J. Algebra Comb. Discrete Appl.
3(1) • 31–36

Received: 31 October 2015
Accepted: 30 November 2015

Journal of Algebra Combinatorics Discrete Structures and Applications

An active attack on a multiparty key exchange protocol

Research Article

Reto Schnyder, Juan Antonio López-Ramos, Joachim Rosenthal, Davide Schipani

Abstract: The multiparty key exchange introduced in Steiner et al. and presented in more general form by the authors is known to be secure against passive attacks. In this paper, an active attack is presented assuming malicious control of the communications of the last two users for the duration of only the key exchange.

2010 MSC: 94A60

Keywords: Multiparty key exchange, Active attacks, Group actions

1. Introduction

The increased use of light and mobile devices has led to the study of the so called mobile ad hoc networks. These are created, operated and managed by the nodes themselves and therefore are solely dependent upon the cooperative and trusting nature of the nodes. The ad hoc property of these mobile networks implies that the network is formed in an unplanned manner to meet an immediate demand and specific goal, and that the nodes are continuously joining or leaving the network. Thus, key management in this type of networks is a very important issue and has been the aim of numerous works since then (see [1] or [5] and their references).

One of the most widely known such schemes is due to Steiner et al. and is known as Cliques (cf. [4]). Cliques is a multiparty key exchange protocol generalizing the Diffie-Hellman key exchange based on the discrete logarithm problem. It is composed of an initial key agreement (IKA) to set up a first common key and an auxiliary key agreement (AKA) in order to refresh the key at any later stage.

The Research was supported in part by the Swiss National Science Foundation under grant No. 149716. Juan Antonio López-Ramos is partially suppported by Ministerio de Educacion, Cultura y Deporte grant "Salvador de Madariaga" PRX14/00121, Ministerio de Economia y Competitividad grant MTM2014-54439 and Junta de Andalucia (FQM0211). Reto Schnyder is supported by Armasuisse.

Reto Schnyder, Joachim Rosenthal, Davide Schipani (Corresponding Author); Institute of Mathematics, University of Zurich, Switzerland (email: reto.schnyder@math.uzh.ch, rosenthal@math.uzh.ch, davide.schipani@math.uzh.ch).

Juan Antonio López-Ramos; Department of Mathematics, University of Almeria, Spain (email: jlopez@ual.es).

In [3], the authors propose a systematic way for analyzing protocol suites which extend the Diffie-Hellman key-exchange scheme to a group setting. They find interesting attacks which exploit algebraic properties of Diffie-Hellman exponentiation. However, our attack uses a different approach that exploits a weakness of a specific protocol and allows for prolonged eavesdropping.

We will consider in particular one of the proposed initial key agreements referred to (in [4]) as IKA.2. The authors generalize these schemes in [2], considering a general action on a semigroup, and this is how IKA.2 is presented below.

We will then show an active attack on this protocol that requires control of the communications of two particular parties for only the duration of the key exchange. That is, unlike in a regular man-in-the-middle attack, it is not necessary for the attacker to control the communications after the key exchange in order to translate messages, since all users are made to agree on the same key.

Although it is not possible for the attacker to keep a copy of the key after the users initiate AKA operations, we will show how she can avoid being noticed at that point.

2. An initial key agreement protocol

The protocol below gives n users the possibility to share an initial common key built using their private keys. A proof of its correctness and security against passive attacks can be found in [2, 4], assuming the Diffie-Hellman problem is hard for the given group action.

Suppose we have n users $\mathcal{U}_1, \ldots, \mathcal{U}_n$ who wish to agree upon a common key.

Let G be an abelian group, written multiplicatively. Let S be a set, and suppose we have a group action

$$\begin{aligned} G \times S &\to S \\ (g, s) &\mapsto g \cdot s. \end{aligned}$$

The users publicly agree on a common element $C_0 = s \in S$, and for each $i = 1, \ldots, n$, the user $\mathcal{U}_i$ selects a secret group element $g_i \in G$.

The protocol proceeds as follows:

1. For $i = 1, \ldots, n-2$, $\mathcal{U}_i$ sends to $\mathcal{U}_{i+1}$ the message $C_i = g_i \cdot C_{i-1}$.
2. $\mathcal{U}_{n-1}$ broadcasts $C_{n-1} = g_{n-1} \cdot C_{n-2}$ to the other users $\mathcal{U}_1, \ldots, \mathcal{U}_{n-2}, \mathcal{U}_n$.
3. $\mathcal{U}_n$ computes the shared key $K = g_n \cdot C_{n-1}$.
4. For $i = 1, \ldots, n-1$, $\mathcal{U}_i$ sends $D_i = g_i^{-1} \cdot C_{n-1}$ to $\mathcal{U}_n$.
5. $\mathcal{U}_n$ broadcasts $\{g_n \cdot D_1, g_n \cdot D_2, \ldots, g_n \cdot D_{n-1}, C_{n-1}\}$ to $\mathcal{U}_i$, $i = 1, \ldots, n-1$.
6. For $i = 1, \ldots, n-1$, $\mathcal{U}_i$ computes the shared key $K = g_i \cdot (g_n \cdot D_i)$.

It is easy to see that for $i = 1, \ldots, n-1$, we have that

$$C_i = \left(\prod_{j=1}^{i} g_j \right) \cdot s,$$

$$D_i = \left(\prod_{\substack{j=1 \\ j \neq i}}^{n-1} g_j \right) \cdot s,$$

and finally

$$K = C_n = \left(\prod_{j=1}^{n} g_j \right) \cdot s.$$

From the above, we can also observe that C_{n-1} is not needed by any user to recover the session key K. However, this information is disclosed for future rekeying purposes, as we will see later.

Example 2.1. *Let $\mathbb{F}_q$ be a finite field. Let us consider an element g of prime order p, generating the subgroup $S \subset \mathbb{F}_q^*$. Then the action $\Phi\colon \mathbb{Z}_p^* \times S \to S$ defined by $\Phi(x,h) = h^x$ provides the Initial Key Agreement protocol introduced in [4, Section 4.2] as IKA.2.*

Example 2.2. *Let us denote by ε the group of points of an elliptic curve of prime order p. Then the action $\Phi\colon \mathbb{Z}_p^* \times \varepsilon \to \varepsilon$ defined by $\Phi(x,P) = xP$ gives an elliptic curve version of IKA.2 cited above.*

3. An active attack on the initial key agreement

We describe an active attack on the protocol of the preceding section. Suppose that the attacker $\mathcal{M}$ wants the users $\mathcal{U}_1, \ldots, \mathcal{U}_n$ to agree on a shared key as usual, except that she is in possession of the key as well.

In order to carry out our attack, $\mathcal{M}$ needs to have full control over the communication of the users $\mathcal{U}_{n-1}$ and $\mathcal{U}_n$ for the duration of the key exchange. However, unlike in a regular man-in-the-middle attack, she does not need to maintain this control after the key exchange is completed.

In the beginning, $\mathcal{M}$ chooses her own secret group element $\hat{g} \in G$. She then proceeds as follows:

1. Step (1) is carried out as usual.
2. $\mathcal{M}$ intercepts the broadcast of $\mathcal{U}_{n-1}$ during step (2) and remembers the value C_{n-1}. At this point, all users except for $\mathcal{U}_{n-1}$ are sitting in step (2), waiting for the broadcast that was halted.
3. $\mathcal{U}_{n-1}$ proceeds to step (4), where he sends $g_{n-1}^{-1} \cdot C_{n-1} = C_{n-2}$ to $\mathcal{U}_n$. This is also intercepted by $\mathcal{M}$. $\mathcal{U}_{n-1}$ is now waiting in step (5).
4. $\mathcal{M}$ now makes $\mathcal{U}_n$ believe that he received the broadcast of step (2), but actually sends him $\hat{g} \cdot C_{n-1}$. At this point, $\mathcal{U}_n$ computes the shared key $K = g_n \hat{g} \cdot C_{n-1}$ and waits in step (4).
5. $\mathcal{M}$ now sends to $\mathcal{U}_n$ the values $\{m_1, \ldots, m_{n-3}, C_{n-2}, C_{n-1}\}$, pretending that they were sent by the other users in step (4). The m_i are random elements of the orbit $G \cdot s$.
6. In step (5), $\mathcal{U}_n$ sends back, among others, the values $g_n \cdot C_{n-2}$ and $g_n \cdot C_{n-1}$, which $\mathcal{M}$ intercepts. The user $\mathcal{U}_n$ is now finished, and $\mathcal{M}$ can compute the shared key $K = \hat{g} g_n \cdot C_{n-1}$.
7. Until now, $\mathcal{U}_1, \ldots, \mathcal{U}_{n-2}$ have been waiting for the broadcast in step (2), which $\mathcal{M}$ now provides in the form of $g_n \cdot C_{n-1}$.
8. $\mathcal{U}_i$, $i = 1 \ldots, n-2$, go to step (4) and send back $g_i^{-1} g_n \cdot C_{n-1}$, which $\mathcal{M}$ intercepts.
9. In step (5), $\mathcal{M}$ broadcasts to $\mathcal{U}_i$, $i = 1 \ldots, n-2$, the message

$$\{\hat{g} g_1^{-1} g_n \cdot C_{n-1}, \hat{g} g_2^{-1} g_n \cdot C_{n-1}, \ldots, \hat{g} g_{n-1}^{-1} g_n \cdot C_{n-1}, g_n \cdot C_{n-1}\}$$

   User $\mathcal{U}_{n-1}$ is sent the same message, but the last element, $g_n \cdot C_{n-1}$ is substituted by C_{n-1}.
10. The users $\mathcal{U}_1, \ldots, \mathcal{U}_{n-2}$ now all compute the shared secret $K = g_i \hat{g} g_i^{-1} g_n \cdot C_{n-1}$.

Let us make some comments on the attack introduced above. First, we can observe that at the end of this procedure, all users as well as the attacker share the same key

$$K = \left(\prod_{j=1}^{n} g_j\right) \cdot (\hat{g} \cdot s).$$

Any passive observer will still be unable to determine the key, for the same reason that the original protocol is secure against passive attacks, cf. [4, Theorem 2.1], whose proof also applies to the general setting given in Section 2 whenever the action is transitive and the Diffie-Hellman problem is hard.

The attacker's secret $\hat{g}$ is not strictly required for the attack to work, but without it, the users may notice that something is amiss. Namely, in step (e), if we leave out $\hat{g}$, the user $\mathcal{U}_n$ may notice that $\mathcal{M}$ sent the same value C_{n-1} as in step (d). Similarly, in step (i), the other users could notice that the attacker just returned their transmission from (h). Using $\hat{g}$, however, the users should be unable to tell the difference between a regular execution of the protocol and the attack, again as a consequence of [4, Theorem 2.1].

As in the Initial Key Agreement (IKA) protocol introduced in Section 2, the broadcast element $g_n \cdot C_{n-1}$ is added at the end of the message in (i) in view of future rekeying operations and is not needed by any of the users $\mathcal{U}_1, \ldots, \mathcal{U}_{n-2}$ to recover the shared key. Note that users $\mathcal{U}_i$, $i = 1, \ldots, n-2$, expect that the last element of the message sent in step (i) is the one broadcast in step (2) of the protocol, which the attacker substitutes precisely by $g_n \cdot C_{n-1}$. In the case of user $\mathcal{U}_{n-1}$, who is also expecting the element sent in step (2) of the protocol, the element that $\mathcal{M}$ sends in step (b) is C_{n-1}. If this is not satisfied, the users might notice that something is wrong.

4. An exit strategy

After the attack of Section 3, the attacker $\mathcal{M}$ shares the key with the users $\mathcal{U}_1, \ldots, \mathcal{U}_n$ and can listen in on their conversation without any further active measures. However, at some point after that, the users may wish to execute an AKA operation, which is to say a key refreshment, the addition of a new member to the group, etc. as described in [4, Section 5]. After this point, the attacker can certainly no longer listen to the conversation. Even worse, the values the users remember from step (5) of the protocol are substantially different from normal, and any key refresh operation will thus fail completely, alerting the users about the attack.

In what follows, we will describe how the attacker can avoid being noticed by forging key refresh operations herself, assuming that any user may initiate a key refreshment at any time.

First, we recall the key refresh operation after a regular execution of IKA.2, adapted from [4, Section 5.6]. Suppose user $\mathcal{U}_c$ wishes to initiate a key refreshment. He remembers from step (5) of the key agreement protocol the values $\{E_1, \ldots, E_n\}$, where $E_k = \left(\prod_{j=1, j\neq k}^{n} g_j\right) \cdot s$, $k = 1, \ldots, n$. He picks a new secret $g_c' \in G$ and broadcasts

$$\{g_c' \cdot E_1, \ldots, g_c' \cdot E_{c-1}, E_c, g_c' \cdot E_{c+1}, \ldots, g_c' \cdot E_n\}.$$

Now, all users can compute the new key $g_c' \cdot C_n = g_c' \cdot \left(\prod_{j=1}^{n} g_j\right) \cdot s$. User $\mathcal{U}_c$ also replaces his own secret with $g_c' g_c$, and everyone replaces the information remembered from step (5) with this new broadcast.

Remark 4.1. *One important detail to note is that when $\mathcal{U}_c$ initiates the key refreshment, the value E_c he sends in position c is unchanged and already known to the other users. Hence, if $\mathcal{M}$ wishes to forge a key refreshment coming from $\mathcal{U}_c$, she has to make sure that each user receives in position c the value he previously held there. Otherwise, the attack could be discovered.*

Suppose now that the attacker $\mathcal{M}$ has just executed the attack from Section 3. Instead of $\{E_1, \ldots, E_n\}$, the users now remember the following values:

- For $i = 1, \ldots, n-2$, $\mathcal{U}_i$ remembers $\{\hat{g} \cdot E_1, \ldots, \hat{g} \cdot E_{n-1}, C_n\}$.
- $\mathcal{U}_{n-1}$ remembers $\{\hat{g} \cdot E_1, \ldots, \hat{g} \cdot E_{n-1}, E_n\}$.
- $\mathcal{U}_n$ remembers $\{g_n \cdot m_1, \ldots, g_n \cdot m_{n-3}, E_{n-1}, C_n, \hat{g} \cdot E_n\}$.

Evidently, if some user tries to initiate a key refreshment with these values, the operation will fail. However, $\mathcal{M}$ can bring the users into a consistent state by forging two key refresh operations herself. For this, she needs to still have control over the communications of $\mathcal{U}_{n-1}$ and $\mathcal{U}_n$, as in the original attack.

First, $\mathcal{M}$ picks two new random values $\hat{f}$ and $\hat{h} \in G$. Then, she forges a key refresh operation by sending the following values to the different users:

- To $\mathcal{U}_i$, $i = 1, \ldots, n-2$, she sends
$$\{\hat{h}\hat{g} \cdot E_1, \hat{h}\hat{g} \cdot E_2, \ldots, \hat{h}\hat{g} \cdot E_{n-2}, \hat{g} \cdot E_{n-1}, \hat{f}\hat{h}\hat{g} \cdot E_n\},$$
pretending it came from $\mathcal{U}_{n-1}$.
- To $\mathcal{U}_{n-1}$, she sends
$$\{\hat{f}\hat{h}\hat{g} \cdot E_1, \hat{h}\hat{g} \cdot E_2, \ldots, \hat{h}\hat{g} \cdot E_{n-2}, \hat{h}\hat{g} \cdot E_{n-1}, E_n\},$$
pretending it came from $\mathcal{U}_n$.
- To $\mathcal{U}_n$, she sends
$$\{\hat{f}\hat{h}\hat{g} \cdot E_1, \hat{h}\hat{g} \cdot E_2, \ldots, \hat{h}\hat{g} \cdot E_{n-2}, C_n, \hat{h}\hat{g}E_n\},$$
pretending it came from $\mathcal{U}_{n-1}$.

After this, the users will agree on the shared key $\hat{h}\hat{g} \cdot C_n$, which is also known to $\mathcal{M}$. As remarked above, if a user is made to believe that he received a key refreshment from $\mathcal{U}_c$, he must receive in position c the value he already held there.

Now, the values held by the users are still inconsistent, so $\mathcal{M}$ has to forge a second key refreshment:

- To $\mathcal{U}_i$, $i = 1, \ldots, n-2$, she sends
$$\{\hat{f}\hat{h}\hat{g} \cdot E_1, \ldots, \hat{f}\hat{h}\hat{g} \cdot E_n\},$$
pretending it came from $\mathcal{U}_n$.
- To $\mathcal{U}_{n-1}$ and $\mathcal{U}_n$, she sends
$$\{\hat{f}\hat{h}\hat{g} \cdot E_1, \ldots, \hat{f}\hat{h}\hat{g} \cdot E_n\},$$
pretending it came from $\mathcal{U}_1$.

Now, all users and the attacker agree on the shared key $\hat{f}\hat{h}\hat{g} \cdot C_n$. Furthermore, all users remember the same consistent values for key refreshment. If in the future any user initiates a key refreshment or other AKA operation, the attacker will lose access to the key, but the operation itself will work out without problem and without the users noticing anything wrong.

Remark 4.2. *An alternative course of action for $\mathcal{M}$ is to convert the attack into a regular man-in-the-middle attack on $\mathcal{U}_n$ at the time of the first key refreshment. For this, note that given the values each user remembers, a key refreshment initiated by $\mathcal{U}_c$, $c \leq n-2$, works well for all users but $\mathcal{U}_n$. The attacker can then intercept the broadcast arriving at $\mathcal{U}_n$ and replace it with random values, except that at position n she sends $\hat{h} \cdot E_n$ for some random $\hat{h} \in G$, and at position c she sends $g_n \cdot m_c$, which she knows from step (f) of the attack. Then, $\mathcal{M}$ will have the key $\hat{g}g'_c \cdot C_n$ in common with $\mathcal{U}_i$, $i \leq n-1$, as well as $\hat{h} \cdot C_n$ with $\mathcal{U}_n$. From then on, she can run a regular man-in-the-middle attack. A similar attack can be carried out if $\mathcal{U}_n$ initiates a key refreshment, but not if $\mathcal{U}_{n-1}$ does so. In this case, the attacker can intercept and apply $\hat{g}$ to the message for $\mathcal{U}_n$ so that all users agree on a common key without noticing the previous attack.*

5. Conclusions

We have presented an active attack against Cliques, one of the most popular multiparty key exchange protocols currently known. By assuming control of the communications of the last two users in the initial key agreement process, an attacker can eavesdrop into the subsequent communications between legitimate users without the need of a regular man-in-the-middle attack. The attacker is also able to leave the group unnoticed when later key renewals do not allow further eavesdropping.

References

[1] P. P. C. Lee, J. C. S. Lui, D. K. Y. Yau, Distributed collaborative key agreement and authentication protocols for dynamic peer groups, IEEE/ACM Trans. Netw. 14(2) (2006) 263–276.
[2] J. A. López-Ramos, J. Rosenthal, D. Schipani, R. Schnyder, Group key management based on semigroup actions, arxiv.org/pdf/1509.01075v1.pdf.
[3] O. Pereira, J. -J. Quisquater, Generic insecurity of cliques-type authenticated group key agreement protocols, Proc. 17th IEEE Computer Security Foundations Workshop, 16–29, 2004.
[4] M. Steiner, G. Tsudik, M. Waidner, Key agreement in dynamic peer groups, IEEE Trans. Parallel Distrib. Systems 11(8) (2000) 769–780.
[5] J. Van der Merwe, D. Dawoud, S. McDonald, A survey on peer-to-peer key management for mobile ad hoc networks, ACM Computing Surveys 39(1) (2007) 1–45.