Cryptanalysis of the N-Party Encrypted Diffie-Hellman Key Exchange Using Different Passwords

Raphael C.-W. Phan[1] and Bok-Min Goi[2,⋆]

[1] Information Security Research (iSECURES) Lab,
Swinburne University of Technology (Sarawak Campus), 93576 Kuching, Malaysia
rphan@swinburne.edu.my
[2] Centre for Cryptography & Information Security (CCIS),
Faculty of Engineering, Multimedia University, 63100 Cyberjaya, Malaysia
bmgoi@mmu.edu.my

Abstract. We consider the security of the n-party EKE-U and EKE-M protocols proposed by Byun and Lee at ACNS '05. We show that EKE-U is vulnerable to an impersonation attack, offline dictionary attack and undetectable online dictionary attack. Surprisingly, even the strengthened variant recently proposed by the same designers to counter an insider offline dictionary attack by Tang and Chen, is equally vulnerable. We also show that both the original and strengthened EKE-M variants do not provide key privacy, a criterion desired by truly contributory key exchange schemes and recently formalized by Abdalla *et al.* We discuss ways to protect EKE-U against our attacks and argue that the strengthened EKE-U scheme shows the most potential as a provably secure n-party PAKE.

Keywords: Password-authenticated key exchange, n-party, cryptanalysis, dictionary attack, collusion, key privacy.

1 Introduction

Password authenticated key exchange (PAKE) protocols [1, 5, 7, 8, 13, 16, 17, 20] enable two or more parties to share a common secret key for securing (via secret-key cryptography) subsequent communications among them. For systems that depend on human interactions, using a password is more practical than a high-entropy secret key since the former is easier for a human to memorize by heart rather than be tempted to write it down somewhere [13].

One of the first PAKEs was the Encrypted Key Exchange (EKE) due to Bellovin and Merritt [5] for establishing a secret key between 2 parties. This was later extended to the 3-party case by Steiner *et al.* [20]. Further analysis and variants of the latter are found in [11, 16, 17, 1].

⋆ The second author acknowledges the Malaysia IRPA grant (04-99-01-00003-EAR).

J. Zhou, M. Yung, and F. Bao (Eds.): ACNS 2006, LNCS 3989, pp. 226–238, 2006.

In extending from a 2-party PAKE to a 3-party one, the basic question raised is how the parties will share the password. Consequently, we can classify group-based (involving more than 2 parties) PAKEs into two broad types [7], namely those that use a single shared password among all parties (SPWA) [6] and those where each party shares a distinct password with a trusted server (DPWA) [20, 16, 17, 1].

DPWA-type PAKEs allow trust to be partitioned among all clients such that in the event of any client being compromised or corrupted, it will not affect the security of the entire group; e.g. only secrets (session keys shared with him, and his password) known to the affected client need to be changed, but other innocent clients can continue using their existing passwords. This also means that less trust needs to be put on each individual client since the compromise of any client is less devastating to the security of the group. In contrast, a compromise of any client in an SPWA-type PAKE would require that the password shared by all clients be updated and re-communicated to each of them. Further, DPWA-type PAKEs are very much suited for mobile and distributed computing networks which are increasingly becoming prevalent, where the parties (clients) come from diverse environments thus are less understood. Under such circumstances, one would not want to put too much trust on any client.

Abdalla *et al.* [1] presented a formal security model for 3-party DWPA-type PAKEs by combining the Bellare *et al.* model [3] for 2-party PAKEs with the Bellare-Rogaway model [4] for 3-party key distribution schemes generalized to the password case. They also formally defined the notion of *key privacy* to differentiate truly contributory key exchange protocols from key distribution protocols. This notion, first mentioned in [20], roughly means that even though a third-party server's help is required to establish a session key between two clients, the server is not able to obtain any information on the value of that established session key. The goal of key privacy is to limit the amount of trust put into the server, where it is assumed that the server is honest but curious [1], thus clients prefer to have their established session key known only to themselves. This appropriately models real-life situations where privacy of secret information is well guarded by individuals. In fact, some other work in related information security fields are also moving in this direction, e.g. protocols proposed without the use of trusted third parties (TTP) in [9, 22], and research showing the subtlety of putting too much trust on TTPs [19, 12]. To achieve key privacy, it is necessary [1] to have a 2-party authenticated key exchange (AKE) between the two clients.

In this paper, we are concerned with DPWA schemes for the *n-party* case. More specifically, at ACNS '05 Byun and Lee [7] presented two variants of an n-party EKE protocol, respectively called n-party EKE-U and EKE-M for unicast and multicast networks. These appear to be the first known *n-party* EKE protocols with provable security. Tang and Chen [21] subsequently showed that EKE-U is vulnerable to an offline dictionary attack, and that EKE-M is vulnerable to an undetectable online dictionary attack [11]. Byun and Lee [8] promptly countered with strengthened variants, which we will also discuss in Sections 3 and 4.

Although PAKEs have been extensively studied especially in the last few years [1,3,5,6,7,8,13,16,17,20], most of them consider either the 2-party or 3-party case. And it was only very recently that the first provably secure PAKEs for the 3-party and n-party cases were presented in [1] and [7] respectively. Thus this field (that of provably secure group PAKEs) has potentially unexplored areas of future work, e.g. how to extend the existing provably secure 2-party or 3-party PAKEs to the n-party case in an efficient yet secure manner, i.e. without involving too many inter-client communications that would cause a bottleneck to the network especially when n is large.

We show attacks on both the original and strengthened EKE-U that exploit the server as an oracle to generate messages supposedly from an innocent client. Meanwhile for both the original and strengthened EKE-M, we point out that they do not achieve the key privacy property that is desired of contributory key exchange protocols.

In our concluding section, we discuss how to improve the *strengthened* EKE-U to resist our attacks and argue that it is a worthwhile candidate for a provably secure n-party PAKE.

2 The N-Party EKE Protocols

The n-party EKE protocols due to Byun and Lee [7] involve $n-1$ clients and 1 server, and are specially designed to suit modern communication environments such as ad-hoc networks and ubiquitous computing, in particular EKE-U for unicast networks and EKE-M for multicast ones. *Unicast* networks allow for communication only between a single sender and a single receiver, while *multicast* networks allow for communication between a single sender and multiple receivers. For multicast networks, all messages from individual single senders can be sent in parallel during a single round to all receivers, thus more round-efficient group-based protocols can be designed in such networks.

Note that all arithmetic operations in this paper are performed under cyclic group $\mathbf{G} = \langle g \rangle$ of prime order.

2.1 N-Party EKE-U Protocol Variants

The EKE-U makes use of three types of functions which differ mainly in the number of elements produced at their respective outputs:

$$\begin{aligned}
\pi(\alpha_1,\ldots,\alpha_{i-1},\alpha_i) &= \{\alpha_1,\ldots,\alpha_{i-1}\},\\
\phi(\{\alpha_1,\ldots,\alpha_{i-1},\alpha_i\},x) &= \{\alpha_1^x,\ldots,\alpha_{i-1}^x,\alpha_i,\alpha_i^x\},\\
\xi(\{\alpha_1,\ldots,\alpha_{i-1},\alpha_i\},x) &= \{\alpha_1^x,\ldots,\alpha_{i-1}^x,\alpha_i^x\}.
\end{aligned}$$

Note that π produces an output that is simply equal to its input less the last input element, and further π is only used by C_{n-1}. The functions ϕ and ξ take in the same number of input elements, and their outputs are similar except that ϕ has one more output element than ξ.

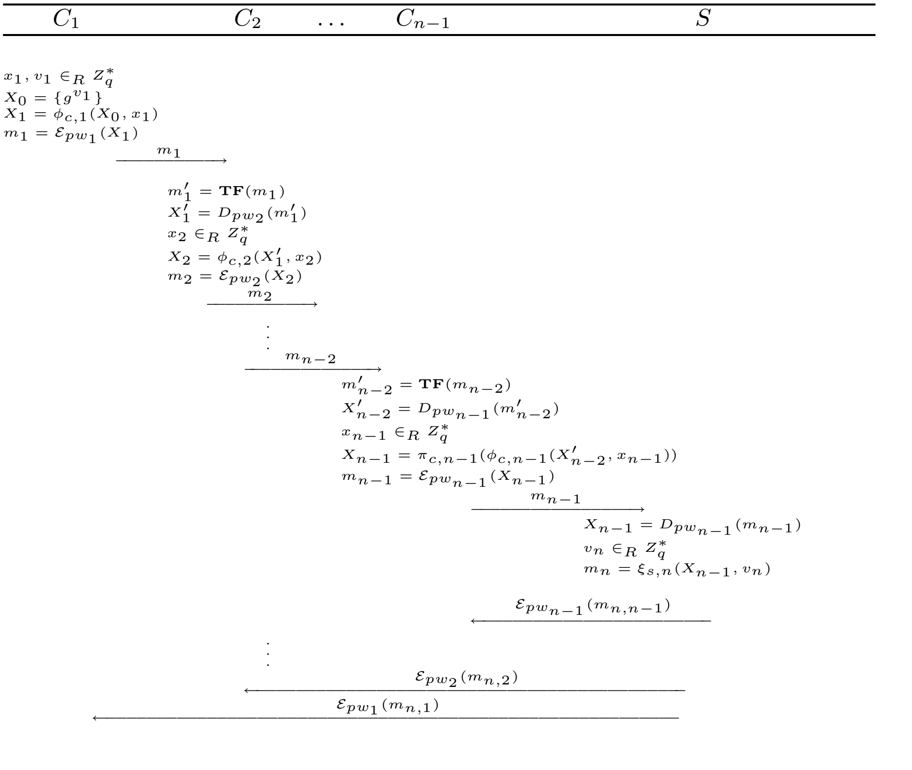

Fig. 1. Main protocol of n-party EKE-U

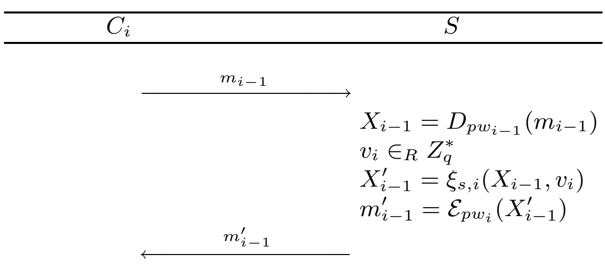

Fig. 2. TF protocol of n-party EKE-U

The main bulk of the EKE-U protocol is illustrated in Fig. 1, where clients $C_1, \ldots, C_{n-1}$ and the server S are arranged in a line. During the up-flow stage starting from C_1, each client C_i basically chooses its own secret x_i and calls the ϕ function to raise the intermediate value X'_{i-1} to the power of this x_i in order to generate the value X_i. This is encrypted with client C_i's password pw_i and sent to the next client C_{i+1} as the message m_i. Upon receipt of this,

	S	C_1	C_2	$\cdots$	C_{n-1}
Round 1	$s_i \in_R Z_q^*$ $\mathcal{E}_{pw_i}(g^{s_i})$	$x_1 \in_R Z_q^*$ $\mathcal{E}_{pw_1}(g^{x_1})$	$x_2 \in_R Z_q^*$ $\mathcal{E}_{pw_2}(g^{x_2})$	$\cdots$ $\cdots$	$x_{n-1} \in_R Z_q^*$ $\mathcal{E}_{pw_{n-1}}(g^{x_{n-1}})$
Round 2	$N \in_R Z_q^*$ $sk_1 \oplus N\| \ldots \|sk_{n-1} \oplus N$				

Fig. 3. N-party EKE-M

C_{i+1} initiates an additional sub-protocol known as the **TF** protocol (see Fig. 2 and note that the function ξ is used here) with the server S so that the received message which was encrypted under client C_i's password could be decrypted and re-encrypted under client C_{i+1}'s password to form m_i'. This then allows C_{i+1} to access the decrypted contents of m_i', namely X_i' and the same process repeats until S receives the message m_{n-1} from C_{n-1}. The down-flow stage then starts by having S compute from m_{n-1} the keying material $m_{n,i}$ meant for client C_i ($i = 1, \ldots, n-1$), encrypt under pw_i and send these out to each corresponding client. Finally, each client with his own pw_i and x_i can perform the decryption and compute the session key $sk = (m_{n,i})^{x_i} = g^{v_n \prod_{i=1}^{n-1}(v_i x_i)}$.

Byun and Lee [7] also mention that an optional mutual authentication step based on key confirmation could be appended to the scheme if it is desired to ensure that all other clients have really computed the agreed session key sk. In this case, each client computes an authenticator $\mathcal{H}(C_i||sk)$, which is the hash value of client index (C_i) and new session key (sk), and sends this to all other clients for verification. Note however that even if this step is made compulsory, it does not protect EKE-U against our attacks in Sections 3.3 and 3.4.

There is a strengthened version [8] of EKE-U and this will be explained in Section 3.4.

2.2 N-Party EKE-M Protocol Variants

EKE-M is much simpler than EKE-U and is shown in Fig. 3. It consists of two rounds. Round 1 is basically a simultaneous run of a 2-party PAKE between each client with the server to set up a secure channel (in the confidentiality sense) between them. In Round 2, the server distributes a common keying message to all clients via the secure channel. This will be used to form the common secret session key sk among all clients. More precisely, denote $sk_i = \mathcal{H}_1(\mathcal{E}_{pw_1}(g^{x_1})\| \ldots \|\mathcal{E}_{pw_{n-1}}(g^{x_{n-1}})\|g^{x_i s_i})$ and $sk = \mathcal{H}_2(\mathcal{E}_{pw_1}(g^{x_1})\| \ldots \| \mathcal{E}_{pw_{n-1}}(g^{x_{n-1}})\|sk_1 \oplus N\| \ldots \|sk_{n-1} \oplus N\|N)$. Note that $\mathcal{H}_1$ and $\mathcal{H}_2$ are standard hash functions.

There is also a strengthened version of EKE-M proposed by Byun and Lee [8] to prevent the undetectable online dictionary attack in [21]. The basic idea is to add an extra step after Round 1 where an authenticator $\mathcal{H}(sk_i||C_i)$ is broadcast by each client (or server) to be checked by all parties before Round 2 starts.

3 Cryptanalysis of the N-Party EKE-U Variants

In view of the low entropy password, the basic requirement for a PAKE is security against dictionary attacks on the password. Such attacks are typically online or offline, depending on whether or not the attacker needs to verify each guessed password by interacting online (being involved in a protocol run) with other parties. Another basic requirement of PAKEs is that they do not allow impersonation attacks where an attacker masquerades as any legitimate party because if this happens, there will be a non-achievement of mutual authentication.

3.1 Tang-Chen Attack

Before describing our attacks, we first briefly discuss an insider offline dictionary attack on EKE-U given by Tang and Chen [21]. See [15] for a formal treatment of insider attacks on group AKEs.

The basic idea behind this attack is that a malicious client C_j modifies the first two components (g_1, g_2) in the message X_j of $m_j = \mathcal{E}_{pw_j}(X_j)$ that it sends to C_{j+1} during the up-flow stage of the main protocol, such that they satisfy the relation $g_1^{\alpha} = g_2$. Then right at the end of the **TF** protocol when the server S returns $m'_j = \mathcal{E}_{pw_{j+1}}(X'_j)$ to C_{j+1}, this is intercepted by the malicious C_j who then guesses the value of pw_{j+1} and verifies his guess by checking if the first two components that he had initially modified satisfy the given relation.

At first glance, it seems that this attack requires having to modify the message $m_j = \mathcal{E}_{pw_j}(X_j)$. However, as later pointed out in the same paper [21], this attack could work without this requirement. Instead, it suffices to decrypt m'_j with the guessed password pw_{j+1} and check if the last two components (β, γ) of X'_j satisfy the relation $\beta^{x_j} = \gamma$.

Note however that even with this relaxation, the latter attack still limits the malicious C_j to attack only his next neighbour C_{j+1} but not on the other clients because the components within in his possessed X'_{j-1}, X_j do not allow him to verify any two components of these other clients' messages without having to guess the secrets of the server v_i, $i \in \{1, \ldots, n\}$ or the secrets of other clients x_t $(t \neq j)$.

3.2 By Any Outsider

Byun and Lee [7] have cleverly designed the EKE-U protocol such that the $m_{n,i}$ within each keying material message $\mathcal{E}_{pw_i}(m_{n,i})$ distributed by S to each client C_i in the down-flow stage does not have the random secret x_i chosen by client C_i in its exponent, thus only client C_i would be able to make use of its $m_{n,i}$ ($i^{\mathtt{th}}$ component of the message m_n) to generate the session key material $K = (m_{n,i})^{x_i} = (g^{x_1 \cdots x_{n-1}})^{v_1 \cdots v_n}$. Further, different functions (ϕ, π, ξ) are used in the main and **TF** protocols, e.g. each of the three functions produces an output having different number of elements, and ϕ is used in the main protocol while ξ is used in the **TF** protocol; thus it appears an attacker cannot exploit one protocol as an oracle for answering challenge-response-like queries in the other protocol.

However, note that this is only true for the communications during the up-flow stage of the main protocol from C_1 through C_{n-1}, but not true from C_{n-1} to S because for the latter there is an extra function π (see Fig. 1 in addition to the function ϕ that is used by C_{n-1}. Thus the output of the composition of the functions $\pi \circ \phi$ done by C_{n-1} during the main protocol results in the same number of elements as that of the output of the ξ function computed by S in the **TF** protocol; i.e. S can be exploited during the **TF** protocol as an oracle to generate messages supposedly generated by C_{n-1} during the main protocol when in fact C_{n-1} need not be present at all.

Our attack further exploits the fact that the messages transmitted during the **TF** protocol (Fig. 2) between a client and the server are similar in form to the messages transmitted during the up- or down-flow of the main unicast protocol (Fig. 1). In particular, message m_i and m'_{i-1} are both functions of $\mathcal{E}_{pw_i}(\cdot)$. Thus, the server S which is intended by the designers to act as an interpreter between two neighbouring clients, C_i and C_{i-1} could be used by the attacker as an oracle to generate messages m_i supposedly generated by the next neighbouring client C_i even when C_i is not present.

For ease of illustration, we take $n = 4$ (as in Fig. 4) though it similarly applies for any n. Note that in this case, $C_{n-1} = C_3$.

1. The attacker captures the message $m_2 = \mathcal{E}_{pw_2}(X_2)$ sent from C_2 to C_3 during the up-flow stage of the main protocol.
2. The attacker then initiates the **TF** protocol by forwarding this m_2 to S.
3. S thinks[1] this is from C_3 and decrypts it with pw_2 to obtain X_2. It then computes
$$X'_2 = \xi(X_2, v_3) = \{g^{v_1 v_2 x_2 v_3}, g^{v_1 x_1 v_2 v_3}, g^{v_1 x_1 v_2 x_2 v_3}\} \quad (1)$$
and encrypts this with pw_3 to get $m'_2 = \mathcal{E}_{pw_3}(X'_2)$ and returns this m'_2 thus completing the **TF** protocol.
4. The attacker now has m'_2 which he simply reuses as $m_3 = \mathcal{E}_{pw_3}(X_3) = \mathcal{E}_{pw_3}(X'_2)$ and then impersonates C_3 by sending this to S in the main protocol. This completes the up-flow stage.
5. To start the down-flow stage, S decrypts m_3 to obtain
$$X_3 = \{g^{v_1 v_2 x_2 v_3}, g^{v_1 x_1 v_2 v_3}, g^{v_1 x_1 v_2 x_2 v_3}\} \quad (2)$$
and then chooses v_4 to compute
$$m_4 = \xi(X_3, v_4) = \{g^{v_1 v_2 x_2 v_3 v_4}, g^{v_1 x_1 v_2 v_3 v_4}, g^{v_1 x_1 v_2 x_2 v_3 v_4}\}. \quad (3)$$
Each of these elements of m_4, denoted in turn as $m_{4,1}, m_{4,2}, m_{4,3}$ are then encrypted with the respective passwords pw_i of client C_i $(i = 1, \ldots, 3)$ and sent to each client respectively as $\mathcal{E}_{pw_i}(m_{4,i})$ for $(i = 1, \ldots, 3)$.

[1] Note that there is no explicit authentication of a client by S. An apparent way for S to properly keep in sequence is to track the number of **TF** sessions that have been initiated with it. The i^{th} session would be taken to come from client C_{i+1} since C_1 does not initiated any **TF** with S.

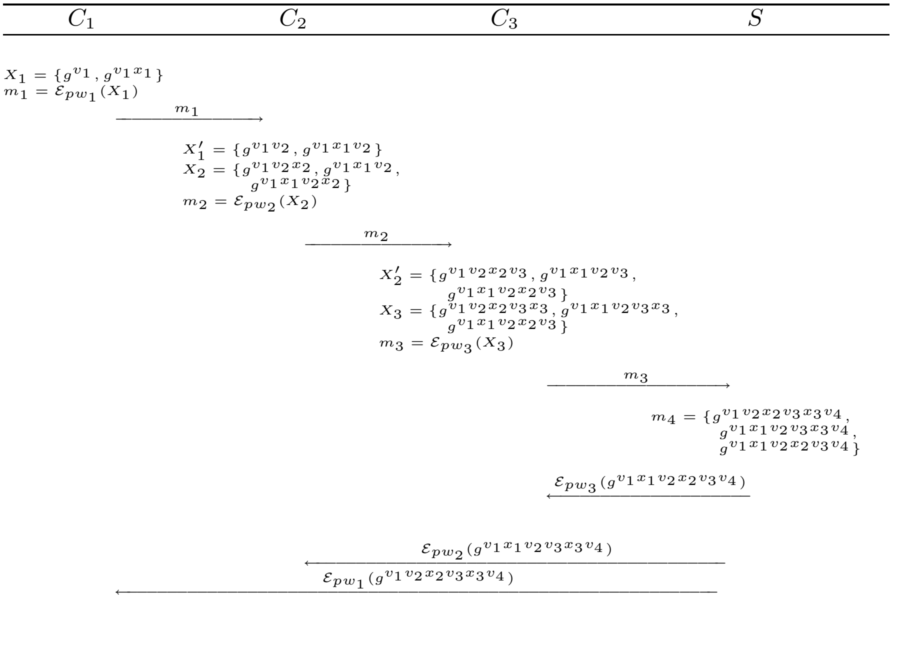

Fig. 4. An example of n-party EKE-U main protocol for n=4

6. Each client C_i $(i = 1, \ldots, 3)$ can then decrypt $\mathcal{E}_{pw_i}(m_{4,i})$ and thus compute $sk = (m_{4,i})^{x_i} = (g^{x_1 x_2})^{v_1 v_2 v_3 v_4}$.

Note that though our attack can be used to attack only C_{n-1} and not any other client, the main plus is that it can be mounted by any outsider (in contrast to the attack in [21] which requires a malicious insider) and applies even without needing client C_{n-1} to be present. Having said that, C_{n-1}'s presence would pose no problem for the attacker either. Though the attacker is unable to recover the session key sk himself, he has successfully led all parties (except client C_{n-1} who is not present) to establish a totally new session key among them. This could also be viewed as a variant of the unknown key-share attack [10, 2, 14] in the n-party case since each client (except C_{n-1}) believes it is sharing a session key with all other clients including C_{n-1} which is true, but C_{n-1} is not present and does not know that such a key has been established. In constrast, recall that an unknown key-share attack on a 2-party case is where one party A believes it is sharing a session key with B which is rightly so, but B instead believes it is sharing a session key with $E \neq A$.

To prevent this attack, the mutual authentication step (e.g. via key confirmation [14]) must be made compulsory. Nevertheless, when performed by a malicious insider, the mutual authentication step is no longer effective to prevent this attack, and it further becomes an offline dictionary attack allowing him to retrieve the password of C_{n-1}, as will be explained next.

3.3 By a Malicious Insider

A malicious client C_i could launch a more devastating variant of the previous attack since he could exploit it to further obtain the password of the innocent client C_{n-1}. This offline dictionary attack works as follows:

1. The attacker, client C_i $(i \neq n-1)$ performs steps 1 through 5 of Section 3.2.
2. Further, since the attacker is an insider, he could also decrypt the keying material intended for him $\mathcal{E}_{pw_i}(m_{4,i})$. We illustrate with an example. Consider C_1 is the malicious client. It can be similarly shown for all other clients C_i for $(i \neq n-1)$. He can obtain $g^{v_1v_2x_2v_3v_4}$ from $\mathcal{E}_{pw_1}(m_{4,1}) = \mathcal{E}_{pw_1}(g^{v_1v_2x_2v_3v_4})$.
3. With his value of x_1, he can compute $y = (g^{v_1v_2x_2v_3v_4})^{x_1} = g^{v_1x_1v_2x_2v_3v_4}$.
4. He intercepts $\mathcal{E}_{pw_3}(m_{4,3}) = \mathcal{E}_{pw_3}(g^{v_1x_1v_2x_2v_3v_4})$ meant for client C_3, and makes guesses for all possible values of pw_3. For each guessed pw_3, he decrypts $\mathcal{E}_{pw_3}(m_{4,3})$ and obtains $z = g^{v_1x_1v_2x_2v_3v_4}$. He then checks if z equals y. The correct pw_3 would satisfy this.

This attack can be mounted by any client C_i against C_{n-1}, thus it complements the attack in [21] where the attack is mounted by any client C_i against his neighbour C_{i+1}.

Note also that this attack works even with the mutual authentication step included since C_i has no problem in computing sk.

3.4 Attacking the Strengthened N-Party EKE-U

In [8], Byun and Lee suggested a strengthened n-party EKE-U protocol to counter the insider offline dictionary attack due to Tang and Chen [21].

Their basic idea to counter the attack is to use an ephemeral session key $sk_i = \mathcal{H}(C_i \| S \| g^{a_i} \| g^{b_i} \| g^{a_ib_i})$ instead of the password pw_i to encrypt keying material during both the up- and down-flow of the main protocol, where a_i and b_i are the random number chosen by C_i and S respectively.

Nevertheless, we first remark that this strengthened variant also falls to our attacks in the Sections 3.2 and 3.3 since it inherits from the original version the same properties we exploited, i.e. (1) the composition of functions $\pi \circ \phi$ produces an output with the same number of elements as that produced by ξ; (2) messages transmitted during the **TF** protocol are the same in form to messages transmitted during the main protocol.

More interestingly, we have a further undetectable online dictionary attack [11] on this strengthened variant as follows, again assuming for the purpose of illustration that $n = 4$ thus we have the parties C_1, C_2, C_3 and S:

1. All malicious clients except C_1 collude [18], meaning they share their secrets x_i.
2. They choose v and x, and for each guess of pw_1,
   (a) They compute $m_1 = \mathcal{E}_{pw_1}(X_1)$ where $X_1 = \{g^v, g^{vx}\}$.
   (b) Then C_2 starts the **TF** protocol with S, etc., and the rest of the up-flow proceeds as normal.

(c) Then during the down-flow, the keying material messages sent by S to C_1, C_2 and C_3 would be $\mathcal{E}_{pw_1}(g^{vv_2v_3v_4x_2x_3})$, $\mathcal{E}_{pw_2}(g^{vv_2v_3v_4xx_3})$ and $\mathcal{E}_{pw_3}(g^{vv_2v_3v_4xx_2})$.
(d) Now the colluding clients C_2 and C_3 can easily obtain $y = g^{vv_2v_3v_4}$ from $\mathcal{E}_{pw_2}(g^{vv_2v_3v_4xx_3})$ or $\mathcal{E}_{pw_3}(g^{vv_2v_3v_4xx_2})$, and their knowledge of x, x_2 and x_3.
(e) They then use their current guess of pw_1 to decrypt $\mathcal{E}_{pw_1}(g^{vv_2v_3v_4x_2x_3})$ to get z. They compare this z with $y^{x_2x_3}$, where y was computed in the previous step. A match means the guess of pw_1 is correct.

This is online because every time pw_1 is guessed, the attackers have to initiate a protocol run with S, but this is undetectable because S would not notice anything wrong while C_1 does not even have to be present.

The weakness exploited here is that the message from C_1 to C_2 is encrypted with a low-entropy password pw_1 instead of sk_1. Thus a direct fix is to use sk_1 in place of pw_1 similar to how sk_i (for $i \neq 1$) were used in place of pw_i for this strengthened EKE-U scheme.

4 N-Party EKE-M Does Not Provide Key Privacy

Byun and Lee [7] also proposed a multicast variant known as the n-party EKE-M protocol. It is illustrated in Fig. 3.

This variant does not exhibit the 'S-oracle' property of the U variant, i.e. the server S cannot be exploited as an oracle to generate messages that appear to be from a client, thus it does not appear to fall to our attacks on EKE-U. Nevertheless, there is one major problem with this M variant, namely that the server S is able to compute the session key sk established by the clients. This is quite unlike the U variant where even S is unable to know what sk is, and thus this M variant is undesirable in the sense that the privacy of the clients' communications cannot be safeguarded against a third-party server.

This *key privacy* property is important because it would mean less trust [12, 19] needs to be put on a third-party server, who may not always be malicious but could sometimes be curious [1]. The first known n-party (for n=3) EKE scheme to have this property is due to Steiner *et al.* [20] and this concept was later formally treated by Abdalla *et al.* [1]. Abdalla *et al.* argue that key privacy is the main difference between a key distribution protocol (for which the session key is known to the server) and a key exchange protocol (for which the session key remains unknown to the server). Thus, a true key exchange protocol where each party (in this case the client) contributes equal parts to the established session key, should have key privacy because the third-party server should not be able to listen in on future secret communications among the clients, and hence should not be able to know what this session key is.

Note that the strengthened EKE-M variant in [8] has the same problem even when mutual authentication via key confirmation is included, because the point here is that the server can compute sk even when C_i is not present, so mutual authentication is irrelevant.

We do not see any way to fix this with minor tweaks without destroying the basic structure of this M scheme, because essentially each client interacts only with the server, and never with each other, thus the keying material components that they contribute to the final establishment of the session key via a Diffie-Hellman way, can only be translated (decrypted with one password and re-encrypted with another) by the middleman S, thus S is able to view all communicated messages that it translates.

Alternatively, one could adopt the approach in [1] by appending one more phase where each client interacts directly with the other clients by contributing its secret part to jointly form the key but this would be infeasible for $n > 3$ parties. Unless one resorts to using the method used for EKE-U where each client in turn adds his secret to the key material accummulatively while forwarding from one client to the next until it reaches the server. However, this is then essentially EKE-U and thus we end up destroying the original EKE-M structure.

If it is desired that this key privacy against the server be upheld, then this variant should not be used.

5 Conclusion

We have illustrated attacks (impersonation, dictionary or collusion attacks) on the n-party EKE-U variants proposed by Byun and Lee [7, 8].

EKE-U [7], even with strengthening [8], falls to our attacks in Sections 3.2 to 3.4, while EKE-M is not desirable as it does not provide key privacy. But to fix the key privacy problem requires clients to directly communicate with one another to contribute their secret key parts accummulatively, leading us therefore to EKE-U.

Thus it appears that *strengthened* EKE-U is the potential way to proceed for provably secure n-party PAKEs. Hence, to fix EKE-U, the mutual authentication step is compulsory in order to prevent the attack in Section 3.2, though attacks in Sections 3.3 and 3.4 still apply. A simple fix to prevent the attack in Section 3.3 is to require the server to check that $x_{n-1} \neq 1$ before replying so that it is not exploited as an oracle. To prevent the attack in Section 3.4, C_1 needs to also initiate the **TF** protocol to generate sk_1 with the server and use sk_1 instead of pw_1 in constructing m_1.

Acknowledgement

We thank the anonymous referees, especially for pointing out to explicitly differentiate between the basic unknown key-share attack applied to 2-party AKEs and our impersonation attack for the n-party case in Section 3.2. We thank IACR for maintaining the ePrint Archive which has become the de facto venue for publicly disseminating crypto documents (full versions, short notes etc.). We thank God for His many blessings. The first author dedicates this work to the memory of his grandmother MOK Kiaw (1907-1999), for being a constant inspiration.

References

1. M. Abdalla, P.-A. Fouque, and D. Pointcheval. Password-Based Authenticated Key Exchange in the Three-Party Setting. In *Proc. PKC '05*, LNCS 3386, pp. 65-84, Springer-Verlag, 2005.
2. J. Baek, and K. Kim. Remarks on the Unknown Key-share Attacks. In *IEICE Transactions on Fundamentals*, Vol. E83-A, No. 12, pp.2766-2769, 2000.
3. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In *Advances in Cryptology - EUROCRYPT '00*, LNCS 1807, pp. 139-155, Springer-Verlag, 2000.
4. M. Bellare, and P. Rogaway. Provably Secure Key Distribution - the Three Party Case. In *Proc. ACM-SToC '96*, pp. 72-84, 1996.
5. S.M. Bellovin, and M. Merritt. Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attacks. In *Proc. IEEE S&P '92*, pp. 72-84, IEEE Press, 1992.
6. E. Bresson, O. Chevassut, and D. Pointcheval. Group Diffie Hellman Key Exchange Secure against Dictionary Attacks. In *Advances in Cryptology - ASIACRYPT '02*, LNCS 2501, pp. 497-514, Springer-Verlag, 2002.
7. J.W. Byun, and D.H. Lee. N-Party Encrypted Diffie-Hellman Key Exchange Using Different Passwords. In *Proc. ACNS '05*, LNCS 3531, pp. 75-90, Springer-Verlag, 2005.
8. J.W. Byun, and D.H. Lee. Comments on Weaknesses in Two Group Diffie-Hellman Key Exchange Protocols. IACR ePrint Archive, 2005/209, 2005.
9. J.G. Choi, K. Sakurai, and J.H. Park. Does It Need Trusted Third Party? Design of Buyer-Seller Watermarking Protocol without Trusted Third Party. In *Proc. ACNS '03*, LNCS 2846, pp. 265-279, Springer-Verlag, 2003.
10. W. Diffie, P. van Oorschot, and M. Wiener. Authentication and Authenticated Key Exchanges. In *Design, Codes and Cryptography*, Vol. 2, No. 2, pp.107-125, 1992.
11. Y. Ding, and P. Horster. Undetectable On-line Password Guessing Attacks. In *ACM Operating Systems Review*, Vol. 29, No. 4, pp.77-86, 1995.
12. M.K. Franklin, and M.K. Reiter. Fair Exchange with a Semi-trusted Third Party (Extended Abstract). In *Proc. ACM-CCS '97*, pp. 1-5, 1997.
13. D. Jablon. Strong Password-only Authenticated Key Exchange. In *ACM Computer Communications Review*, Vol. 20, No. 5, pp.5-26, 1996.
14. B.S. Kaliski, Jr. An Unknown Key-share Attack on the MQV Key Agreement Protocol. In *ACM Transactions on Information and System Security*, Vol. 4, No. 3, pp.275-288, 2001.
15. J. Katz, and J.S. Shin. Modeling Insider Attacks on Group Key-Exchange Protocols. In *Proc. ACM-CCS '05*, pp. 180-189, 2005.
16. C.-L. Lin, H.-M. Sun and T. Hwang. Three-Party Encrypted Key Exchange: Attacks and a Solution. In *ACM Operating Systems Review*, Vol. 34, No. 4, pp.12-20, 2000.
17. C.-L. Lin, H.-M. Sun, M. Steiner and T. Hwang. Three-Party Encrypted Key Exchange Without Server Public-Keys. In *IEEE Communication Letters*, Vol. 5, No. 12, pp.497-499, IEEE Press, 2001.
18. S. Saeednia, and R. Safavi-Naini. Efficient Identity-based Conference Key Distribution Protocols. In *Proc. ACISP '98*, LNCS 1438, pp. 320-331, Springer-Verlag, 1998.
19. V. Shmatikov, and J.C. Mitchell. Finite-state Analysis of Two Contract Signing Protocols. In *Theoretical Computer Science*, Vol. 283, No. 2, pp.419-450, Elsevier, 2002.

20. M. Steiner, G. Tsudik, and M. Waider. Refinement and Extension of Encrypted Key Exchange. In *ACM Operating Systems Review*, Vol. 29, No. 3, pp.22-30, 1995.
21. Q. Tang, and L. Chen. Weaknesses in Two Group Diffie-Hellman Key Exchange Protocols. IACR ePrint Archive, 2005/197, 2005.
22. J. Zhou, F. Bao, and R.H. Deng. Validating Digital Signatures without TTP's Time-Stamping and Certificate Revocation. In *Proc. ISC '03*, LNCS 2851, pp. 96-110, Springer-Verlag, 2003.