Password-Only Authenticated Key Exchange Using Distributed Server

N.Narmadha[1], S. Rajathi[2]
[1](Department of Computer Science, SRM University, India)
[2](Department of Information Technology, Assistant Professor,SRM University, India)

ABSTRACT : *Authentication using Password-authenticated key exchange using distributed server (PAKEUDE) is done where a cryptographic key - exchange of messages. Database of all passwords to authenticate clients are stored in a distributed server. If the server is compromised, the attacker cannot act like a client with the information from the compromised server. Solution produced for distributed-server PAKE is by having parallel two peer servers which have equal contribution to authentication or asymmetric solution for distributed-server PAKE, where the client can establish different cryptographic key with the control server.*

Keywords - *Distributed Server (DS) , Dictionary Attack (DA) , Diffie-Hellman Key Exchange, ElGamal Encryption, Password-authenticated Key exchange using Distributed Server(PAKEUDE).*

INTRODUCTION

Passwords are used for secure authentication of log in process that controls access to secure transformation of messages, steganography , encryption decryption techniques , cryptography and so on. A system user may require passwords for many purposes: logging in to computer accounts, retrieving e-mail from servers, accessing programs, databases, networks, websites.

Earlier password-based authentication systems transmitted a cryptographic hash of the password over a public channel which makes the hash value accessible to an attacker. When this is done, and it is very common, the attacker can work offline, rapidly testing possible passwords against the true password's hash value. Studies have consistently shown that a large fraction of user-chosen passwords are readily guessed automatically. For example, according to Bruce Schneier, examining data from a 2006 phishing attack, 55 percent of MySpace passwords would be crackable/trackable in 8 hours using a commercially available Password Recovery Toolkit capable of testing 200,000 passwords per second in 2006. Recent research advances in password-based authentication have allowed a client and a server mutually to authenticate with a password and meanwhile to establish a cryptographic key for secure communications after authentication. In general, current solutions for password based authentication follow two models.

The first model, called PKI-based model, assumes that the client keeps the server's public key in addition to share a password with the server. In this setting, the client can send the password to the server by public key encryption. Gong et al., were the first to present this kind of authentication protocols with heuristic resistant to offline dictionary attacks, and Halevi and Krawczyk were the first to provide formal definitions and rigorous proofs of security for PKI-based model.

The second model is called password-only model. Bellovin and Merritt were the first to consider authentication based on password only, and introduced a set of so-called "encrypted key exchange" protocols, where the password is used as a secret key to encrypt random numbers for key exchange purpose. Formal models of security for the password-only authentication were first given independently by Bellare et al. and Boyko et al. Katz et al. were the first to give a password-only authentication protocol which is both practical and provably secure under standard cryptographic assumption.

Based on the identity-based encryption technique, Yi et al. suggested an identity-based model where the client needs to remember the password only while the server keeps the password

in addition to private keys related to its identity. In this setting, the client can encrypt the password based on the identity of the server. This model is between the PKI-based and the password only models.

Typical protocols for password-based authentication assume a single server stores all the passwords necessary to authenticate clients. If the server is compromised, due to, for example, hacking, or installing a "Trojan horse," or even insider attack, user passwords stored in the server are disclosed. To address this issue, two-server password-based authentication protocols were introduced where two servers cooperate to authenticate a client on the basis of password and if one server is compromised, the attacker still cannot pretend to be the client with the information from the compromised server.

Current solutions for two-server PAKE are either symmetric in the sense that two peer servers equally contribute to the authentication, asymmetric in the sense that one server authenticates the client with the help of another server . A symmetric two server PAKE protocol, for example, Katz et al.'s protocol , can run in parallel and establishes secret session keys between the client and two servers, respectively. In case one of the two servers shuts down due to the denial-of-service attack, another server can continue to provide services to authenticated clients. In terms of parallel computation and reliable service, a symmetric protocol is superior to an symmetric protocol. So far, only Katz et al.'s two-server PAKE protocol has been symmetric. But their protocol is not efficient for practical use. An asymmetric two-server PAKE protocol runs in series and only the front-end server and the client need to establish a secret session key. Current asymmetric protocols, for example, Yang et al.'s protocol and Jin et al.'s protocol , need two servers to exchange messages for several times in series. These asymmetric designs are less efficient than a symmetric design which allows two servers to compute in parallel.

In this paper, we propose a new asymmetric solution for authentication using distributed server. In all existing two server PAKE protocols, a client can be authenticated directly by the server(s). In our protocol we are using one control server (CS) and service server(s) to authenticate a client. The CS get the password at the time of registration and split that password into two halves and send pwd1 to the service server 1 (SS1) and second half to the service server2 (SS2).

Security analysis has shown that our protocol is secure against both passive and active attacks in case that one server is compromised. Performance analysis has shown that our protocol is more efficient than existing symmetric and asymmetric two-server PAKE protocols in terms of parallel computation.

Our protocol can be applied in distributed systems where multiple servers exist. For example, Microsoft active directory domain service (AD DS) is the foundation for distributed networks built on Windows server operating systems that use domain controllers. AD DS provides structured and hierarchical data storage for objects in a network such as users, computers, printers, and services. AD DS also provides support for locating and working with these objects. For a large enterprise running its own domain, there must be two AD DS domain controllers, for fault-tolerance purpose. To authenticate a user on a network, the user usually needs to provide his/her identification and password to one AD DS domain controller. Based on our two-server PAKE protocol, we can split the user's password into two parts and store them, respectively, on the two AD DS domain controllers, which can then cooperate to authenticate the user. Even if one domain controller is compromised, the system can still work. In this way, we can achieve more secure AD DS.

I. KEY REQUIREMENTS

Diffie–Hellman key exchange (**D–H**)[nb 1] is a specific method of exchanging cryptographic keys. It is one of the earliest practical examples of key exchange implemented within the field of cryptography. The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an

insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.

The scheme was first published by Whitfield Diffie and Martin Hellman in 1976, although it had been separately invented a few years earlier within GCHQ, the British signals intelligence agency, byJames H. Ellis, Clifford Cocks and Malcolm J. Williamson but was kept classified. In 2002, Hellman suggested the algorithm be called **Diffie–Hellman–Merkle key exchange** in recognition ofRalph Merkle's contribution to the invention of public-key cryptography (Hellman, 2002).

Although Diffie–Hellman key agreement itself is an *anonymous* (non-*authenticated*) key-agreement protocol, it provides the basis for a variety of authenticated protocols, and is used to provideperfect forward secrecy in Transport Layer Security's ephemeral modes (referred to as EDH or DHE depending on the cipher suite).

The method was followed shortly afterwards by RSA, an implementation of public key cryptography using asymmetric algorithms.

Diffie-Hellman key agreement is not limited to negotiating a key shared by only two participants. Any number of users can take part in an agreement by performing iterations of the agreement protocol and exchanging intermediate data (which does not itself need to be kept secret). For example, Alice, Bob, and Carol could participate in a Diffie-Hellman agreement as follows, with all operations taken to be modulo p:

1. The parties agree on the algorithm parameters p and g.
2. The parties generate their private keys, named a, b, and c.
3. Alice computes g^a and sends it to Bob.
4. Bob computes $(g^a)^b = g^{ab}$ and sends it to Carol.
5. Carol computes $(g^{ab})^c = g^{abc}$ and uses it as her secret.
6. Bob computes g^b and sends it to Carol.
7. Carol computes $(g^b)^c = g^{bc}$ and sends it to Alice.
8. Alice computes $(g^{bc})^a = g^{bca} = g^{abc}$ and uses it as her secret.
9. Carol computes g^c and sends it to Alice.
10. Alice computes $(g^c)^a = g^{ca}$ and sends it to Bob.
11. Bob computes $(g^{ca})^b = g^{cab} = g^{abc}$ and uses it as his secret.

An eavesdropper has been able to see g^a, g^b, g^c, g^{ab}, g^{ac}, and g^{bc}, but cannot use any combination of these to reproduce g^{abc}.

To extend this mechanism to larger groups, two basic principles must be followed:

- Starting with an "empty" key consisting only of g, the secret is made by raising the current value to every participant's private exponent once, in any order (the first such exponentiation yields the participant's own public key).
- Any intermediate value (having up to $N-1$ exponents applied, where N is the number of participants in the group) may be revealed publicly, but the final value (having had all N exponents applied) constitutes the shared secret and hence must never be revealed publicly. Thus, each user must obtain their copy of the secret by applying their own private key last (otherwise there would be no way for the last contributor to communicate the final key to its recipient, as that last contributor would have turned the key into the very secret the group wished to protect).

These principles leave open various options for choosing in which order participants contribute to keys. The simplest and most obvious solution is to arrange the N participants in a circle and have N keys rotate around the circle, until eventually every key has been contributed to by

all N participants (ending with its owner) and each participant has contributed to N keys (ending with their own). However, this requires that every participant perform N modular exponentiations.

By choosing a more optimal order, and relying on the fact that keys can be duplicated, it is possible to reduce the number of modular exponentiations performed by each participant to $\log_2(N) + 1$ using a divide-and-conquer-style approach, given here for eight participants:

1. Participants A, B, C, and D each perform one exponentiation, yielding g^{abcd}; this value is sent to E, F, G, and H. In return, participants A, B, C, and D receive g^{efgh}.
2. Participants A and B each perform one exponentiation, yielding g^{efghab}, which they send to C and D, while C and D do the same, yielding g^{efghcd}, which they send to A and B.
3. Participant A performs an exponentiation, yielding $g^{efghcda}$, which it sends to B; similarly, B sends $g^{efghcdb}$ to A. C and D do similarly.
4. Participant A performs one final exponentiation, yielding the secret $g^{efghcdba} = g^{abcdefgh}$, while B does the same to get $g^{efghcdab} = g^{abcdefgh}$; again, C and D do similarly.
5. Participants E through H simultaneously perform the same operations using g^{abcd} as their starting point.

Once this operation has been completed all participants will possess the secret $g^{abcdefgh}$, but each participant will have performed only four modular exponentiations, rather than the eight implied by a simple circular arrangement.

II. OUR PROTOCOL

Our protocol runs in three phases—registration, authentication and correctness.

Registration

In registration client can send the password to the control server (CS). After receiving the password from client CS can split this password into two halves pwd1 and pwd2. Pwd1 can send to the Service server 1 and Pwd2 can send to the service server 2 after establishing a key exchange between control server and service server.

Authentication

Step 1: Client can send the password (P) with random number (r1) to the CS.

Step 2: CS can receive the P and established a session key with service server SK1 and SK2.

Step 3: CS can split the password P into P1 and P2 and send to service server M2 = {p1, r1,SK1} , M2 = {p2,r1,SK2}

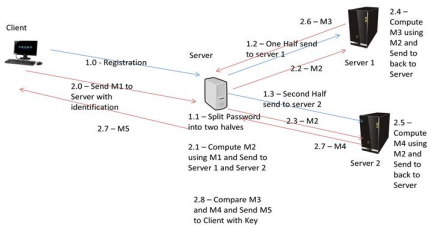
Step 4: Service server 1(SC1) received M2, compute M3 and send back to CS.

Step 5: Service server 2(SC2) received M2, computer M4 and send back to CS.

Step 6: CS compare M3 and M4 then send M5 to client with session key to exchange message.

The detailed authentication and key exchange of our protocol has been described as above. From Fig., we can see that the two service servers SS1 and SS2 equally contribute to the authentication and key exchange. Therefore, our protocol is asymmetric. We need to show the client has established the secret session keys with the two servers, respectively.

III. OUR MODEL



IV. CONCLUSION

In this paper, we have presented a asymmetric protocol for distributed -server, password-only authentication and key exchange. Security analysis has shown that our protocol is secure against passive and active attacks in case that one of the two servers is compromised. Performance analysis has shown that our protocol is more efficient than existing symmetric and asymmetric two-server PAKE protocols.

REFERENCES

[1] M. Abdalla and D. Pointcheval, "Simple Password-Based Encrypted Key Exchange Protocols," Proc. Int'l Conf. Topics in Cryptology (CT-RSA), pp. 191-208, 2005.

[2] M. Abdalla, O. Chevassut, and D. Pointcheval, "One-Time Verifier-Based Encrypted Key Exchange," Proc. Eighth Int'l Conf. Theory and Practice in Public Key Cryptography (PKC '05), pp. 47-64, 2005.

[3] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated Key Exchange Secure against Dictionary Attacks," Proc. 19th Int'l Conf. Theory and Application of Cryptographic Techniques (Eurocrypt '00), pp. 139-155, 2000.

[4] S. Bellovin and M. Merritt, "Encrypted Key Exchange: Password- Based Protocol Secure against Dictionary Attack," Proc. IEEE Symp. Research in Security and Privacy, pp. 72-84, 1992.

[5] D. Boneh and M. Franklin, "Identity Based Encryption from the Weil Pairing," Proc. 21st Ann. Int'l Cryptology Conf. (Crypto '01), pp. 213-229, 2001.

[6] D. Boneh and M. Franklin, "Identity Based Encryption from the Weil Pairing," SIAM J. Computing, vol. 32, no. 3, pp. 586-615, 2003.

[7] D. Boneh, "The Decisional Diffie-Hellman Problem," Proc. Third Int'l Algorithmic Number Theory Symp., pp. 241-250, 1998.

[8] V. Boyko, P. Mackenzie, and S. Patel, "Provably Secure Password- Authenticated Key Exchange Using Diffie-Hellman," Proc. 19th Int'l Conf. Theory and Application of Cryptographic Techniques (Eurocrypt '00), pp. 156-171, 2000.

[9] J. Brainard, A. Jueles, B.S. Kaliski, and M. Szydlo, "A New Two- Server Approach for Authentication with Short Secret," Proc. 12th Conf. USENIX Security Symp., pp. 201-214, 2003.

[10] W. Diffie and M.E. Hellman, "New Directions in Cryptography," IEEE Trans. Information Theory, IT-22, no. 6, pp. 644-654, Nov. 1976.

[11] M. Di Raimondo and R. Gennaro, "Provably Secure Threshold Password Authenticated Key Exchange," Proc. 22nd Int'l Conf. Theory and Applications of Cryptographic Techniques (Eurocrypt '03), pp. 507-523, 2003.

[12] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," IEEE Trans. Information Theory, vol. IT-31, no. 4, pp. 469-472, July 1985.

[13] W. Ford and B.S. Kaliski Jr., "Server-Assisted Generation of a Strong Secret from a Password," Proc. IEEE Ninth Int'l Workshop Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 176-180, 2000.

[14] O. Goldreich and Y. Lindell, "Session-Key Generation using Human Passwords Only," Proc. 21st Ann. Int'l Cryptology Conf. Advances in Cryptology (Crypto '01), pp. 408-432, 2001.

[15] L. Gong, T.M.A. Lomas, R.M. Needham, and J.H. Saltzer, "Protecting Poorly-Chosen Secret from Guessing Attacks," IEEE J. Selected Areas in Comm., vol. 11, no. 5, pp. 648-656, June 1993.

[16] S. Halevi and H. Krawczyk, "Public-Key Cryptography and Password Protocols," ACM Trans. Information and System Security, vol. 2, no. 3, pp. 230-268, 1999.

[17] D. Jablon, "Password Authentication Using Multiple Servers," Proc. Conf. Topics in Cryptology: The Cryptographer's Track at RSA (RSA-CT '01), pp. 344-360, 2001.

[18] H. Jin, D.S. Wong, and Y. Xu, "An Efficient Password-Only Two- Server Authenticated Key Exchange System," Proc. Ninth Int'l Conf. Information and Comm. Security (ICICS '07), pp. 44-56, 2007.

[19] J. Katz, R. Ostrovsky, and M. Yung, "Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords," Proc. Int'l Conf. Theory and Application of Cryptographic Techniques: Advances in Cryptology (Eurocrypt '01), pp. 457-494, 2001.