Enhancing Security of a Group Key Exchange Protocol for Users with Individual Passwords

Junghyun Nam†

May 6, 2007

Abstract

Group key exchange protocols allow a group of parties communicating over a public network to come up with a common secret key called a *session key*. Due to their critical role in building secure multicast channels, a number of group key exchange protocols have been suggested over the years for a variety of settings. Among these is the so-called EKE-M protocol proposed by Byun and Lee for password-based group key exchange in the *different password authentication model*, where group members are assumed to hold an individual password rather than a common password. While the announcement of the EKE-M protocol was essential in the light of the practical significance of the different password authentication model, Tang and Chen showed that the EKE-M protocol itself suffers from an undetectable on-line dictionary attack. Given Tang and Chen's attack, Byun et al. have recently suggested a modification to the EKE-M protocol and claimed that their modification makes EKE-M resistant to the attack. However, the claim turned out to be untrue. In the current paper, we demonstrate this by showing that Byun et al.'s modified EKE-M is still vulnerable to an undetectable on-line dictionary attack. Besides reporting our attack, we also figure out what has gone wrong with Byun et al.'s modification and how to fix it.

Keywords: Group key exchange, password-based authentication, undetectable on-line dictionary attack.

1 Introduction

The highest priority in designing a key exchange protocol is placed on ensuring the security of session keys to be established by the protocol. Roughly speaking, establishing a session key securely means that the key is being known only to the intended parties at the end of the protocol run. Even if it is computationally infeasible to break the cryptographic algorithms used, the whole system becomes vulnerable to all

†Department of Computer Science, Konkuk University, 322 Danwol-dong, Chungju-si, Chungcheongbuk-do 380-701, Republic of Korea.
E-mail: `jhnam@kku.ac.kr`

manner of attacks if the keys are not securely established. But unfortunately, the experience has shown that the design of secure key exchange protocols is notoriously difficult. In particular, the difficulty is greatly increased in the group setting where a session key is to be established among an arbitrary number of parties. Indeed, there is a long history of protocols for this domain being proposed and years later found to be flawed (a very partial list of examples includes [14, 17, 7, 20, 16, 12, 1]). Thus, group key exchange protocols must be subjected to a thorough and systematic scrutiny before they are deployed into a public network, which might be controlled by an adversary.

Secure session-key generation requires an authentication mechanism to be integrated into key exchange protocols. In turn, achieving any form of authentication inevitably requires some secret information to be established between users in advance of the authentication stage. Cryptographic keys, either secret keys for symmetric cryptography or private/public keys for asymmetric cryptography, may be one form of the underlying secret information pre-established between users. However, these high-entropy cryptographic keys are random in appearance and thus are difficult for humans to remember, entailing a significant amount of administrative work and costs. Eventually, it is this drawback that password-based authentication has come to be widely used in reality. Passwords are drawn from a relatively small space like a dictionary, and are easier for humans to remember than cryptographic keys with high entropy.

In the past few years there have been several protocols proposed for password-authenticated group key exchange. However, most of the protocols have been built in the so-called *same password authentication model* which assumes a common password pre-established among all users participating in the protocol (e.g., [8, 15, 1, 19, 13, 6, 2]). Hence, these protocols may be inadequate for many client-server applications in which each user (called client) shares its password only with the server, but not with other users.

Given the situation above, Byun and Lee [10] have recently proposed two new protocols, called EKE-U and EKE-M, for password-authenticated group key exchange in the *different password authentication model* where each user is assumed to hold an individual password rather than a common password. But later, Tang and Chen [18] showed that both EKE-U and EKE-M are not as much secure as originally claimed, suffering from an off-line dictionary attack and an undetectable on-line dictionary attack, respectively. Generating a sequence of attack and defence moves, Byun et al. [11] suggested countermeasures against Tang and Chen's attacks on EKE-U and EKE-M. However, we found that Byun et al.'s countermeasure for protecting EKE-M against the undetectable on-line dictionary attack is not satisfactory enough. Extending the attack-defence sequence, this paper reports a security defect of the countermeasure and presents how to remedy the security defect.

2 The EKE-M Protocol and Its Weakness

This section reviews the EKE-M protocol presented by Byun and Lee [10] and its weakness pointed out by Tang and Chen [18].

2.1 Description of EKE-M

The EKE-M protocol is designed for use in a multicast network. The protocol participants consist of a single server S and multiple clients $C_1, \ldots, C_n$. The protocol assumes that each client C_i has shared a password pw_i with the server S via a secure channel. The followings are the public system parameters used in the protocol.

1. A cyclic group $\mathbb{G}$ of prime order q and a generator g of $\mathbb{G}$.
2. A pair of symmetric encryption/decryption algorithms $(\mathcal{E}, \mathcal{D})$ modeled as an ideal cipher [3].
3. Two one-way hash functions $\mathcal{H}_1$ and $\mathcal{H}_2$ modeled as random oracles [5].

With the passwords and the parameters established, the EKE-M protocol runs in two communication rounds as follows:

Round 1: The server S chooses n random numbers $s_1, \ldots, s_n \in \mathbb{Z}_q^*$, computes $t_1 = \mathcal{E}_{pw_1}(g^{s_1}), \ldots, t_n = \mathcal{E}_{pw_n}(g^{s_n})$, and sends t_i to the client C_i for $i = 1, \ldots, n$. Concurrently, each C_i selects a random $x_i \in \mathbb{Z}_q^*$, computes $y_i = \mathcal{E}_{pw_i}(g^{x_i})$, and broadcasts y_i to the rest of the group. S and C_i decrypt respectively y_i and t_i using pw_i, and share the pairwise key $sk_i = \mathcal{H}_1(sid \| g^{s_i x_i})$ where $sid = y_1 \| y_2 \| \cdots \| y_n$.

Round 2: S selects a random group secret $K \in \mathbb{Z}_q^*$ and computes $k_1 = K \oplus sk_1, \ldots, k_n = K \oplus sk_n$. Then S broadcasts all the k_i's to the clients. After receiving the broadcast message, each C_i computes the group secret $K = k_i \oplus sk_i$ and the session key $sk = \mathcal{H}_2(SID \| K)$ where $SID = sid \| k_1 \| k_2 \| \cdots \| k_n$.

If key confirmation is required, the EKE-M protocol can be extended to incorporate the well-known technique [9] which in turn is based on earlier work of [4, 3].

2.2 Attack on EKE-M

As mentioned in the Introduction, Tang and Chen [18] presented an undetectable on-line dictionary attack on the EKE-M protocol. The attack can be mounted by any registered client C_j against any other registered client C_i $(1 \le i \le n, i \ne j)$. Through the attack, the adversary C_j can undetectably verify the correctness of its guess for the password of each victim C_i. Seriously, the repeated mounting of the attack could lead to the exposure of the real passwords of the victims. Seen from a high level, the attack scenario is quite clear: the adversary C_j simply runs the protocol with the server S while playing multiple roles of the clients $C_1, \ldots, C_n$. A detailed description of the attack follows.

1. For client C_i $(1 \le i \le n, i \ne j)$, the adversary C_j makes a guess pw'_i for the password pw_i, selects a random $x_i \in \mathbb{Z}_q^*$, and computes $y_i = \mathcal{E}_{pw'_i}(g^{x_i})$. In addition, C_j also computes y_j as specified in the protocol.

2. When the server S sends $t_1, \ldots, t_n$ to the clients in the first round, C_j intercepts these t_i's and sends y_j (with its true identity) and each y_i (posing as C_i) to S. Then, C_j computes the pairwise key $sk_j = \mathcal{H}_1(sid \| g^{s_j x_j})$.

3. Now, when S broadcasts $k_1, \ldots, k_n$ in the second round, C_j intercepts these k_i's and recovers K by computing $K = k_j \oplus sk_j$. Finally, C_j verifies the correctness of each guess pw'_i by computing $sk'_i = \mathcal{H}_1(sid \| (\mathcal{D}_{pw'_i}(t_i))^{x_i})$ and by checking the equality $K \stackrel{?}{=} k_i \oplus sk'_i$.

It is clear that the vulnerability of the EKE-M protocol to the attack above is mainly because the server does not require the clients to authenticate themselves in the protocol execution.

3 Byun et al.'s Modification to EKE-M

We here describe the modified EKE-M due to Byun et al. [11], which we call EKE-M$^+$. The EKE-M$^+$ protocol aims to be secure against undetectable on-line password guessing attacks. The first round of EKE-M$^+$ proceeds exactly like that of EKE-M while the second round is extended to let S and C_i exchange the authenticators $\mathcal{H}_2(sk_i \| S)$ and $\mathcal{H}_2(sk_i \| C_i)$. In more detail the EKE-M$^+$ protocol works as follows:

Round 1: S selects random numbers $s_1, \ldots, s_n \in \mathbb{Z}_q^*$, computes $t_1 = \mathcal{E}_{pw_1}(g^{s_1})$, ..., $t_n = \mathcal{E}_{pw_n}(g^{s_n})$, and sends t_i to C_i for $i = 1, \ldots, n$. Concurrently, each C_i selects a random $x_i \in \mathbb{Z}_q^*$, computes $y_i = \mathcal{E}_{pw_i}(g^{x_i})$, and broadcasts y_i to the rest of the group. S and C_i decrypt respectively y_i and t_i using pw_i, and share the pairwise key $sk_i = \mathcal{H}_1(sid \| g^{s_i x_i})$ where $sid = y_1 \| y_2 \| \cdots \| y_n$.

Round 2: S selects a random group secret $K \in \mathbb{Z}_q^*$ and computes $k_i = K \oplus sk_i$ and $\alpha_i = \mathcal{H}_2(sk_i \| S)$ for $i = 1, \ldots, n$. Then S broadcasts all the k_i's and α_i's to the clients. Concurrently, each C_i computes and broadcasts $\beta_i = \mathcal{H}_2(sk_i \| C_i)$. S verifies the correctness of each β_i to detect any potential on-line dictionary attack. Meanwhile, C_i verifies the correctness of α_i and only if the verification succeeds, proceeds to compute the group secret $K = k_i \oplus sk_i$ and the session key $sk = \mathcal{H}_2(SID \| K)$ where $SID = sid \| k_1 \| k_2 \| \cdots \| k_n$.

Like EKE-M, the EKE-M$^+$ protocol can be extended to include the well-known technique for key confirmation [9].

4 Attack on EKE-M$^+$

Byun et al. [11] claim that their EKE-M$^+$ protocol is secure against undetectable on-line dictionary attacks. In support of this claim, they argue that the malicious client C_j cannot generate the authenticator $\beta_i = \mathcal{H}_2(sk_i \| C_i)$ because it does not know the pairwise key sk_i. However, this claim is flawed. The fact is that C_j is easily able to compute sk_i and so is able to generate β_i. A direct consequence of this fact is that unlike the claim, the EKE-M$^+$ protocol is still vulnerable to an undetectable on-line dictionary attack, as shown below. Our observation leading to the attack is that computing sk_i in EKE-M$^+$ does not necessarily require the knowledge of the correct password pw_i. The attack proceeds as follows:

1. The adversary C_j begins by preparing the messages to be sent to S in the first round. For each y_i $(1 \le i \le n, i \ne j)$, C_j computes them as $y_i = \mathcal{E}_{pw'_i}(g^{x_i})$ where pw'_i is a guess for the password pw_i and x_i is a random number from $\mathbb{Z}_q^*$. For its own y_j, C_j computes it exactly as specified in the protocol.

2. When the server S sends $t_1, \ldots, t_n$ to the clients in the first round, C_j intercepts these t_i's and sends y_j (with its true identity) and each y_i (posing as C_i) to S. Then, C_j computes the pairwise key sk_j and the authenticator β_j as per the protocol specification.

3. Now, when S broadcasts k_i's and α_i's in the second round, C_j intercepts the broadcast message and recovers K by computing $K = k_j \oplus sk_j$. Then C_j computes $sk_i = k_i \oplus K$ and $\beta_i = \mathcal{H}_2(sk_i \| C_i)$ for $i \in \{1, \ldots, n\} \setminus \{j\}$, and sends β_j and each β_i immediately to S. Finally, C_j verifies the correctness of each guess pw'_i by computing $sk'_i = \mathcal{H}_1(sid \| (\mathcal{D}_{pw'_i}(t_i))^{x_i})$ and by checking the equality $K \stackrel{?}{=} k_i \oplus sk'_i$.

5 Enhancing Security of EKE-M$^+$

One intuitive way of preventing the attack above is to modify the EKE-M$^+$ protocol so that the server S broadcasts k_i's and α_i's only after it receives and verifies the authenticators β_i's. With this modification, the attack would be no longer possible because the adversary could not compute β_i without having received k_i from S. However, this solution might be not so elegant in that it comes at an additional communication round.

A better way to fix the EKE-M$^+$ protocol can be found by figuring out the fundamental cause of the security failure. A little thought will make it clear that the main design flaw in EKE-M$^+$ is to use the same sk_i in computing both $k_i = K \oplus sk_i$ and $\beta_i = \mathcal{H}_2(sk_i \| C_i)$. This oversight allows the adversary to derive β_i easily from K and k_i, and thus creates the vulnerability to the undetectable on-line dictionary attack.

Having identified the source of the problem, it is now apparent how to repair the protocol. The computations of k_i and β_i should be modified so that one of the two cannot be derived from the other. To this end, it suffices to change the computation of β_i to $\beta_i = \mathcal{H}_2(\overline{sk}_i \| C_i)$ where $\overline{sk}_i \stackrel{\text{def}}{=} \mathcal{H}_1(g^{s_i x_i} \| C_1 \| \cdots \| C_n)$. The verification of β_i changes correspondingly, but all other computations remain unchanged. This modification effectively prevents the undetectable on-line dictionary attack because the adversary C_j can no longer generate β_i even with sk_i at hand. As for efficiency, the modification does not increase the number of communication rounds but only takes an additional evaluation of a hash function.

References

[1] M. Abdalla, E. Bresson, O. Chevassut, and D. Pointcheval, "Password-based group key exchange in a constant number of rounds," in *Proc. 9th International Workshop on Practice and Theory in Public Key Cryptography*, LNCS vol. 3958, pp. 427–442, 2006.

[2] M. Abdalla and D. Pointcheval, "A scalable password-based group key exchange protocol in the standard model," in *Proc. Asiacrypt '06*, LNCS vol. 4284, pp. 332–347, 2006.

[3] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Proc. Eurocrypt '00*, LNCS vol. 1807, pp. 139–155, 2000.

[4] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Proc. Crypto '93*, LNCS vol. 773, pp. 232–249, 1993.

[5] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proc. 1st ACM Conference on Computer and Communications Security*, pp. 62–73, 1993.

[6] J.-M. Bohli, M. I. G. Vasco, and R. Steinwandt, "Password-authenticated constant-round group key establishment with a common reference string," *Cryptology ePrint Archive*, Report 2006/214, 2006.

[7] C. Boyd and J. M. G. Nieto, "Round-optimal contributory conference key agreement," in *Proc. 6th International Workshop on Practice and Theory in Public Key Cryptography*, LNCS vol. 2567, pp. 161–174, 2003.

[8] E. Bresson, O. Chevassut, and D. Pointcheval, "Group Diffie-Hellman key exchange secure against dictionary attacks," in *Proc. Asiacrypt '02*, LNCS vol. 2501, pp. 497–514, 2002.

[9] E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater, "Provably authenticated group Diffie-Hellman key exchange," in *Proc. 8th ACM Conference on Computer and Communications Security*, pp. 255–264, 2001.

[10] J. Byun and D. Lee, "N-Party encrypted Diffie-Hellman key exchange using different passwords," in *Proc. 3rd International Conference on Applied Cryptography and Network Security*, LNCS vol. 3531, pp. 75–90, 2005.

[11] J. Byun, D. Lee, and J. Lim, "Password-based group key exchange secure against insider guessing attacks," in *Proc. 2005 International Conference on Computational Intelligence and Security*, LNAI vol. 3802, pp. 143–148, 2005.

[12] K.-K. Choo, C. Boyd, and Y. Hitchcock, "Errors in computational complexity proofs for protocols," in *Proc. Asiacrypt '05*, LNCS vol. 3788, pp. 624–643, 2005.

[13] R. Dutta and R. Barua, "Password-based encrypted group key agreement," *International Journal of Network Security*, vol. 3, no. 1, pp. 23–34, 2006.

[14] M. Just and S. Vaudenay, "Authenticated multi-party key agreement," in *Proc. Asiacrypt '96*, LNCS vol. 1163, pp. 36–49, 1996.

[15] S.-M. Lee, J. Y. Hwang, and D. H. Lee, "Efficient password-based group key exchange," in *Proc. 1st International Conference on Trust and Privacy in Digital Business*, LNCS vol. 3184, pp. 191–199, 2004.

[16] J. Nam, S. Kim, and D. Won, "A weakness in the Bresson-Chevassut-Essiari-Pointcheval's group key agreement scheme for low-power mobile devices," *IEEE Communications Letters*, vol. 9, no. 5, pp. 429–431, 2005.

[17] O. Pereira and J.-J. Quisquater, "A security analysis of the Cliques protocols suites," in *Proc. 14th IEEE Computer Security Foundations Workshop*, pp. 73–81, 2001.

[18] Q. Tang and L. Chen, "Weaknesses in two group Diffie-Hellman key exchange protocols," *Cryptology ePrint Archive*, Report 2005/197, 2005. Available at `http://eprint.iacr.org/`.

[19] Q. Tang and K.-K. Choo, "Secure password-based authenticated group key agreement for data-sharing peer-to-peer networks," in *Proc. 4th International Conference on Applied Cryptography and Network Security*, LNCS vol. 3989, pp. 162–177, 2006.

[20] F. Zhang and X. Chen, "Attack on an ID-based authenticated group key agreement scheme from PKC 2004," *Information Processing Letters*, vol. 91, no. 4, pp. 191–193, 2004.