the host to server as only genuine host can compute one way function- $F_A(...), F_B(...)$ etc. Because only host knows password, which is seed for this function. This protocol provides host authentication and server authentication as a result man-in-the middle attacks are averted. Server authentication is done through one time keys it defeats malicious insider attacks [15]. This is a type of attack where a genuine host turns out to be hostile in subsequent protocol run and misuses the information that it has already acquired in previous protocol run.

Online guessing attacks are not possible since R_A, R_B are encrypted with one time keys. Dictionary attacks and offline guessing attacks are not possible since there is no verifiable information present in the protocol run to verify attacker's guess. This protocol also provides perfect forward key secrecy. It also provides Key non-disclosure, Key integrity, and Key confirmation. We use one way hash functions for authentication and key confirmation as conventional encryption and decryption makes protocol design messy [15]. One way hash function suffices decryption. N_{AB} in last step multiplies key space to be searched in case of brute force attack. To guard further against dictionary attacks one way function- $F_A(...), F_B(...)$ may be encrypted with K_{AS}, K_{BS} respectively. Even if *H (P)* is compromised it is equivalent to breaking Diffie-Hellman protocol [8]. Since R_A, R_B are encrypted with $H(P_A)$ and $H(P_B)$ respectively this averts identity mis-binding attacks.

5. CONCLUSION

We propose a three party protocol secure against online and dictionary attacks. It provides host and server authentication. Hosts are not forced to store plain text version of password at server. Proposed protocol does not make use of any public key infrastructure. Instead of commutative one way hash functions digital signatures can also be used for host authentication purpose.

6. ACKNOWLEDGMENT

This research was supported by Department of Communication and Information Technology, Government of India under Information Security Education and Awareness Project and being carried out at department of Computer Science and Engineering, NIT Rourkela, INDIA.

7. REFERENCES

1. Menezes A.,Oorschot P. van and Vanstone S. *"Handbook of Applied Cryptography",* CRC Press, (1996)
2. Schneier Bruce. "*Applied Cryptography: Protocols and Algorithms"*, John Wiley and Sons, (1994)
3. Stallings Williams. *"Cryptography and Network Security"*, 3rd Edition, Pearson Educa- tion, (2004)
4. Mel H.X.,Baker Doris.M. and Burnett Steve. "*Cryptography Decrypted"*, Addison- Wesley, (2004)
5. Bellare Mihir, Rogaway Phillip. *"Provably Secure Session Key Distribution-The Three Party Case"*. In Proceedings of the 27th annual ACM symposium on Theory of computing STOC '95, ACM Press, May 1995
6. L. Gong, M. A. Lomas, R. M. Needham, and J. H. Saltzer, *"Protecting Poorly Chose Secrets From Guessing Attacks".* SELECTED AREAS IN COMMUNICATIONS, vol. 11, no. 5, pp. 648–656, June 1993
7. M. Strangio, *"An Optimal Round Two-Party Password-Authenticated Key Agreement Protocol".* In The First International Conference on Availability, Reliability and Security, p. 8, April 2006
8. W. Diffie and M. Hellman, "*New Directions In Cryptography".* IEEE Transactions on Information Theory IT-11, pp. 644–654, November 1976
9. S. Bellovin and M. Merritt, *"Encrypted Key Exchange: Password Based Protocols Secure Against Dictionary Attacks".* In Proceedings IEEE Symposium on Research in Security and Privacy, pp. 72–84, 1992

10. Y. Her-Tyan and S. Hung-Min, *“Simple Authenticated Key Agreement Protocol Resistant To Password Guessing Attacks”,* ACM SIGOPS Operating Systems Review, vol. 36, no. 4, pp. 14–22, October 2002
11. M. Steiner, G. Tsudik, and M. Waidner, *“Refinement And Extension Of Encrypted Key Exchange”.* ACM Operating System Review, vol. 29, no. 3, pp. 22–30, 1995
12. Y. Ding and P. Horster, *“Undetectable On-Line Password Guessing Attacks”.* ACM Operating System Review, vol. 29, no. 4, pp. 77–86, October1995
13. C. L. Lin, H. M. Sun, and Hwang, *“Three-Party Encrypted Key Exchange: Attacks And A Solution”.* ACM Operating System Review, vol. 34, no. 4, pp. 12–20, October 2000
14. S. Bellovin and M. Merritt, *“Augmented Encrypted Key Exchange: A Password Based Protocols Secure Against Dictionary Attacks And Password File Compromise”.* In 1st ACM Conf. on Computer and Communications Security. ACM Press, pp. 244–250, December 1993
15. T. Gene and H. Van, *“On Simple and Secure Key Distribution”.* In Proceedings of the 1st ACM conference on Computer and communications security CCS 93. ACM Press, pp. 49–57, December 1993s