Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER Applied Mathematics and Computation 167 (2005) 996–1003
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

Efficient verifier-based key agreement protocol for three parties without server's public key

Sung-Woon Lee [a], Hyun-Sung Kim [b], Kee-Young Yoo [a,*]

[a] *Department of Computer Engineering, Kyungpook National University, Daegu 702-701, Republic of Korea*
[b] *Department of Computer Engineering, Kyungil University, Kyungsansi, Kyungpook Province 712-701, Republic of Korea*

Abstract

So far, there have been several key agreement protocols for three-parties, in which two clients establish a common session key through a authentication server. Most of those protocols require to use server's public key to prevent password guessing attacks. However, because clients need to verify and safely keep the server's public key, the protocols may not be practical for some environments. This paper proposes a new efficient verifier-based key agreement protocol for three parties which does not require server's public key. The proposed protocol is resistant against various attacks and provides the perfect forward secrecy.

Keywords: Cryptography; Authentication; Password; Key agreement; Key exchange

* Corresponding author.
E-mail address: yook@knu.ac.kr (K.-Y. Yoo).

0096-3003/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2004.06.129

1. Introduction

When the communicating parties initiate a connection, it is necessary to verify the identity of the other party. This authentication is usually provided in combination with a key agreement protocol between the parties. The password-based authentication is the most widely used method due to the advantages of simplicity, convenience, adaptability, mobility, and less hardware requirement. It requires users only to remember their knowledge such as a password. However, traditional password-based protocols are susceptible to password guessing attacks (called dictionary attacks) since many users tend to choose memorable passwords of relatively low entropy. Since Bellovin and Merrit [1] presented an encrypted exchange protocol called EKE for password-based authentication and key agreement which was resistant to password guessing attacks, many password-based authenticated key agreement protocols for two parties have been proposed.

Password-based authenticated key agreement protocols are divided into two classes in IEEE Std1363a-2002 [2]. The first is the balanced password-based authenticated key agreement scheme [4,6–8,10], in which two parties use a shared password to negotiate one or more shared ephemeral keys such that the shared keys are established if and only if they use the same password. The second is the augmented password-based authenticated key agreement scheme [10] (usually called verifier-based protocol), in which two parties (denoted Client and Server) use related password-based values to negotiate one or more shared ephemeral keys such that the shared keys are established if and only if they use values that correspond to the same password. Server uses password verification data (usually called verifier) that is derived from client's password data. The schemes force an attacker who steals the password verification data to further perform a successful brute-force attack in order to masquerading as client.

In 1995, Steiner et al. [4] proposed a three-party EKE protocol, called STW-3PEKE, without using server's public key based on the EKE protocol, in which each client shares a memorable password with an authentication server which is responsible for authenticating two clients and helps the clients to agree a common session key. However, Ding and Horster [5] showed that the protocol is vulnerable to undetectable on-line password guessing attacks. In 2000, Lin et al. [6] pointed out that the protocol is also vulnerable to off-line password guessing attacks, and then proposed an improved protocol, called LSH-3PEKE, using server's public key to resist to the attacks. In 2001, Lin et al. [7] proposed a new encrypted key exchange protocol for three parties, called LSSH-3PEKE, which is resistant to password guessing attacks and does not require server's public key. However, LSSH-3PEKE needs two additional rounds more than LSH-3PEKE. Recently, Chang and Chang [8] proposed an improved protocol to reduce two rounds in LSSH-3PEKE by employing

super-poly-to-one trapdoor functions [9]. On the other hand, Sun et al. [10] proposed an improved protocol to prevent password guessing attacks in STW-3PEKE and further proposed a verifier-based key agreement protocol for three-parties, which require server's public key. The method using server's public key to prevent password guessing attacks puts a burden on the clients because the clients have to obtain, verify, and keep safely the public key of the server.

In this paper, we propose a new efficient verifier-based key agreement protocol for three parties without server's public key. The proposed protocol is resistant against various attacks and provides the perfect forward secrecy. Moreover, because the proposed protocol does not require server's public key and requires each client only to remember a memorable password, it is quite suitable for applications in which light-weight clients needs secure communication.

The remainder of this paper is organized as follows. In Section 2, we describe the proposed protocol. In Section 3, we show security analysis of our protocol. In Section 4, we analyze of the efficiency of the proposed protocol. Finally, Section 5 gives our conclusions.

2. Efficient verifier-based key agreement protocol for three-parties

In this section, we present an efficient verifier-based key agreement protocol for three parties which does not require server's public key.

In the proposed protocol, each client uses a memorable password, while the server does store verifiers instead of plaintext-equivalent passwords to resist to server compromise. A verifier v is the information computed from a password π. Let p be a large prime and g a generator in the cyclic group Z_p^*, in which Discrete Logarithm Problem and Diffie–Hellman problem are considered hard. $h(\cdot)$ means a collision-free one-way hash function.

Assume that two clients, called *Alice* and *Bob*, want to agree a common session key though a authentication server, called *AS*. *A*, *B*, and *S* indicate identifiers of *Alice*, *Bob*, and *AS*, respectively. For registering for *AS*, *Alice* and *Bob* respectively choose passwords π_A and π_B, compute verifiers $v_A = g^{h(A,S,\pi_A)}$ and $v_B = g^{h(B,S,\pi_B)}$, and then send v_A and v_B to *AS* over a secure channel. *AS* stores v_A and v_B in a password table. We will omit '$\bmod p$' from expressions for simplicity. The steps for the proposed protocol are as follows (Fig. 1):

1. *Alice* computes $X_A = g^a$ by choosing $a \in_R Z_p^*$ and then sends A and X_A to *Bob*.
2. After receiving the message from *Alice*, *Bob* computes $X_A = g^b$ by choosing $b \in_R Z_p^*$ and sends A, X_A, B, and X_B to *AS*. *Bob* also sends X_B to *Alice*.

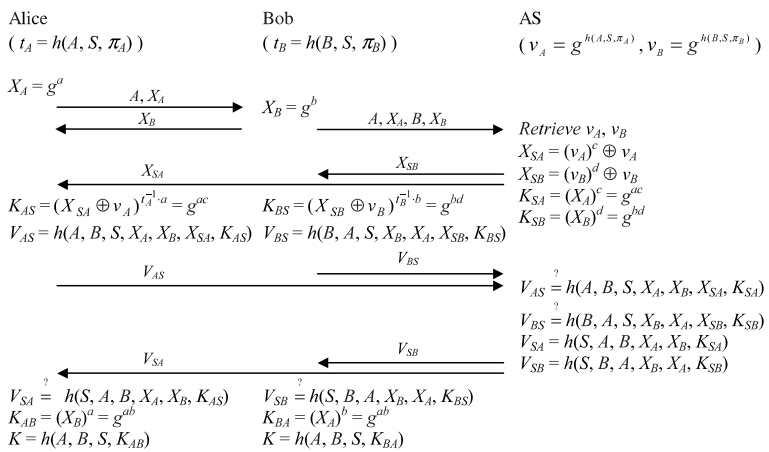

Fig. 1. The proposed protocol.

3. After receiving the message from *Bob*, *AS* retrieves v_A and v_B from a password table, computes $X_{SA} = (v_A)^c \oplus v_A$ and $X_{SB} = (v_B)^d \oplus v_B$ by choosing $c, d \in_R Z_p^*$, and sends X_{SA} and X_{SB} to *Alice* and *Bob*, respectively. While waiting for messages from *Alice* and *Bob*, *AS* computes $K_{SA} = (X_A)^c = g^{ac}$ and $K_{SB} = (X_B)^d = g^{bd}$.
4. After receiving the messages from *AS* and *Bob*, *Alice* computes $K_{AS} = (X_{SA} \oplus v_A)^{t_A^{-1} \cdot a} = g^{ac}$ and $V_{AS} = h(A, B, S, X_A, X_B, X_{SA}, K_{AS})$ and sends V_{AS} to *AS*. Similarly, after receiving the message from *AS*, *Bob* computes $K_{BS} = (X_{SB} \oplus v_B)^{t_B^{-1} \cdot b} = g^{bd}$ and $V_{BS} = h(B, A, S, X_B, X_A, X_{SB}, K_{BS})$ and sends V_{BS} to *AS*.
5. After receiving the messages from *Alice* and *Bob*, *AS* checks whether $V_{AS} \stackrel{?}{=} h(A, B, S, X_A, X_B, X_{SA}, K_{SA})$ and $V_{BS} \stackrel{?}{=} h(B, A, S, X_B, X_A, X_{SB}, K_{SB})$ hold or not. If they hold, *AS* is convinced that *Alice* and *Bob* are validated. Then, *AS* computes $V_{SA} = h(S, A, B, X_A, X_B, K_{SA})$ and $V_{SB} = h(S, B, A, X_B, X_A, K_{SB})$ and sends V_{SA} and V_{SB} to *Alice* and *Bob*, respectively.
6. After receiving the message from *AS*, *Alice* checks whether $V_{SA} \stackrel{?}{=} h(S, A, B, X_A, X_B, K_{AS})$ holds or not. If they hold, *Alice* is convinced that both *Bob* and *AS* are validated. Similarly, after receiving the message from *AS*, *Bob* checks whether $V_{SB} \stackrel{?}{=} h(S, B, A, X_B, X_A, K_{BS})$ holds or not. If they hold, *Bob* is convinced that both *Alice* and *AS* are validated. Finally, *Alice* and *Bob* compute $K_{AB} = (X_B)^a = g^{ab}$ and $K_{BA} = (X_A)^b = g^{ab}$, and then compute a common session key $K = h(A, B, S, K_{AB}) = h(A, B, S, K_{BA}) = h(A, B, S, g^{ab})$, respectively.

To enhance the efficiency of the protocol, t, t^{-1} and v can be pre-computed by each client before the protocol runs. The successive message transmissions without waiting time can be considered as one round. The proposed protocol

needs message exchanges of five rounds as follows: (1) $Alice \rightarrow Bob$: $\{A, X_A\}$, (2) $Bob \rightarrow AS$: $\{A, X_A, B, X_B\}$, $Bob \rightarrow Alice$: $\{X_B\}$, (3) $AS \rightarrow Alice$: $\{X_{SA}\}$, $AS \rightarrow Bob$: $\{X_{SB}\}$, (4) $Alice \rightarrow AS$: $\{V_{AS}\}$, $Bob \rightarrow AS$: $\{V_{BS}\}$, (5) $AS \rightarrow Alice$: $\{V_{SA}\}$, $AS \rightarrow Bob$: $\{V_{SB}\}$. The use of '$\oplus$' operations by each party can prevent off-line password guessing attacks. Through validating V_{AS} and V_{BS}, AS can confirm not only whether *Alice* and *Bob* are legal but also whether *Alice* and *Bob* correctly gave and took their own messages to the other. Similarly, through validating V_{SA} and V_{SB}, *Alice* and *Bob* cannot only authenticate both AS and the other but also know whether *Alice* and *Bob* correctly sent their own messages to the other, respectively.

3. Security analysis

First, suppose that all communication among the interacting parties is under the control of an attacker, called *Eve*. That is, *Eve* can read the messages produced by the parties, provide messages of her own to them, modify messages before they reach their destination, delay messages or replay them, and make new instances of any parties. The security of our protocol is based on the difficulty of the Discrete Logarithm Problem (DLP) and the Diffie–Hellman problem (DHP) which are believed infeasible to solve in polynomial time. We analyze the security of our protocols with regard to several attacks.

1. Assume that after capturing the transmitted messages, A, X_A, B, X_B, X_{SA}, X_{SB}, V_{AS}, V_{BS}, V_{SA}, and V_{SB}, *Eve* directly tries to compute the passwords or the session key of the clients from them. However, it is computationally infeasible due to the difficulty of DLP and DHP and the properties of one-way hash function.
2. Assume that *Eve* tries to masquerade *Alice* or *Bob*. However, AS can detect this attack when verifying $V_{AS} \stackrel{?}{=} h(A, B, S, X_A, X_B, X_{SA}, K_{SA})$ and $V_{BS} \stackrel{?}{=} h(B, A, S, X_B, X_A, X_{SB}, K_{SB})$, because *Eve* cannot compute the valid K_{SA} or K_{SB} due to not knowing their correct passwords.
3. Assume that *Eve* tries to masquerade AS. However, *Alice* and *Bob* can respectively detect this attack when verifying $V_{SA} \stackrel{?}{=} h(S, A, B, X_A, X_B, K_{AS})$ and $V_{SB} \stackrel{?}{=} h(S, B, A, X_B, X_A, K_{BS})$, because *Eve* cannot compute the valid K_{AS} and K_{BS} due to not knowing their correct verifiers.
4. Password guessing attacks succeed when there are pieces of information in communications that can be used to verify the correctness of the guessed password. After capturing the transmitted messages or capturing the transmitted messages after masquerading one or more parties, *Eve* guesses π'_A or π'_B and then tries to verify her guesses using the transmitted messages. However, she has no way of verifying her guesses because each party uses '$\oplus$' operation.

5. Perfect forward secrecy is provided in the situation that even though passwords are compromised, *Eve* cannot derive previous session keys. To consider this, suppose that *Eve* knows π_A or π_B. *Eve* tries to find previous session keys from the information collected in past communication sessions. However, it is infeasible due to the difficulty of DLP and DHP and the properties of one-way hash function.
6. For the protocol being secure against server compromise means an attacker not being able to pose as a client after the server is compromised. In the proposed protocol, if AS is compromised, *Eve* may knows two clients' verifiers $v_A = g^{h(A,S,\pi_A)}$ and $v_B = g^{h(B,S,\pi_B)}$. However, she cannot pose as the clients because of not knowing $t_A = h(A, S, \pi_A)$ and $t_B = h(B, S, \pi_B)$ being used in step 4. Therefore, the proposed protocol is secure against server compromise.

4. Efficiency analysis

Performance of key agreement protocols can be approximated in terms of communication and computation loads. The proposed protocol is compared with one related verifier-based protocol, SCH-3PEKE. Table 1 shows the comparison regarding with several efficiency factors such as the number of rounds, random numbers, exponentiations, asymmetric encryption/decryption, symmetric encryption/decryption, and hash functions.

As shown in Table 1, the proposed protocol is similar with SCH-3PEKE in terms of most of the factors. Usually, it is considered that the cost of modular exponentiation and the cost of asymmetric encryption/decryption are similar. However, total execution time of the proposed protocol is very small. For the measure of the total execution time, we will consider only modular exponentiations, which are time-consuming operations. Suppose that E(A:B:S)

Table 1
Comparison with the related verifier-based protocol

Protocol		Factor						
		Random number	Exp.	Asym. enc./dec.	Sym. enc./dec.	Hash function	Round	E(A:B:S)
The proposed protocol	A	1	3	0	0	3	5	5E
	B	1	3	0	0	3		
	S	2	4	0	0	4		
SCH-3PEKE	A	1	2	1	2	0	5	9E
	B	1	2	1	2	0		
	S	3	2.5	2	0	0		

means parallel execution for modular exponentiations between three parties. In the proposed protocol, *Alice* and *Bob* can respectively pre-compute X_A and X_B and compute K_{AS} and K_{BS} at the same time. It is reasonable that AS can compute both K_{SA} and K_{SB} while *Alice* and *Bob* respectively compute K_{AB} and K_{BA} because of the communication time and the computation time. Therefore, the proposed protocol needs only 5E, that is, E(X_A:X_B:–), E(–:–:X_{SA}), E(–:–:X_{SB}), E(K_{SA}:K_{SB}:K_{SA}, K_{SB}), E(K:K:–). Here, '–' means no exponentiation. As compared with this, SCH-3PEKE needs 9E.

SCH-3PEKE needs to get and validate server's public-key and an additional storage device to safely save and keep server's public-key, while the proposed protocol requires clients only to remember their own passwords. Thus, the proposed protocol provides another solution to SCH-3PEKE. That is, our protocol is useful for environments where clients cannot be expected to carry around or correctly validate the server's public key.

5. Conclusion

In this paper, we have proposed a new verifier-based key agreement protocol for three-parties, which does not require server's public key but requires each client only to remember a memorable password. The proposed protocol is secure to various attacks and provides the perfect forward secrecy. Moreover, it has very good efficiency in terms of total execution time as compared with the related verifier-based protocol SCH-3PEKE. In particular, because it is implemented without the server's public key, it is useful in an environment where clients cannot correctly validate the server's public key.

Acknowledgments

This work was supported by the Brain Korea 21 Project in 2003 and partially by University IT Research Center Project.

References

[1] S.M. Bellovin, M. Merrit, Encrypted key exchange: password-based protocols secure against dictionary attacks, in: Proceedings of the IEEE Symposium on Research in Security and Privacy, 1992, pp. 72–84.
[2] IEEE, Standard specifications for public key cryptography, IEEE1363, 2002.
[4] M. Steiner, G. Tsudik, M. Waidner, Refinement and extension of encrypted key exchange, ACM Operating Systems Review 29 (3) (1995) 22–30.
[5] Y. Ding, P. Horster, Undetectable on-line password guessing attacks, ACM Operating Systems Review 29 (4) (1995) 77–86.

[6] C.-L. Lin, H.-M. Sun, T. Hwang, Three-party encrypted key exchange: attacks and a solution, ACM Operating Systems Review 34 (4) (2000) 12–20.
[7] C.-L. Lin, H.-M. Sun, M. Steiner, T. Hwang, Three-party encrypted key exchange without server public-keys, IEEE Communication Letters 5 (12) (2001) 497–499.
[8] C.-C. Chang, Y.-F. Chang, A novel three-party encrypted key exchange protocol, Computer Standards and Interfaces 26 (5) (2004) 471–476.
[9] R. Impagaliazzo, S. Rudich, Limits on the provable consequences of one-way permutations, in: Proceedings of the 21st ACM Symposium on the Theory of Computing, 1989, pp. 44–61.
[10] H.-M. Sun, B.-C. Chen, T. Hwang, Secure key agreement protocols for three-party against guessing attacks, The Journal of Systems and Software, in press.