An Offline Dictionary Attack against a Three-Party Key Exchange Protocol

Junghyun Nam*, Kim-Kwang Raymond Choo**, Juryon Paik***, Dongho Won***

*Department of Computer Engineering, Konkuk University,
322 Danwol-dong, Chungju-si, Chungcheongbuk-do 380-701, Korea
jhnam@kku.ac.kr

**Information Assurance Research Group, Advanced Computing Research Centre,
University of South Australia, Mawson Lakes, SA 5095, Australia
raymond.choo@unisa.edu.au

***Department of Computer Engineering, Sungkyunkwan University,
300 Cheoncheon-dong, Jangan-gu, Suwon-si, Gyeonggi-do 440-746, Korea
wise96@ece.skku.ac.kr, dhwon@security.re.kr

Abstract

Despite all the research efforts made so far, the design of protocols for password-authenticated key exchange (PAKE) still remains a non-trivial task. One of the major challenges in designing such protocols is to protect low-entropy passwords from the notorious *dictionary attacks*. In this work, we revisit Abdalla and Pointcheval's three-party PAKE protocol presented in Financial Cryptography 2005, and demonstrate that the protocol is vulnerable to an off-line dictionary attack whereby a malicious client can find out the passwords of other clients.

Keywords : Password-authenticated key exchange (PAKE), three-party key exchange, password, dictionary attack.

1 Introduction

Unlike high-entropy cryptographic keys, passwords are drawn from a relatively small space like a dictionary, and are easy for humans to remember and use. Eventually, it is this advantage that password-based authentication has come to be widely used in today's computing environments. Bellovin and Merritt [6] was the first to consider how two parties, who only share a password, establish a common session key over a public network which might be fully controlled by an adversary. Due to the practical significance of password-based authentication, this initial work has been followed by a large number of password-authenticated key exchange (PAKE) protocols [5, 4, 9, 10].

A major threat to the design of secure PAKE protocols is off-line dictionary attacks [11, 1, 12]. Unlike on-line dictionary attacks where each password guess is verified via a new on-line transaction, off-line dictionary attacks allow the attacker to verify password guesses in an off-line manner using an automated program. Hence, off-line dictionary attacks are much more powerful and practical than on-line dictionary attacks, and must be prevented. The concern about off-line dictionary attacks is significantly increased in the three-party setting. In contrast to the two-party setting where the two parties are assumed to share the same password, the three-party setting assumes that each party, called a client, does not share any password with other clients, but holds their individual password which is shared only with a trusted authentication server. This means that three-party PAKE protocols should be designed to be secure even against attacks by a malicious insider who is a legitimate protocol participant [2, 14, 15].

In Financial Cryptography 2005, Abdalla and Pointcheval proposed a simple three-party PAKE protocol [3] which we denote by AP-3PAKE. Compared with the protocol of [2], the AP-3PAKE protocol is very efficient both in terms of computation and communication complexities. Moreover, unlike the protocols of [11, 14, 15], AP-3PAKE requires no use of cryptographic keys like server's public/private keys, allowing clients to manage only their passwords. Although the claimed proof of security for AP-3PAKE was found to be invalid [13], there have been so far no known offline dictionary attacks against the protocol. In this paper, we present a previously unpublished off-line dictionary attack against the AP-3PAKE protocol. Our dictionary attack can be viewed as an insider attack since it is mounted by one protocol participant against (the password of) the other participant. By identifying the vulnerability, we hope that similar structural mistakes can be avoided in the future design of three-party PAKE protocols.

2 A Review of the AP-3PAKE Protocol

This section revisits the AP-3PAKE protocol, Abdalla and Pointcheval's three-party PAKE protocol [3]. The protocol participants consist of a single server S and two clients A and B. The clients A and B wish to establish a session key between them while the server S exists to provide the clients with authentication services. Let pw_A and pw_B be the passwords of A and B, respectively. Each client's password is assumed to be shared with the authentication server S via a secure channel. The followings are the public system parameters used in the protocol.

- A finite cyclic group $\mathbb{G}$ of prime order q and a random generator g of the group $\mathbb{G}$.
- A hash function H modeled as a random oracle. The outputs of H are κ-bit strings, where κ is a security parameter representing the length of session keys.
- Two hash functions G_1 and G_2 modeled as random oracles. The outputs of G_1 and G_2 are the elements of the cyclic group $\mathbb{G}$.

With the system parameters established, the AP-3PAKE protocol runs in two communication rounds as follows:

$$
\begin{array}{ccc}
\boxed{A} & \boxed{S} & \boxed{B} \\
(pw_A) & (pw_A, pw_B) & (pw_B) \\
x \in \mathbb{Z}_q, X = g^x & & y \in \mathbb{Z}_q, Y = g^y \\
pw_{A,1} = G_1(A, B, pw_A) & & pw_{B,1} = G_1(A, B, pw_B) \\
X^* = X \cdot pw_{A,1} & & Y^* = Y \cdot pw_{B,1} \\
& \xrightarrow{\quad X^* \quad} \; \xleftarrow{\quad Y^* \quad} & \\
& pw_{A,1} = G_1(A, B, pw_A) & \\
& pw_{B,1} = G_1(A, B, pw_B) & \\
& X = X^*/pw_{A,1} & \\
& Y = Y^*/pw_{B,1} & \\
& z \in \mathbb{Z}_q, R \in \{0,1\}^\gamma & \\
& \overline{X} = X^z, \overline{Y} = Y^z & \\
& pw_{A,2} = G_2(A, B, R, pw_A, X^*) & \\
& pw_{B,2} = G_2(A, B, R, pw_B, Y^*) & \\
& \overline{X}^* = \overline{X} \cdot pw_{B,2} & \\
& \overline{Y}^* = \overline{Y} \cdot pw_{A,2} & \\
& \xleftarrow{R, Y^*, \overline{X}^*, \overline{Y}^*} \; \xrightarrow{R, X^*, \overline{X}^*, \overline{Y}^*} & \\
pw_{A,2} = G_2(A, B, R, pw_A, X^*) & & pw_{B,2} = G_2(A, B, R, pw_B, Y^*) \\
\overline{Y} = \overline{Y}^*/pw_{A,2} & & \overline{X} = \overline{X}^*/pw_{B,2} \\
K = \overline{Y}^x & & K = \overline{X}^y \\
T = R \| X^* \| Y^* \| \overline{X}^* \| \overline{Y}^* & & T = R \| X^* \| Y^* \| \overline{X}^* \| \overline{Y}^* \\
SK = H(A \| B \| S \| T \| K) & & SK = H(A \| B \| S \| T \| K)
\end{array}
$$

Figure 1: Abdalla and Pointcheval's three-party PAKE protocol

1. Client A chooses a random $x \in \mathbb{Z}_q$ and computes $X = g^x$ and $pw_{A,1} = G_1(A, B, pw_A)$.[1] Then A computes $X^* = X \cdot pw_{A,1}$ and sends X^* to the server S.

2. Similarly, client B chooses a random $y \in \mathbb{Z}_q$ and computes $Y = g^y$ and $pw_{B,1} = G_1(A, B, pw_B)$. Then B computes $Y^* = Y \cdot pw_{B,1}$ and sends Y^*

[1] In the protocol description of [3], the identities of the clients were incorrectly omitted from the input of the hash functions G_1 and G_2. Abdalla and Pointcheval corrected this omission in the full version of the paper, which is available at `http://www.di.ens.fr/~mabdalla/pubs.html`.

to S.

3. Upon receiving X^* and Y^*, S first recovers X and Y by computing $X = X^*/G_1(A, B, pw_A)$ and $Y = Y^*/G_1(A, B, pw_B)$. Next, S selects a random element $z \in \mathbb{Z}_q$ and a random string $R \in \{0, 1\}^\gamma$, where γ is a security parameter which determines the bit-length of R. S then computes

$$\begin{aligned}
\overline{X} &= X^z, \\
\overline{Y} &= Y^z, \\
pw_{A,2} &= G_2(A, B, R, pw_A, X^*), \\
pw_{B,2} &= G_2(A, B, R, pw_B, Y^*), \\
\overline{X}^* &= \overline{X} \cdot pw_{B,2}, \\
\overline{Y}^* &= \overline{Y} \cdot pw_{A,2},
\end{aligned}$$

and sends $\langle R, Y^*, \overline{X}^*, \overline{Y}^* \rangle$ and $\langle R, X^*, \overline{X}^*, \overline{Y}^* \rangle$ to A and B, respectively.

4. After receiving $\langle R, Y^*, \overline{X}^*, \overline{Y}^* \rangle$ from S, A computes

$$\begin{aligned}
pw_{A,2} &= G_2(A, B, R, pw_A, X^*), \\
\overline{Y} &= \left(\frac{\overline{Y}^*}{pw_{A,2}}\right), \\
K &= \overline{Y}^x.
\end{aligned}$$

Then A defines the transcript $T = R \| X^* \| Y^* \| \overline{X}^* \| \overline{Y}^*$ and computes the session key $SK = H(A \| B \| S \| T \| K)$.

5. With $\langle R, X^*, \overline{X}^*, \overline{Y}^* \rangle$ received from S, B computes

$$\begin{aligned}
pw_{B,2} &= G_2(A, B, R, pw_B, Y^*), \\
\overline{X} &= \left(\frac{\overline{X}^*}{pw_{B,2}}\right), \\
K &= \overline{X}^y.
\end{aligned}$$

Then B defines the transcript $T = R \| X^* \| Y^* \| \overline{X}^* \| \overline{Y}^*$ and computes the session key $SK = H(A \| B \| S \| T \| K)$.

The correctness of AP-3PAKE can be easily verified from $K = \overline{Y}^x = \overline{X}^y = g^{xyz}$. Figure 1 shows a high-level depiction of AP-3PAKE.

3 An Off-Line Dictionary Attack against AP-3PAKE

The AP-3PAKE protocol seems to be secure against off-line dictionary attacks if we only consider honest clients who stick to the protocol specification. But in the 3-party setting, there may be malicious clients who deviate from the protocol. Indeed, the existence of insider attacks by malicious clients is one of the major differences between the 2-party and the 3-party settings [2, 14, 15].

We here show that AP-3PAKE is not secure against an off-line dictionary attack in the presence of a malicious client.

Our attack exploits the fact that, once the session key SK has been established, the clients A and B will exchange their subsequent messages that are generated using the key SK. Let msg_{SK} be the first such message. Without loss of generality and for simplicity, we assume that it is the client A who generates/sends msg_{SK}. Then, our off-line dictionary attack can be mounted by B against A's password. (We note, however, that our attack also works even when msg_{SK} is generated/sent by B. In this case, the attack can be mounted by A against B's password.) The attack proceeds as follows:

Step 1. The attacker B runs the protocol with the client A and the server S. In this run, everything proceeds as specified by the protocol, except that:

B replaces the message $\langle R, Y^*, \overline{X}^*, \overline{Y}^* \rangle$ from S to A with the forged message $\langle R, Y^*, \overline{X}^*, \widetilde{Y}^* \rangle$, where $\widetilde{Y}^* = \overline{Y}^* \cdot Y$.

As a result of the replacement, A will compute its session key as $SK = H(A\|B\|S\|\widetilde{T}\|\widetilde{K})$, where $\widetilde{T} = R\|X^*\|Y^*\|\overline{X}^*\|\widetilde{Y}^*$ and

$$\begin{aligned}
\widetilde{K} &= \Big(\frac{\widetilde{Y}^*}{pw_{A,2}}\Big)^x \\
&= \Big(\frac{\overline{Y}^* \cdot Y}{pw_{A,2}}\Big)^x \\
&= \Big(\frac{\overline{Y} \cdot pw_{A,2} \cdot Y}{pw_{A,2}}\Big)^x \\
&= (\overline{Y} \cdot Y)^x \\
&= g^{xyz} \cdot g^{xy}.
\end{aligned}$$

Step 2. Now since A has computed its session key SK, A will generate msg_{SK} using the key SK in one of the following three scenarios:

- Key Confirmation: If key confirmation is required, A will send a confirmation message $cfrm$ computed as a function of SK. Assume, without affecting our result, that the well-known technique of [5, 7] is used for key confirmation. Then $cfrm$ is computed as $cfrm = H(SK\|v)$ for some known value v. In this case, $msg_{SK} = cfrm$.
- Message Authentication: A typical usage of session keys is to authenticate subsequent messages. Let m be a message to be sent from A to B. If m has to be authenticated, A will generate a message authentication code (MAC) σ for the message m under the session key SK by running a MAC generation algorithm Mac (i.e., $\sigma = Mac(SK, m)$), and then will send the message/MAC pair $mTag = (m, \sigma)$ to B. In this case, $msg_{SK} = mTag$.
- Message Encryption: Another typical usage of session keys is to encrypt subsequent messages. Suppose that A has to send a message m in an encrypted form. Then, A is likely to run a symmetric encryption algorithm Enc with key SK and input message m to get back a ciphertext $cprt$ (i.e., $cprt = Enc(SK, m)$). The ciphertext $cprt$ can then be transmitted to B. In this case, $msg_{SK} = cprt$.

Once in possession of $msg_{SK} \in \{cfrm, mTag, cprt\}$, the attacker B aborts the protocol session alleging that session-key computation has failed due to an unexpected error.

Step 3. Using the message $\langle R, X^*, \overline{X}^*, \overline{Y}^* \rangle$ received from S, B computes

$$\begin{aligned} K &= \Big(\frac{\overline{X}^*}{pw_{B,2}}\Big)^y \\ &= g^{xyz}. \end{aligned}$$

Step 4. Next, B makes a guess pw'_A for the password pw_A and computes $X' = X^*/G_1(A, B, pw'_A)$, $\widetilde{K}' = K \cdot X'^y$, and $SK' = H(A\|B\|S\|\widetilde{T}\|\widetilde{K}')$ with $\widetilde{T}$ defined as above. B then performs one of the following three computations depending on the type of msg_{SK}:

$$\begin{aligned} cfrm' &= H(SK'\|v) && \text{if } msg_{SK} = cfrm, \\ \sigma' &= Mac(SK', m) && \text{if } msg_{SK} = mTag, \\ m' &= Dec(SK', cprt) && \text{if } msg_{SK} = cprt. \end{aligned}$$

Here, Dec is the symmetric decryption algorithm associated with Enc.

Step 5. Now, B verifies the correctness of pw'_A by checking the appropriate one of the following conditions:

$$\begin{aligned} cfrm' &\overset{?}{=} cfrm && \text{if } msg_{SK} = cfrm, \\ \sigma' &\overset{?}{=} \sigma && \text{if } msg_{SK} = mTag, \\ \text{Is } m' &\text{ meaningful?} && \text{if } msg_{SK} = cprt. \end{aligned}$$

Notice that if pw'_A and pw_A are equal, then the conditions ought to be satisfied. Although verifying the third condition "Is m' meaningful?" takes longer than verifying the first/second conditions, it can still be done automatically by a computer program and can be substantially speed up by tuning the condition.

Step 6. B repeats Steps 4 and 5 until a correct password is found.

The attack above is an off-line dictionary attack since the steps for verifying password guesses can be performed in an off-line manner by an automated computer program.

References

[1] Abdalla M., Bresson E., Chevassut O., and Pointcheval D., Password-Based Group Key Exchange in a Constant Number of Rounds, In PKC 2006, LNCS 3958, 427–442, 2006.

[2] Abdalla M., Fouque P., and Pointcheval D., Password-Based Authenticated Key Exchange in the Three-Party Setting, In PKC 2005, LNCS 3386, 65–84, 2005.

[3] Abdalla M. and Pointcheval D., Interactive Diffie-Hellman Assumptions with Applications to Password-Based Authentication, In FC 2005, LNCS 3570, 341–356, 2005.

[4] Abdalla M. and Pointcheval D., Simple Password-Based Encrypted Key Exchange Protocols, In CT-RSA 2005, LNCS 3376, 191–208, 2005.

[5] Bellare M., Pointcheval D., and Rogaway P., Authenticated Key Exchange Secure against Dictionary Attacks, In EUROCRYPT 2000, LNCS 1807, 139–155, 2000.

[6] Bellovin S. and Merritt M., Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks, In 1992 IEEE Symposium on Research in Security and Privacy, 72–84, 1992.

[7] Bresson E., Chevassut O., Pointcheval D., and Quisquater J., Provably Authenticated Group Diffie-Hellman Key Exchange, In 8th ACM Conference on Computer and Communications Security, 255–264, 2001.

[8] Coron J., Patarin J., and Seurin Y., The Random Oracle Model and the Ideal Cipher Model are Equivalent, In CRYPTO 2008, LNCS 5157, 1–20, 2008.

[9] Katz J., Ostrovsky R., and Yung M., Efficient and Secure Authenticated Key Exchange using Weak Passwords, *Journal of the ACM*, 57(2009), 78–116.

[10] Katz J. and Vaikuntanathan V., Round-Optimal Password-Based Authenticated Key Exchange, In TCC 2011, LNCS 6597, 293–310, 2011.

[11] Lin C., Sun H., and Hwang T., Three-Party Encrypted Key Exchange: Attacks and a Solution, *ACM SIGOPS Operating Systems Review*, 34(4)(2000), 12–20.

[12] Nam J., Paik J., Kang H., Kim U., and Won D., An Off-Line Dictionary Attack on a Simple Three-Party Key Exchange Protocol, *IEEE Communications Letters*, 13(3)(2009), 205–207.

[13] Szydlo M., A Note on Chosen-Basis Decisional Diffie-Hellman Assumptions, In FC 2006, LNCS 4107, 166–170, 2006.

[14] Yoneyama K., Efficient and Strongly Secure Password-Based Server Aided Key Exchange, In INDOCRYPT 2008, LNCS 5365, 172–184, 2008.

[15] Zhao J. and Gu D., Provably Secure Three-Party Password-Based Authenticated Key Exchange Protocol, *Information Sciences*, 184(2012), 310–323.