LETTER Special Section on Cryptography and Information Security

Attack on the Sun-Chen-Hwang’s Three-Party Key Agreement Protocols Using Passwords*

Junghyun NAM[a], Seungjoo KIM[b], Nonmembers, and Dongho WON[c], Member

SUMMARY We show that Sun-Chen-Hwang’s three-party key agreement protocols using passwords are insecure against an active adversary. Further, we recommend a small change to the protocols that fixes the security problem.

key words: three-party key agreement, password, verifier

1. Introduction

The possibility of secure password-authenticated key exchange was recognized in the work of Bellovin and Merritt [1], which shows how to bootstrap a high-entropy cryptographic key from a weak, low-entropy password. Due in large part to the practical significance of password-based authentication, this initial work has been extended to a number of settings, including a three-party model where an authentication server exists to help two communicating parties establish a common session key. Roughly a decade ago, Steiner et al. [4] proposed a three-party protocol for password-based key agreement which builds on the earlier protocol, known as encrypted key exchange, or EKE, proposed by Bellovin and Merritt [1] in the two-party setting. However, Steiner et al.’s three-party EKE turned out to be vulnerable not only to undetectable on-line password guessing attacks but also to off-line password guessing attacks [2], [3]. Recently, Sun et al. [5] have presented two new improved versions of Steiner et al.’s three-party EKE, one for password-based authenticated key agreement (SCH.PAKA) and one for verifier-based authenticated key agreement (SCH.V AKA). To prevent password guessing attacks, they consider a hybrid model in which the clients store the server’s public key in addition to sharing a password (or a password verifier) with the server. However, these seemingly well-designed protocols, while focusing on providing security against password guessing attacks, are insecure against an active adversary who can insert, delete and delay messages. In this letter, we present an active attack that enables an adversary to completely compromise the authentication mechanism of Sun et al.’s protocols. We also present a simple patch to the authentication flaw with the protocols.

2. Review of Sun et al.’s Protocols

We assume that two passwords $P_A$ and $P_B$ respectively of $A$ and $B$ are known to $S$ via a secure channel. The protocol description is as follows (see also Fig. 1):

1. The client $A$ chooses a random number $N_A$ and computes $g^{N_A}$. Then $A$ sends $C_A = E_{pk}(g^{N_A}, P_A)$ to $B$, where $E_{pk}(g^{N_A}, P_A)$ denotes the ciphertext of the message $(g^{N_A}, P_A)$ encrypted using the public-key encryption algorithm $E$ under the public key $pk$ of $S$.

2. The client $B$ chooses a random number $N_B$, computes $g^{N_B}$, and sends to $S$ the ciphertexts $C_A$ and $C_B = E_{pk}(g^{N_B}, P_B)$.

3. The server $S$ verifies the validity of the passwords by decrypting the ciphertexts $C_A$ and $C_B$. If the verification succeeds, $S$ chooses three random numbers $a$, $b$ and $N_S$, and computes

$R_A = (g^{N_A} \cdot g^{bN_S})$,

$R_B = (g^{N_B} \cdot g^{aN_S})$.

\[ A \xrightarrow{C_A = E_{pk}(g^{N_A}, P_A)} B \xrightarrow{C_A, C_B, R_A, R_B} S \]

\[ R_A, a, R_B, b \]

\[ C_A, C_B = E_K(C_A) \]

\[ C_{AB} = E_K(C_{BA}) \]

Fig. 1 The SCH.PAKA protocol.
Here, $a$ and $b$ can be chosen to be such that $a, b \ll N_S$. $S$ then sends $(R_A, a, R_B, b)$ to $B$.

4. $B$ computes the session key $K$ as

$$K = R_B^{N_b+b} = g^{(N_b+b)/(N_b+a)}$$

and sends $(R_A, a, C_{BA} = E_K(C_A))$ to $A$, where $E_K(C_A)$ denotes the ciphertext of $C_A$ encrypted using symmetric encryption algorithm $E$ under the key $K$.

5. Now, $A$ computes the session key $K$ as

$$K = R_A^{N_a+a} = g^{(N_a+a)/(N_a+b)}$$

and verifies it through decryption of $C_{BA}$. Finally, $A$ sends $C_{AB} = E_K(C_{BA})$ to $B$, who will in turn decrypt it to check whether $A$ has computed the correct key.

### 2.2 The SCH.VAKA Protocol

Let $x_A$ and $x_B$ be two private integers derived in any predetermined way respectively from $P_A$ and $P_B$. The verifier $v_A = g^{x_A}$ (resp., $v_B = g^{x_B}$) of $P_A$ (resp., $P_B$) is assumed to be known to $S$ via a secure channel. The protocol proceeds similarly as the SCH.PAKA protocol, but using the verifiers in place of the passwords. The details of the protocol are as follows:

1. The client $A$ chooses a random number $N_A$, computes $g^{N_A}$, and sends to $B$ the ciphertext $C_A = E_{pk}(ID_A, g^{N_A}, v_A)$.

2. The client $B$ chooses a random number $N_B$, computes $g^{N_B}$, and sends to $S$ the ciphertexts $C_A$ and $C_B = E_{pk}(ID_B, g^{N_B}, v_B)$.

3. After decrypting $C_A$ and $C_B$ and checking the verifiers $v_A$ and $v_B$, the server $S$ chooses three random numbers $a, b$ and $N_S$, and computes

$$R_A = (g^{N_A} \cdot v_B)^N_S,$$

$$R_B = (g^{N_B} \cdot v_A)^N_S.$$ $S$ then sends $(R_A, a, R_B, b)$ to $A$.

4. $A$ computes the session key $K$ as

$$K = R_A^{N_A+x_A} = g^{(N_A+x_A)/(N_A+a)}$$

and sends $(R_A^b, b, C_{AB} = E_K(C_A))$ to $B$.

5. Now, $B$ computes the session key $K$ as

$$K = R_B^{N_b+x_b} = g^{(N_b+x_b)/(N_b+b)}$$

and verifies it through decryption of $C_{AB}$. Lastly, $C_{BA} = E_K(C_{BA})$ is sent to $A$ who can then confirm that $B$ has computed the same session key $K$.

### 3. Security Analysis

We now show that both of Sun et al.'s three-party key agreement protocols are insecure against an active adversary.

#### 3.1 Attack on the SCH.PAKA Protocol

We assume that the adversary $M$ is a legitimate user who is registered with the authentication server $S$ and thus is able to set up normal protocol sessions with other users. The goal of adversary $M$ is to share a key with $A$ by masquerading as $B$ and to share another key with $B$ by masquerading as $A$. A high-level depiction of the attack is given in Fig. 2, where a dashed line indicates that the corresponding flow is blocked by $M$ from reaching the destination. A more detailed description of the attack is as follows:

1. As a preliminary step, $M$ chooses two random numbers $N_M$ and $N_M^\prime$ and computes $g^{N_M}$, $g^{N_M^\prime}$, $C_M = E_{pk}(g^{N_M}$, $R_M^\prime$) and $C'_M = E_{pk}(g^{N_M^\prime}$, $R_M$).

2. The adversary $M$ launches the attack by intercepting the message going to the server (i.e., the message $(C_A, C_B)$ sent from $B$ to $S$). After intercepting the message, $M$ immediately sends two separate messages $(C_A, C'_M)$ and $(C_B, C_M^\prime)$ to $S$ alleging that she wants to establish two concurrent sessions, each with $A$ and $B$.

3. Since $(C_A, C_M)$ and $(C_B, C'_M)$ are both valid, the server $S$ constructs two response messages, one for the session between $A$ and $M$ and the other for the session between $B$ and $M$, as specified in the protocol; it chooses two pairs of triple random numbers $(a, m, N_S)$ and $(b, m^\prime, N_S^\prime)$ and computes

![Fig. 2 An attack on the SCH.PAKA protocol.](image)

\[ R_b \] was incorrectly stated as $R_a$ in step 4 of Table 3 of the original paper. We have corrected this typographical error.
\[ R_A = (g^{N_A \cdot g^m})^{N_S}, \]
\[ R_M = (g^{N_M \cdot g^m})^{N_S}, \]
\[ R_B = (g^{N_B \cdot g^m})^{N_S}, \]
\[ R_{M}^{*} = (g^{N_{M} \cdot g^m})^{N_S}. \]

S then sends back two messages \((R_A, a, R_M, m)\) and \((R_B, b, R_{M}^{*}, m')\) to the adversary \(M\).

4. Having received these messages, \(M\) computes two session keys \(K\) and \(K'\) to be shared respectively with \(A\) and \(B\) as follows:
\[ K = R_{M}^{N_{M} + m} = g^{N_{A} + m(N_{S} \cdot N_{M} + m)}, \]
\[ K' = R_{M}^{N_{M} + m'} = g^{N_{B} + b(N_{S} \cdot N_{M} + m')}. \]

\(M\) then sends the message \((R_{M}^{*}, m', R_B, b)\) to \(M\) alleging that it comes from \(S\).

5. Since \(B\) thinks that \((R_{M}^{*}, m', R_B, b)\) is the response to \((C_A, C_B)\) from \(S\), he computes his session key as
\[ K' = R_{B}^{N_{A} + b} = g^{N_{A} + m(N_{S} \cdot N_{B} + b)} \]
and sends the message \((R_{M}^{*}, m', C_{BA} = E_{K'}(C_{A}))\) to \(A\). But this message is replaced with \((R_A, a, C_{MA} = E_{K}(C_{A}))\) by the adversary.

6. Now \(A\), believing that \((R_A, a, C_{MA})\) is from \(B\), computes her session key as
\[ K = R_{A}^{N_{A} + m} = g^{N_{A} + m(N_{S} \cdot N_{A} + m)}, \]
verifies that the decryption of \(C_{MA}\) under \(K\) is equal to \(C_A\), and then sends \(C_{AB} = E_{K}(C_{BA})\) to \(B\). But this message is also replaced with \((C_{MB} = E_{K}(C_{BA}))\) by the adversary.

Through the attack, the authentication mechanism of the protocol is completely compromised. At the end of this scenario, the client \(A\) believes that she has established a secure session with \(B\) sharing a secret key \(K\), while in fact she has shared the key with \(M\). Similarly, \(B\) thinks that he has shared with \(A\) a session key \(K'\) which indeed is shared with \(M\). As a result, the adversary \(M\) can not only access and relay any confidential communications between \(A\) and \(B\), but can also send arbitrary messages for her own benefit impersonating one of them to the other.

We note that in the SCH.PAKA protocol, a client’s ciphertext sent to \(S\) can be paired with an unintended ciphertext generated by an arbitrary client. This oversight creates the vulnerability of the protocol to such an attack given above. But fortunately, it is trivial to fix this flaw. It suffices to modify the computations of \(C_A\) and \(C_B\) to the following:
\[ C_A = E_{pk}(ID_A, ID_B, g^{N_A}, P_A), \]
\[ C_B = E_{pk}(ID_B, ID_A, g^{N_B}, P_B). \]

With this modification, the messages sent to the server in each run of the protocol become bounded to the identities of the clients participating in that run, and therefore our attack would be impossible.

3.2 Attack on the SCH.V AKA Protocol

The attack on the SCH.V AKA protocol is essentially same as the attack on the SCH.PAKA protocol. The same attack works because there is still no way for the server to verify whether the two incoming ciphertexts are paired honestly. Note that as in \(E_{pk}(ID_A, g^{N_A}, v_A)\), the inclusion of the sender’s identity as part of the plaintext does not play any role to prevent a client’s ciphertext from being paired with an unintended ciphertext generated by the adversary. Again, the goal of adversary \(M\) is to share a session key with each client separately, while deluding the clients into believing that they have established a secure session between them.

The attack scenario is as follows:

1. To begin with, the adversary \(M\) chooses two random numbers \(N_M\) and \(N_{M}^{*}\) and computes \(g^{N_{M}}\), \(g^{N_{M}^{*}}\), \(C_M = E_{pk}(ID_M, g^{N_{M}}, v_M)\) and \(C_{M}^{*} = E_{pk}(ID_M, g^{N_{M}^{*}}, v_{M})\).

2. \(M\) constructs two message pairs \((C_M, C_{A})\) and \((C_{M}^{*}, C_{B})\) by intercepting \((C_A, C_B)\) sent to \(S\) by \(B\). Then \(M\) sends to \(S\) the message \((C_{M}, C_A)\) alleging that it comes from \(A\), and the message \((C_{M}^{*}, C_B)\) alleging that it comes from \(B\).

3. Since \(S\) thinks that both \(A\) and \(B\) want to establish a session with \(S\), it sends to \(M\) the two messages \((R_M, m, R_A, a)\) and \((R_{M}^{*}, m', R_B, b)\) in response, respectively, to \((C_M, C_A)\) and \((C_{M}^{*}, C_B)\) such that
\[ R_M = (g^{N_{M}} \cdot v_A)^{N_{S}}; \]
\[ R_A = (g^{m_{N_{M}} \cdot v_B})^{N_{S}}; \]
\[ R_{M}^{*} = (g^{N_{M}^{*}} \cdot v_B)^{N_{S}}; \]
\[ R_B = (g^{m_{N_{M}^{*}}} \cdot v_{M})^{N_{S}}. \]

4. Upon receiving the messages from \(S\), the adversary \(M\) computes two session keys \(K\) and \(K'\) to be shared respectively with \(A\) and \(B\) as follows:
\[ K = R_{M}^{m_{N_{M}}} \cdot x_M = g^{m_{N_{M}} \cdot x_M}; \]
\[ K' = R_{M}^{m_{N_{M}^{*}}} \cdot x_M = g^{m_{N_{M}^{*}} \cdot x_M}. \]

\(M\) then sends the message \((R_A, a, R_M, m)\) to \(A\) alleging that it comes from \(S\).

5. After receiving \((R_A, a, R_M, m)\), the client \(A\) computes her session key \(K\) as
\[ K = R_{A}^{m_{N_{M}} + x_A} = g^{m_{N_{M}} \cdot x_A}; \]
and sends the message \((R_M, m, C_{AB} = E_{K}(C_{A}))\) to \(B\). But, \(M\) intercepts this message and instead sends \((R_B, b, C_{MB} = E_{K}(C_{A}))\) to \(B\).
6. The client \( B \), upon receiving \((R_B, b, C_{MB})\), computes his session key as

\[
K' = R^{bN_B + x_B} = g^{(mN'_A + sx_A)(N'_S bN_B + x_B)}
\]

and verifies that the decryption of \( C_{MB} \) under \( K' \) is equal to \( C_A \). \( B \) then sends the response \( C_{BA} = E_{K'}(C_{MB}) \) to \( A \). But, \( M \) intercepts this message and instead sends \( C_{MA} = E_K(C_{AB}) \) to \( A \).

This scenario leads to the same consequence as stated at the end of the attack scenario for the SCH.PAKA protocol, and the exact same solution as given there can be applied to this case. That is, as a countermeasure to our attack, the computations of \( C_A \) and \( C_B \) are modified slightly as follows:

\[
C_A = E_{pk}(IDA, ID_B, g^{N_A}, v_A),
\]

\[
C_B = E_{pk}(ID_B, ID_A, g^{N_B}, v_B).
\]

Finally, we remark that the four-round verifier-based protocol (the optimal round scheme) described in Sect. 4.2 of [5] also suffers from a similar problem as the one studied above. We do not detail it in this letter due to the similarity.

4. Conclusion

We have shown that the three-party key agreement protocols proposed by Sun et al. [5] are susceptible to an attack mounted by an active adversary. Fortunately, the security hole identified here can be easily patched by integrating all participants’ identities as part of the message being encrypted by each client.

References


