Weaknesses in two group Diffie-Hellman key exchange protocols

Qiang Tang
Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
qiang.tang@rhul.ac.uk

24th June 2005

Abstract

In this paper we show that the password-based Diffie-Hellman key exchange protocols due to Byun and Lee suffer from dictionary attacks.

1 Introduction

Recently, Byun and Lee proposed two password-based Diffie-Hellman key exchange protocols [2] which are claimed to be provably secure based on Diffie-Hellman problems. For simplicity of description, we refer to the two protocols as the EKE-U and EKE-M protocols, following the notation used in [2].

Byun and Lee claim that the protocols are secure against dictionary attacks, especially insider dictionary attacks. However, we show that the EKE-U protocol suffers from offline dictionary attacks, and the EKE-M protocol suffers from undetectable online dictionary attacks which can be mounted by any malicious participant.

The rest of this paper is organised as follows. In Section 2 we review both the EKE-U and the EKE-M protocols. In section 3 we demonstrate security vulnerabilities in both protocols. In the final section, we conclude this paper.

2 Review of the EKE-U and the EKE-M protocols

The following assumptions are made in both the EKE-U and the EKE-M protocols. Suppose that g is the generator of a multiplicative cyclic group of prime order q, the server S independently shares a unique password pw_i with user U_i $(1 \leq i \leq n)$, and (ϵ, D) is an ideal cipher [1], where ϵ is the encryption algorithm and D is the decryption algorithm. Additionally, h is a full-domain hash function [3], h_1 and h_2 are one-way hash functions, and $||$ is the string concatenation operator.

For simplicity of description, we assume that $n \geq 3$ in the rest of this paper. It is straightforward to verify that our results also apply to the case where $n = 2$.

2.1 Description of the EKE-U protocol

The EKE-U protocol is designed for use in a unicast network. The users U_i $(1 \leq i \leq n)$ and S perform the following steps.

1. U_1 selects two random numbers v_1 and x_1 $(1 \leq v_1, x_1 \leq q-1)$, and computes $T_1 = \{g^{v_1}, g^{v_1 x_1}\}$. U_1 then sends $M_1 = \epsilon_{pw_1}(T_1)$ to U_2.
2. After receiving M_1 from U_1, U_2 forwards it to S.
3. After receiving M_1 from U_2, S first decrypts it using the password pw_1 to obtain $T_1 = \{g^{v_1}, g^{v_1 x_1}\}$. S then selects a random number v_2 $(1 \leq v_2 \leq q-1)$, and computes $T'_1 = \{g^{v_1 v_2}, g^{v_1 v_2 x_1}\}$. Finally, S sends $M'_1 = \epsilon_{pw_2}(T'_1)$ to U_2.
4. After receiving M'_1 from S, U_2 first decrypts it using his password pw_2 to obtain $T'_1 = \{g^{v_1 v_2}, g^{v_1 v_2 x_1}\}$. U_2 then selects a random number x_2 $(1 \leq x_2 \leq q-1)$, and computes $T_2 = \{g^{v_1 v_2 x_1}, g^{v_1 v_2 x_2}, g^{v_1 v_2 x_1 x_2}\}$. Finally, U_2 sends $M_2 = \epsilon_{pw_2}(T_2)$ to U_3.
5. Recursively, U_j $(3 \leq j \leq n-1)$ and S perform the following steps.

   (a) After receiving M_{j-1} from U_{j-1}, where

   $$M_{j-1} = \epsilon_{pw_{j-1}}(T_{j-1}),$$

   $$T_{j-1} = \{g^{V_{j-1} \cdot (X_{j-1}/x_1)}, g^{V_{j-1} \cdot (X_{j-1}/x_2)}, \cdots, g^{V_{j-1} \cdot (X_{j-1}/x_{j-1})}, g^{V_{j-1} \cdot X_{j-1}}\},$$

   $$V_{j-1} = v_1 \cdot v_2 \cdots v_{j-1},\ X_{j-1} = x_1 \cdot x_2 \cdots x_{j-1},$$

   U_j forwards it to S.

(b) After receiving M_{j-1} from U_j, S first decrypts it using the password pw_{j-1} to obtain T_{j-1}. S then selects a random number v_j $(1 \le v_j \le q-1)$, and computes T'_{j-1}, where

$$T'_{j-1} = \{g^{V_j \cdot (X_{j-1}/x_1)}, g^{V_j \cdot (X_{j-1}/x_2)}, \cdots, g^{V_j \cdot (X_{j-1}/x_{j-1})}, g^{V_j \cdot X_{j-1}}\},$$

$$V_j = v_1 \cdot v_2 \cdots v_j,\ X_{j-1} = x_1 \cdot x_2 \cdots x_{j-1}.$$

Finally, S sends $M'_{j-1} = \epsilon_{pw_j}(T'_{j-1})$ to U_j.

(c) After receiving M'_{j-1} from S, U_j first decrypts it using his password pw_j to obtain T'_{j-1}. U_2 then selects a random number x_j $(1 \le x_j \le q-1)$, and computes T_j as

$$T_j = \{g^{V_j \cdot (X_j/x_1)}, g^{V_j \cdot (X_j/x_2)}, \cdots, g^{V_j \cdot (X_j/x_j)}, g^{V_j \cdot X_j}\},$$

$$V_j = v_1 \cdot v_2 \cdots v_j,\ X_j = x_1 \cdot x_2 \cdots x_j.$$

Finally, U_j sends $M_j = \epsilon_{pw_j}(T_j)$ to U_{j+1}.

6. After receiving M_{n-1} from U_{n-1}, U_n forwards it to S.

7. After receiving M_{n-1} from U_n, S first decrypts it using the password pw_{n-1} to obtain T_{n-1}. S then selects a random number v_n $(1 \le v_n \le q-1)$, and computes T'_{n-1}, where

$$T'_{n-1} = \{g^{V_n \cdot (X_{n-1}/x_1)}, g^{V_n \cdot (X_{n-1}/x_2)}, \cdots, g^{V_n \cdot (X_{n-1}/x_{n-1})}, g^{V_n \cdot X_{n-1}}\},$$

$$V_n = v_1 \cdot v_2 \cdots v_n,\ X_{n-1} = x_1 \cdot x_2 \cdots x_{n-1}.$$

Finally, S sends $M'_{n-1} = \epsilon_{pw_n}(T'_{n-1})$ to U_n.

8. After receiving M'_{n-1} from S, U_n first decrypts it using his password pw_n to obtain T'_{n-1}. U_n then selects a random number x_n $(1 \le x_n \le q-1)$, and computes T_n as

$$T_n = \{g^{V_n \cdot (X_n/x_1)}, g^{V_n \cdot (X_n/x_2)}, \cdots, g^{V_n \cdot (X_n/x_n)}\},$$

$$V_n = v_1 \cdot v_2 \cdots v_n,\ X_n = x_1 \cdot x_2 \cdots x_n.$$

Finally, U_n sends $M_n = \epsilon_{pw_n}(T_n)$ to S.

It should be noted that T_n is computed differently from T_j $(1 \le j \le n-1)$ in order to prevent S from computing the ultimate session key.

9. After receiving M_n from U_n, S first decrypts it using the password pw_n to obtain T_n. S then selects a random number v_{n+1} $(1 \le v_{n+1} \le q-1)$, and computes and sends $E_i = \epsilon_{pw_i}(g^{V_{n+1} \cdot (X_n/x_i)})$ to U_i $(1 \le i \le n)$, where

$$V_{n+1} = v_1 \cdot v_2 \cdots v_{n+1},\ X_n = x_1 \cdot x_2 \cdots x_n.$$

10. After receiving E_i from S, U_i $(1 \leq i \leq n)$ decrypts it using his password pw_i to obtain $g^{V_{n+1}\cdot(X_n/x_i)}$, and then computes the key material and session key as $K = (g^{V_{n+1}\cdot(X_n/x_i)})^{x_i}$ and $sk = h(clients||K)$, where $clients$ is the concatenation of the identifiers of U_i $(1 \leq i \leq n)$.

    If key confirmation is required, then U_i computes and broadcasts $Auth_i = h(i||sk)$.

11. After receiving every $Auth_j$ $(1 \leq j \leq n-1, j \neq i)$, U_i checks whether it equals $h(i||sk)$. If all the checks succeed, U_i confirms that the protocol has succeeded. Otherwise, U_i terminates the protocol as a failure.

2.2 Description of the EKE-M protocol

The EKE-M protocol is designed for use in a multicast network. U_i $(1 \leq i \leq n)$ and S perform the following steps.

1. S selects $q-1$ random numbers s_i $(1 \leq s_i \leq q-1)$, and then sends $\epsilon_{pw_i}(g^{s_i})$ to U_i. Concurrently, U_i selects a random number x_i $(1 \leq x_i \leq q-1)$, and then broadcasts $\epsilon_{pw_i}(g^{x_i})$.

2. After receiving every $\epsilon_{pw_i}(g^{x_i})$ $(1 \leq i \leq n-1)$, S decrypts each of them to obtain g^{x_i}. S then computes the shared ephemeral key with U_i as $sk_i = h_1(sid'||g^{x_i s_i})$, where

    $$sid' = \epsilon_{pw_1}(g^{x_1})||\epsilon_{pw_2}(g^{x_2})||\cdots||\epsilon_{pw_{n-1}}(g^{x_{n-1}})$$

    Finally, S selects a random secret N, and broadcasts $m_i = N \oplus sk_i$, where, as throughout this paper, $\oplus$ denotes the bit-wise exclusive-or operator.

3. After receiving all the messages from S, U_i first constructs sid' in the same way as S, decrypts $\epsilon_{pw_i}(g^{s_i})$, computes $sk_i = h_1(sid'||g^{s_i x_i})$, and then computes $N = m_i \oplus sk_i$. Finally, U_i computes the session key as $sk = h_2(SIDS||N)$, where

    $$SIDS = sid'||sk_1 \oplus N||sk_2 \oplus N||\cdots||sk_{n-1} \oplus N$$

    If key confirmation is required, then U_i computes and broadcasts $Auth_i = h(i||sk)$.

4. After receiving every $Auth_j$ $(1 \leq j \leq n-1, j \neq i)$, U_i checks whether it equals $h(i||sk)$. If all the checks succeed, U_i confirms that the protocol has succeeded. Otherwise, U_i terminates the protocol as a failure.

3 Security vulnerabilities in the EKE-U and the EKE-M protocols

3.1 Security vulnerability in the EKE-U protocol

In the EKE-U protocol, a malicious participant U_j $(1 \leq j \leq n-1)$ can mount offline dictionary attacks against U_{j+1}.

To mount the attack, U_j selects t_1 and t_2, and then sends M'_j to U_{j+1} instead of M_j, where

$$M'_j = \epsilon_{pw_j}(T'_j),$$

$$T'_j = \{g^{t_1}, g^{t_1 t_2}, g^{V_j \cdot (X_j/x_3)}, \cdots, g^{V_j \cdot (X_j/x_j)}, g^{V_j \cdot X_j}\},$$

$$V_j = v_1 \cdot v_2 \cdots v_j,\ X_j = x_1 \cdot x_2 \cdots x_j.$$

After receiving M_j, U_{j+1} will forward it to S. The attack succeeds based on the following lemma.

Lemma 3.1. *As a result of the above attack, U_i can mount an offline dictionary attack against U_{i+1}.*

Proof. After receiving M_j from U_{j+1}, S first decrypts it using the password pw_j to obtain T'_j. S then selects a random number v_{j+1} $(1 \leq v_{j+1} \leq q-1)$, and computes T'_j, where

$$T'_j = \{g^{t_1 v_{j+1}}, g^{t_1 t_2 v_{j+1}}, g^{V_{j+1} \cdot (X_j/x_3)}, \cdots, g^{V_{j+1} \cdot (X_j/x_j)}, g^{V_{j+1} \cdot X_j},$$

$$V_{j+1} = v_1 \cdot v_2 \cdots v_{j+1},\ X_j = x_1 \cdot x_2 \cdots x_j.$$

Finally, S sends $M'_j = \epsilon_{pw_{j+1}}(T'_j)$ to U_{j+1}.

U_i then intercepts M'_j, and mounts an offline dictionary attack as follows.

1. U_i guesses a possible password pw^*_{j+1}, and decrypts M'_j as

   $$D_{pw^*_{j+1}(M'_j)} = \{\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_{j+1}\}$$

2. U_i checks that $(\alpha_1)^{t_2} = \alpha_2$. If the check succeeds, then U_i confirms that $pw^*_{j+1} = pw_{j+1}$ because (ϵ, D) is an ideal cipher. Otherwise, go to step 1.

□

3.2 Security vulnerability in the EKE-M protocol

In the EKE-M protocol, a malicious participant U_j $(1 \leq j \leq n)$ can mount an online dictionary attack against any other participant U_i $(1 \leq i \leq n, i \neq j)$ without being detected by any entity.

To mount the attack, U_j initiates an instance of the protocol, and blocks all messages sent to U_i. In the first step, U_j guesses a possible password pw_i^* possessed by U_i, and impersonates U_i to broadcast $\epsilon_{pw_i^*}(g^{x_i})$. In the third step, U_j impersonates U_i to broadcast the key confirmation message $Auth_i = h(i||N)$. The attack succeeds based on the following lemma.

Lemma 3.2. *As a result of the above attack, U_j can test whether $pw_i^* = pw_i$, the protocol instance will successfully end, and all participants except U_i compute the same session key.*

Proof. In the EKE-M protocol, the session key material N is independently sent to each participant and the session key is computed based on N and other public information. So, it is straightforward to verify that the protocol instance will successfully end and all participants except U_i compute the same session key.

After intercepting $\epsilon_{pw_i}(g^{s_i})$ and $m_i = sk_i \oplus N$ sent by S, U_j first computes the guessed ephemeral session key between U_i and S as

$$sk_i^* = h(sid'||(D_{pw_i^*}(\epsilon_{pw_i}(g^{s_i})))^{x_i})$$

U_j then checks whether $N = m_i \oplus sk_i^*$. Based on the properties of the ideal cipher (ϵ, D), if the check succeeds then U_j can confirm that $pw_i^* = pw_i$; otherwise $pw_i^* \neq pw_i$. □

4 Conclusions

In this paper we have demonstrated certain security vulnerabilities in two password-based Diffie-Hellman key exchange protocols.

Acknowledgements

The author would like to express their deep appreciation for the valuable comments provided by Chris J. Mitchell.

References

[1] J. Black and P. Rogaway. Ciphers with arbitrary finite domains. In B. Preneel, editor, *Proceedings of the Cryptographer's Track at the RSA Conference 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 114–130. Springer, 2002.

[2] J. Byun and D. Lee. N-Party encrypted Diffie-Hellman key exchange using different passwords. In J. Ioannidis, A. D. Keromytis, and M. Yung, editors, *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA*, volume 3531 of *Lecture Notes in Computer Science*, pages 75–90. Springer-Verlag, 2005.

[3] D. Stinson. *Cryptography Theory and Practice*. CRC Press, Inc., second edition, 2002.